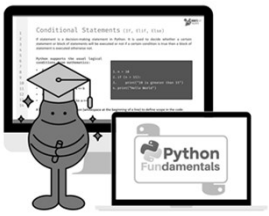
COURSE STUDENT ZONE

This is Robots Got Talents Students Zone, here you will find all the course topics, activities and explanation guides, We wish you enjoy the course, Good Luck

Lesson One	Lesson Two
Lesson Three	Lesson Four
Lesson Five	Lesson Six
Lesson Seven	Bonus Projects



ROBOTS GOT TALENTS™ CLASSROOM COURSES

This Course is created and published by Robots Got Talents, RGT is a volunteer-based Educational Platform founded in 2017 aiming to spread Robotics and Coding Knowledge all over the world by creating and publishing online free STEM educational content for Educational institutes and Individuals. In the past years RGT curriculums were used by hundreds of schools over in 90 countries. Learn More About RGT

THIS COURSE IS AVAILABLE ON:

CLASSROOM ROBOTICS
BY ROBOTS GOT TALENTS

ROBOAPP
ACTIVATING CREATIVITY MODE

ROBOTS GOT TALENTS

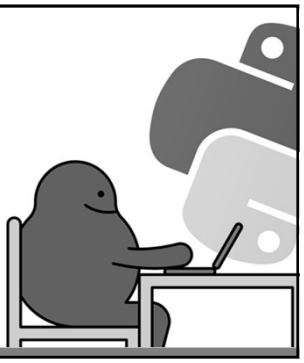
COURSE INTRODUCTION:

Python Fundamentals is a free classroom course developed by Robots Got Talents. Throughout this course students will learn the basics of programming using the python programming language, with +25 interactive programming exercises, 6 online quizzes, +30 example programs and 5 Bonus Projects. The course is a built to be an introductory course for general-purpose python programming and would be followed by more specialized Python resources. After finishing the course content participants would be able to create their own simple python projects.

- Introduction to Algorithms
- Programming & Programming Languages
- What is Python
- Data types in Python
- Built-in Functions
- Comments
- Variables in Python
- Casting functions
- Mathematical Operators
- Math functions
- Augment Assignment Operators
- String functions
- Conditional Statements
- Loops
- Lists in Python
- Creating functions
- Libraries and modules.
- Understanding Libraries Functions
- Using Python Turtle Library
- Bonus Projects

LESSON ONE

- Introduction to Algorithms
- Introduction to Programming
- Programming Languages
- What is Python
- Python data types
- Built-in functions
- Exercises 1 – 3
- Comments



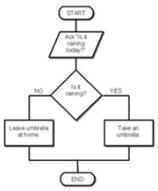
INTRODUCTION TO ALGORITHMS:

An algorithm is a procedure used for solving a problem or performing a computation. Algorithms act as an exact list of instructions that conduct specified actions step by step in either hardware- or software-based routines.

- Algorithms are not just related to Programming or Computer Science they are everywhere. A recipe for making food is an algorithm, the method used to solve differentiation, integration or long division problems are all considered algorithms.

Presenting Algorithms

FLOWCHART



PSEUDOCODE

1. PRINT "Please enter the length of the side in cm "
2. INPUT Side
3. Area = Side * Side
4. PRINT Area " cm^2"

1. PRINT "Please enter the length of the side in cm "
2. INPUT Side
3. Area = Side * Side
4. PRINT Area " cm^2"

INTRODUCTION TO PROGRAMMING:

Programming is the process of creating a set of instructions that tell a device (robot, machine, computer, or any smart device) how to perform a task. Programs are created to implement algorithms .

Programming Languages:

A programming language is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks. There are 2 main types of programming languages;

- High Level Programming Language
- Low Level Programming Language

TYPES OF PROGRAMMING LANGUAGES:

100100
10000000011
101010101010
0000000001111

Low Level Language (LLL)

Ex. Machine Code, Assembly Code

```
if(i<5)
{
printf("I am true block ");
}
else{
printf("I am false block");
}
```

High Level Language (HLL)

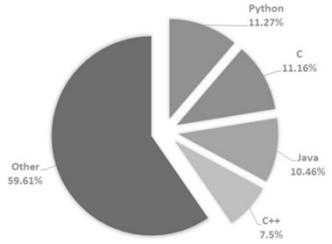
Ex. Python, C, Java, C++, C#

High Level Language	Low Level Language
It is programmer friendly language	It is a machine friendly language
High level language is less memory efficient.	Low level language is high memory efficient
It is easy to understand, Learn and Debug	It is tough to understand, Learn and Debug
It is portable	It is non-portable.
It can run on any platform	It is machine-dependent
It needs compiler or interpreter for translation	It needs assembler for translation.

MOST USED HIGH LEVEL PROGRAMMING LANGUAGES

August 2020- August 2021

Source: www.tiobe.com



WHAT IS PYTHON?

Python is a general purpose high-level programming language developed by Guido van Rossum in 1991, which is designed to be highly readable, with a very simple syntax.

- Easy to Read, Learn and Write
- Interpreted Language
- Dynamically Typed
- Free and Open-Source
- Vast Libraries Support
- Portability

code

```
a = 1
while a < 10:
 print (a)
 a += 2
```

output

variables

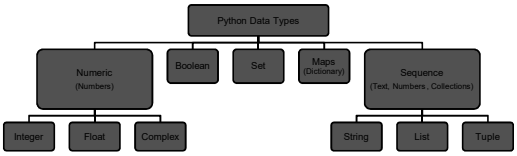
COMPAGNIES & PLATFORMS USING PYTHON:

GETTING STARTED:

Since we will only be creating simple projects in the next lessons, instead of installing a python IDE (Integrated Development Environment) to your device, You can use the online IDE available in the student zone, or the online python IDE by Programiz, throughout the course, if you want to try any project press this button to open Programiz online python compiler.

DATA TYPES

Data types are the classification of data items. It represents the type of value that tells what operations can be performed on a particular data. As presented in the figure below there are 5 main types of data in python: Numeric, Boolean, Set, Dictionary and Sequence.



Numeric Values (Numbers)

- **Integers:** This data type is represented with the help of int class. It consists of positive or negative whole numbers (without fraction or decimal).

```
2  -12  13294  0345
```

- **Float:** This type is represented by the float class. It is a true number with floating-point representation. It is specified by a decimal point.

```
2.8  -30.6  4.00  0.55
```

- **Complex Numbers:** Complex numbers are represented by complex classes. It is specified as (real part) + (imaginary part)

```
3+6j    6-15j   4+2j
```

Sequence Values (Text data)

- **String:** A string is a collection of one or more Unicode characters put in a single quote, double quote or triple quote.

```
"A"  "Python"  "Robots Got Talents"  "RGT 2022"
```

- **List:** A List is an ordered sequence of changeable items (Variables), starting from 0 to n and separated by a comma (,). A list could include items of different types.

```
[1, 2.2, 'python']    ["Programming", "Robotics", "RGT"]    [2,4,6]
```

- **Tuple:** A tuple is an ordered sequence of unchangeable items (Constants), starting from 0 to n and separated by a comma (,). A Tuple could include items of different types

```
[1, 2.2, 'python']    ["Programming", "Robotics", "RGT"]    [2,4,6]
```

Boolean

Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are (true), and those equal to False are (false)

Set

A Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces ({ }).

```
{5,2,3,1,4}
```

Dictionary

Dictionary is an unordered collection of data values, which is used to store data values like a map, which, unlike other Data Types that hold only a single value as an element, a Dictionary consists of key-value pair. Key-value is provided within the dictionary to form it more optimized. In the representation of a dictionary data type, each key-value pair during a Dictionary is separated by a colon: whereas each key's separated by a comma (,).

```
{1:'Hello',2:5.5, 3:'World'}
```

Built-in Functions (General)

A function is a block of organized, reusable code that is used to perform a single, related action. Each version of Python has its own built-in functions which could also be known as statements, you can also create your own functions in your code. The latest version of Python has over 60 predefined (built-in) functions. Below we will cover 5 useful Built-in Functions.

- **input()** reads and return a line of string
- **print()** prints the specified message on the screen
- **type()** returns the type of the specified object
- **abs()** returns the absolute value (|X|) of the specified number
- **pow()** returns the result of two specified values one to the power of another (X^y)

```
#Built in functions
input("Enter your name: ")
print("Welcome")
type(1.5)
abs(-1299)
pow(2,3)
```

Syntax (Built-in Functions)

```
input()
input("Input Message")
#The Input Message is optional and makes the program more user friendly
```

```
print("Message")
```

```
type(x) #x could be anything
```

```
abs(n) #n could be a positive or negative number
```

```
Pow(x,y) #x and y must be numbers ( x to the power of y)
```

Example 1

main.py

```
input("Enter your name: ")
print("Welecome")
type(1.5)
abs(-1299)
pow(2,3)
```

Shell

```
Enter your name: Youssef
Welecome
>
```

Example 2

main.py

```
input("Enter your name: ")
print("Welecome")
print(type(1.5))
print(abs(-1299))
print(pow(2,3))
print(pow(abs(-1299),2))
#Combined Functions
```

Shell

```
Enter your name: Youssef
Welecome
<class 'float'>
1299
8
1687401
>
```

EXERCISE ONE

Using the functions covered in this lesson, Create a program to print your personal identification details when you input anything. **Do not use more than 7 lines of code**.

```
ID CARD

Name: your name
Age: your age
Country: your country
ID: ID formula (age power of 2 * 365)
```

Notes: use * for multiplication

```
Type in anything to print your id: i
ID CARD

Name: John Pinkman
Age: 34
Country: UK
ID:  421940
```

EXERCISE ONE SOLUTION

```
1. input("Type in anything to print your id: ")
2. print("ID CARD\n")
3. print("Name: John Pinkman")
4. print("Age: 34")
5. print("Country: UK")
6. print("ID: ",365*pow(34,2))
```

main.py | Run | Shell

```
1  input("Type in anything to print your id: ")
2  print("ID CARD\n")
3  print("Name: John Pinkman")
4  print("Age: 34")
5  print("Country: UK")
6  print("ID: ",365*pow(34,2))
7
8
```

```
Type in anything to print your id: i
ID CARD

Name: John Pinkman
Age: 34
Country: UK
ID:  421940
>
```

EXERCISE TWO

Create a program to do as shown in the shell

Notes:

- Area of square = side * side
- Area of rectangle = length * width
- Area of circle = π *radius * radius (π=3.14)

Shell | Clear

```
AREA OF SHAPES:
```

EXERCISE TWO SOLUTION

```
1. input("AREA OF SHAPES: ")
2. print("Square (side=2)")
3. print("Area = ",pow(2,2))
4. print("Rectangle (Length=3, Width=4)")
5. print("Area = ",3*4 )
6. print("Circle (Raduis=4)")
7. print("Area = ",3.14*pow(4,2)))
```

main.py | Run | Shell | Clear

```
1 input("AREA OF SHAPES: ")
2 print("Square (side=2)")
3 print("Area = ",pow(2,2))
4 print("Rectangle (Length=3, Width=4)")
5 print("Area = ",3*4 )
6 print("Circle (Raduis=4)")
7 print("Area = ",3.14*pow(4,2)))
8
```

```
AREA OF SHAPES: q
Square (side=2)
Area =  4
Rectangle (Length=3, Width=4)
Area =  12
Circle (Raduis=4)
Area =  50.24
>
```

EXERCISE THREE

Fix all the errors in this code:

```
1 Input("Type in anything to start")
2 print("Welcome\n)
3 print(pow(2.2))
4 input("Press Any button to contiue")
5 Print(type(3.124))
6 print(abs(-10233)
```

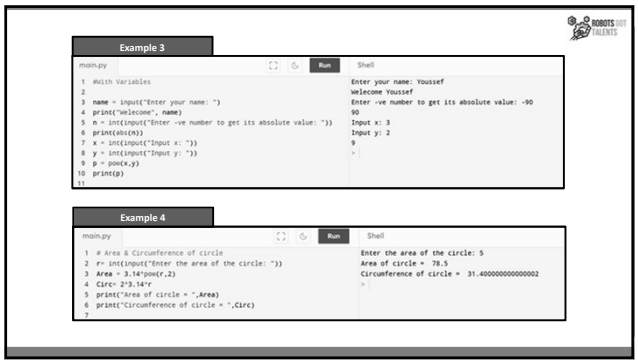
EXERCISE THREE SOLUTION

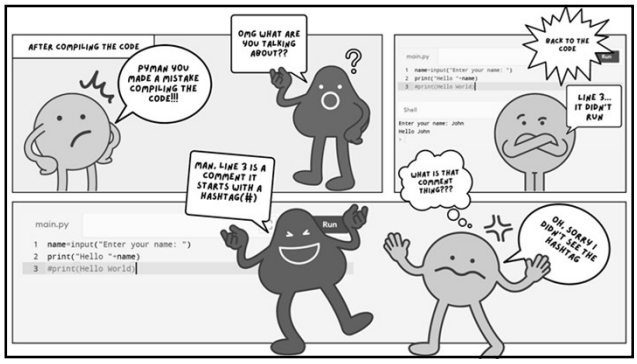
```
1. input("Type in anything to start")
2. print("Welcome\n")
3. print(pow(2,2))
4. input("Press Any button to continue")
5. print(type(3.124))
6. print(abs(-10233))
```

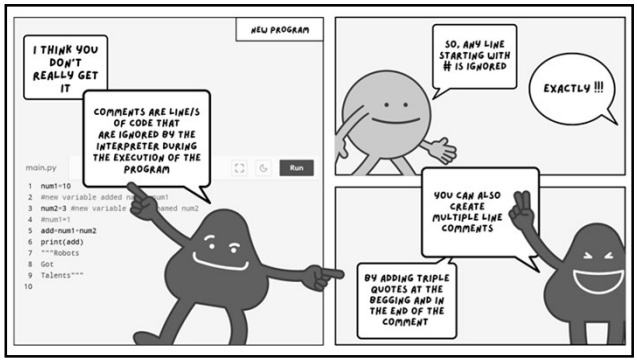
main.py | Run | Shell | Clear

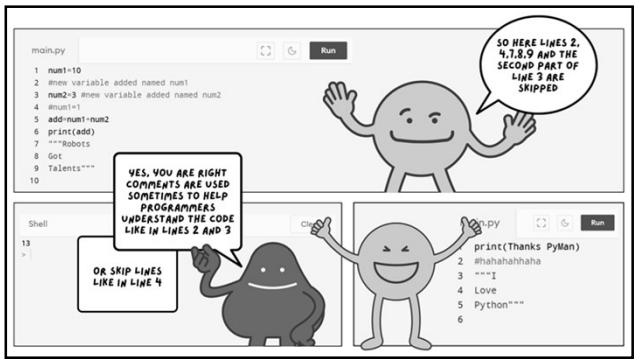
```
1 input("Type in anything to start")
2 print("Welcome\n")
3 print(pow(2,2))
4 input("Press Any button to contiue")
5 print(type(3.124))
6 print(abs(-10233))
7
8
```

```
Type in anything to start0
Welcome

4
Press Any button to contiueK
<class 'float'>
10233
>
```


Example 3
main.py
Run
Shell
1 #With Variables
2
3 name = input("Enter your name: ")
4 print("Welecome", name)
5 n = int(input("Enter -ve number to get its absolute value: "))
6 print(abs(n))
7 x = int(input("Input x: "))
8 y = int(input("Input y: "))
9 p = pow(x,y)
10 print(p)
11
Enter your name: Youssef
Welecome Youssef
Enter -ve number to get its absolute value: -90
90
Input x: 3
Input y: 2
9
Example 4
main.py
Run
Shell
1 # Area & Circumference of circle
2 r= int(input("Enter the area of the circle: "))
3 Area = 3.14*pow(r,2)
4 Circ= 2*3.14*r
5 print("Area of circle = ",Area)
6 print("Circumference of circle = ",Circ)
7
Enter the area of the circle: 5
Area of circle = 78.5
Circumference of circle = 31.400000000000002


AFTER COMPILING THE CODE
PYMAN YOU MADE A MISTAKE COMPILING THE CODE!!!
OMG WHAT ARE YOU TALKING ABOUT??
BACK TO THE CODE
LINE 3... IT DIDN'T RUN
MAN, LINE 3 IS A COMMENT IT STARTS WITH A HASHTAG(#)
WHAT IS THAT COMMENT THING???
OH, SORRY I DIDN'T SEE THE HASHTAG
main.py
Run
1 name=input("Enter your name: ")
2 print("Hello "+name)
3 #print(Hello World)


NEW PROGRAM
I THINK YOU DON'T REALLY GET IT
COMMENTS ARE LINE/S OF CODE THAT ARE IGNORED BY THE INTERPRETER DURING THE EXECUTION OF THE PROGRAM
SO, ANY LINE STARTING WITH # IS IGNORED
EXACTLY !!!
YOU CAN ALSO CREATE MULTIPLE LINE COMMENTS
BY ADDING TRIPLE QUOTES AT THE BEGGING AND IN THE END OF THE COMMENT
main.py
Run
1 num1=10
2 #new variable added num1
3 num2=3 #new variable named num2
4 #num1=1
5 add=num1+num2
6 print(add)
7 """Robots
8 Got
9 Talents"""
10



Comments

Comments are line/s of code that are ignored by the interpreter during the execution of the program, or in simpler terms, they are lines that do not affect the program. Comments enhance the readability of the code and help the programmers to understand the code very carefully.

main.py — Run

```
#user input items of list
# creating an empty list
lst = []
# number of elements as input
n = int(input("Enter number of elements : "))
# iterating till the range
i=0
while i <n:
    ele = int(input("Enter element: "))
    lst.append(ele)
    i=i+1
print(lst)
```

Shell

```
Enter number of elements : 4
Enter element: 6
Enter element: 3
Enter element: 9
Enter element: 12
[6, 3, 9, 12]
>
```

Syntax (Comments)

To create a single line comment add a hashtag (#) at the beginning of the line. Python also supports multiple line comments by adding triple quotes(""") at the begging and in the end of the multiple line comment.

Single Line Comment:

```
#Hello There
#Single Comment 2
```

Single Line Comment:

```
"""Multimple
line comments all the
lines between the triple quotes
would be ignored"""
```

Examples of Comments

Example 1

```
#This is my first python program

#SingleLineComment
Platform = "VEX VR"
Year = 2022

"""Multimple
line comments all the
lines between the triple quotes
would be ignored"""

print("I Love Python & Robotics") #Output Statement
print("Let's Start Using ",Platform)

#print("Open: https://vr.vex.com/")

print("This course was created in ",Year)
```

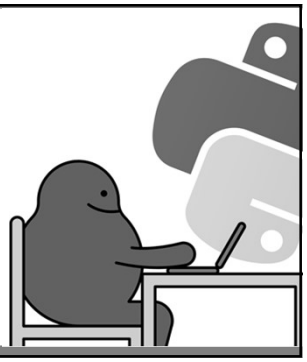
```
I Love Python & Robotics
Let's Start Using  VEX VR
This course was created in  2022
>>>
```

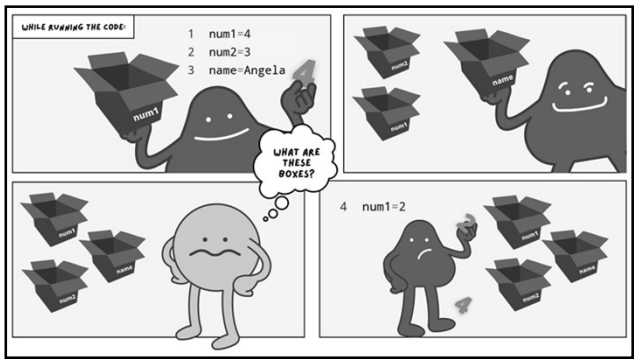
Python Fundamentals

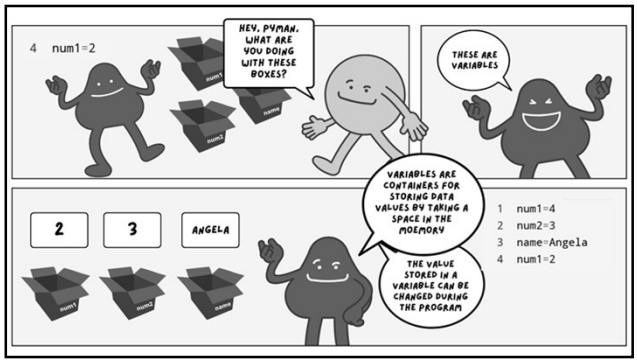
QUIZ ONE

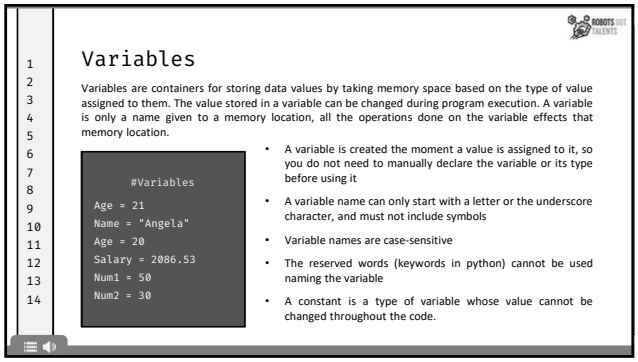
LESSON TWO

- Variables in Python
- Casting functions
- Mathematical Operators
- Math & String functions
- Exercises 4 – 5
- Conditional Statements (Part 1)
- Exercises 6 – 8









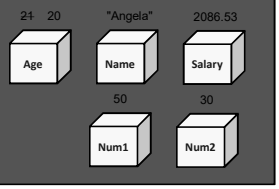
Variables

Variables are containers for storing data values by taking memory space based on the type of value assigned to them. The value stored in a variable can be changed during program execution. A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.

```
#Variables
Age = 21
Name = "Angela"
Age = 20
Salary = 2086.53
Num1 = 50
Num2 = 30
```

- A variable is created the moment a value is assigned to it, so you do not need to manually declare the variable or its type before using it
- A variable name can only start with a letter or the underscore character, and must not include symbols
- Variable names are case-sensitive
- The reserved words (keywords in python) cannot be used naming the variable
- A constant is a type of variable whose value cannot be changed throughout the code.

```
#Variables
1. Age = 21
2. Name = "Angela"
3. Age = 20
4. Salary = 2086.53
5. Num1 = 50
6. Num2 = 30
```



- String variables can be declared either by using single (‘’) or double ("") quotes
- Python allows adding different values in a single line with ","operators
- The type of a variable could be specified or converted via casting built-in functions
- Built-in and Created functions could work with variables

Syntax (Variables)

```
x = 50
y = "John"
z = 18.54
#x, y and z are variable names and 50,John and 18.54 are variable values
```

```
x, y, z= 50,"John",18.54     #Adding different values in a single line
```

```
a, b, c = 30   #Variables a, b and c all have the value 30 assigned to them
```

Casting Functions

```
x = str(50)   # Converts value to string. x will be ‘50‘ as a string
y = int(3.6)  # Converts value to integer. y will be 3
z = float(3)  # Converts value to float. z will be 3.0
```

Example 1

main.py

```
name = input("Enter Your Name: ")
print("\nHello ", name, "\nWelcome to robotsgottalents.com")
# (\n Starts a new line)
```

Shell

```
Enter Your Name: Youssef
Hello  Youssef
Welcome to robotsgottalents.com
>
```

Example 2

main.py

```
print("Basic Calculator")
A = int(input("Input Num A:"))
B = int(input("Input Num B:"))
Sum=A+B
Sub=A-B
Mul=A*B
Divs=A/B
print("\n",A,"+",B,"=",Sum)
print(A,"-",B,"=",Sub)
print(A,"*",B,"=",Mul)
print(A,"/",B,"=",Divs)
```

Shell

```
Basic Calculator
Input Num A:8
Input Num B:10
8 + 10 = 18
8 - 10 = -2
8 * 10 = 80
8 / 10 = 0.8
>
```

```
Useful Mathematical Operators
z = x+y #+ Addition   a = 7+3 #a=10

z = x-y #- Subtraction   a = 7-3 #a=4

z = x*y #* Multiplication   a = 7*3 #a=21

z = x/y #/ Division (Float)   a = 7/3 #a=2.333333333333335

z = x//y #// Division (Integer)   a = 7//3 #a=2

z = x%y #% Modulus    a = 7%3 #a=1
```

Example 3

main.py

```
print("Basic Calculator")
A = int(input("Input Num A:"))
B = int(input("Input Num B:"))
A = int(A)
B = int(B)
Sum=A+B
Sub=A-B
Mul=A*B
Divs=A/B
print("\n",A,"+",B,"=",Sum)
print(A,"-",B,"=",Sub)
print(A,"*",B,"=",Mul)
print(A,"/",B,"=",Divs)
```

Shell

```
Basic Calculator
Input Num A:10
Input Num B:9
10 + 9 = 19
10 - 9 = 1
10 * 9 = 90
10 / 9 = 1.1111111111111112
>
```

Example 4

main.py

```
a = int(input("Input a: "))
b = int(input("Input b: "))
c = int(input("Input c: "))
x1= (-b+pow((pow(b,2)-4*a*c),0.5))/2*a
x2= (-b-pow((pow(b,2)-4*a*c),0.5))/2*a
print("Solution = ",x1,"or",x2)
#quadretic formula
```

Shell

```
Input a: 10
Input b: 40
Input c: 3
Solution =  -7.646159383286566 or -392.3538406167134
>
```

```
Useful Math Functions (#include math)
math.degrees()  #Converts an angle from radians to degrees
math.sin()      #Returns the sine of a number
math.cos()      #Returns the cosine of a number
math.tan()      #Returns the tangent of a number
math.floor()    #Rounds a number down to the nearest integer
math.ceil()     #Rounds a number up to the nearest integer
```

```
Useful Augment Assignment Operators
x+=5 # x=x+5
x-=5 # x=x-5
x*=5 # x=x*5
x/=5 # x=x/5
x%=5 # x=x%5
```

```
Useful String Functions [string.function()]
.upper() #Returns the uppercase version of a string
.lower() #Returns the lowercase version of a string
.swapcase() #Swaps the uppercase to lowercase and lowercase to uppercase
.capitalize() #Returns the first letter as uppercase and rest is lowercase
.title( ) #Returns the first character of every word capitalized
.count( ) #Finds the number of times a specified value, appears in a string
.find( ) #Finds the first occurrence (index) of the specified value
.replace("x","y") #Replaces a specified phrase with another specified phrase
.join( ) #Connects all items in an iterable and joins them into one string
```

EXERCISE FOUR

Using the functions covered in this lesson and variables, Create a program to print the sin, cos, and tan values (as float) of an inputted angle.

Notes: use math functions, import math library first (import math)

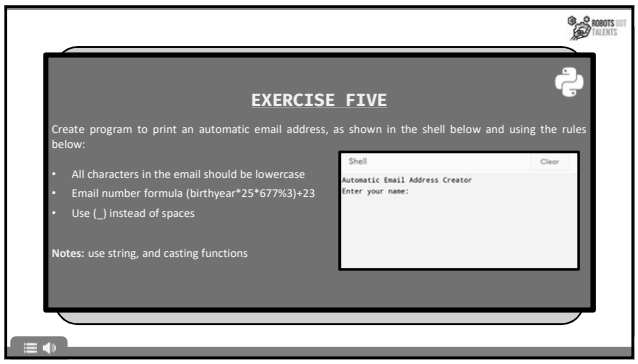
EXERCISE FOUR SOLUTION

```
1. import math
2. angle=int(input("Enter angle:"))
3. print(math.sin(angle))
4. print(math.cos(angle))
5. print(math.tan(angle))
```

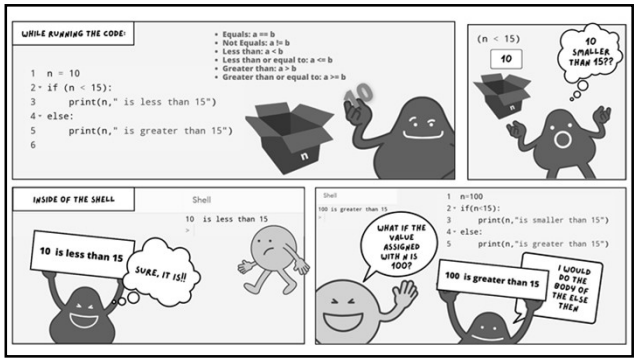
main.py | Run | Shell

```
import math
angle=int(input("Enter angle: "))
print(math.sin(angle))
print(math.cos(angle))
print(math.tan(angle))
```

```
Enter angle: 67
-0.8555199789753223
-0.5177697997895051
1.6523172640102353
>
```


EXERCISE FIVE
Create program to print an automatic email address, as shown in the shell below and using the rules below:
All characters in the email should be lowercase
Email number formula (birthyear*25*677%3)+23
Use (_) instead of spaces
Notes: use string, and casting functions
Shell
Clear
Automatic Email Address Creator
Enter your name:


1. print("Automatic Email Address Creator")
2. name=input("Enter your name: ")
3. birthyear=int(input("Enter your birth year: "))
4. birthyear=(birthyear*25*677%3)+23
5. birthyear=str(birthyear)
6. lname=name.lower()
7. lname=lname.replace(" ","_")
8. email=lname+birthyear+"@robotsgottalents.com"
9. print("New Email: "+email)
main.py
Run
Shell
Clear
Automatic Email Address Creator
Enter your name: Roben torenk
Enter your birth year: 2000
New Email: roben_torenk24@robotsgottalents.com


WHILE RUNNING THE CODE:
n = 10
if (n < 15):
print(n," is less than 15")
else:
print(n," is greater than 15")
Equals: a == b
Not Equals: a != b
Less than: a < b
Less than or equal to: a <= b
Greater than: a > b
Greater than or equal to: a >= b
(n < 15)
10
10 SMALLER THAN 15??
INSIDE OF THE SHELL
Shell
10 is less than 15
10 is less than 15
SURE, IT IS!!
100 is greater than 15
WHAT IF THE VALUE ASSIGNED WITH N IS 100?
n=100
if(n<15):
print(n,"is smaller than 15")
else:
print(n,"is greater than 15")
100 is greater than 15
I WOULD DO THE BODY OF THE ELSE THEN

Conditional Statements (If, Elif, Else)

if statement is a decision-making statement in Python. It is used to decide whether a certain statement or block of statements will be executed or not if a certain condition is true then a block of statement is executed otherwise not.

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

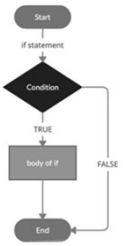
```
n = 10
if (n > 15):
    print("10 is greater than 15")
print("Hello World")
```

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code.

Syntax (IF Statement)

```
1. x = int(input("Enter the num x "))
2. y = int(input("Enter the num y "))
3. if (x > y):
4.     print("x is greater than y")
5.     print(x,">",y)
6. print("End")
```

```
1. Pass = input("Create Password\t")
2. print("Password Created")
3. Ipass = input("Enter Password: ")
4. if (Ipass == Pass):
5.     print("Correct Password")
6.     print("Welcome (:")
```



Example 1

main.py

```
x = int(input("Enter the num x "))
y = int(input("Enter the num y "))
if (x > y):
    print("x is greater than y")
    print(x,">",y)
print("End")
```

Condition is false

Shell

```
Enter the num x 10
Enter the num y 20
End
>
```

main.py

```
x = int(input("Enter the num x "))
y = int(input("Enter the num y "))
if (x > y):
    print("x is greater than y")
    print(x,">",y)
print("End")
```

Condition is true

Shell

```
Enter the num x 20
Enter the num y 3
x is greater than y
20 > 3
End
>
```

Example 2

main.py

```
Pass = input("Create Password\t")
print("Password Created")
Ipass = input("\nEnter Password: ")
if (Ipass == Pass):
    print("Correct Password")
    print("Welcome (:")
```

Condition is false

Shell

```
Create Password 20228910
Password Created

Enter Password: 20304
>
```

main.py

```
Pass = input("Create Password\t")
print("Password Created")
Ipass = input("\nEnter Password: ")
if (Ipass == Pass):
    print("Correct Password")
    print("Welcome (:")
```

Condition is true

Shell

```
Create Password 20228910
Password Created

Enter Password: 20228910
Correct Password
Welcome (:
>
```

EXERCISE SIX

With only three lines of codes, create a program to print (+ve) if the inputted number is positive.

Notes: use the if statement

EXERCISE SIX SOLUTION

```
n=int(input("Enter a number: "))
if(n>0):
    print("+ve")
```

main.py

```
n=int(input("Enter a number: "))
if(n>0):
    print("+ve")
```

Shell

```
Enter a number: -34
>
```

main.py

```
n=int(input("Enter a number: "))
if(n>0):
    print("+ve")
```

Shell

```
Enter a number: 20
+ve
>
```

EXERCISE SEVEN

Create a program to check if a word includes the letter a, then prints (word includes a):

Notes: use the if statement, and string functions

```
1. word=input("Enter a word: ")
2. check=(word.count("a"))
3. if(check>0):
4.     print(word, "includes the letter a")
```

main.py

```
1 word=input("Enter a word: ")
2 check=(word.count("a"))
3 if(check>0):
4     print(word, "includes the letter a")
```

Shell

```
Enter a word: car
car includes the letter a
>
```

```
1 word=input("Enter a word: ")
2 check=(word.count("a"))
3 if(check>0):
4     print(word, "includes the letter a")
```

```
Enter a word: apple
apple includes the letter a
>
```

EXERCISE EIGHT

Fix all the errors in this code:

```
1 nom = 3
2 if num > 0
3     Print(num, "is a positive number.");
4 Num = -1
5 if num > 0
6     print(num, "is a positive number.")
```

EXERCISE EIGHT

```
1. num = 3
2. if num > 0:
3.     print(num, "is a positive number.")
4. num = -1
5. if num > 0:
6.     print(num, "is a positive number.")
```

main.py | Run | Shell

```
1 num = 3
2 if num > 0:
3     print(num, "is a positive number.")
4 num = -1
5 if num > 0:
6     print(num, "is a positive number.")
```

```
3 is a positive number.
>
```

Python Fundamentals

QUIZ TWO

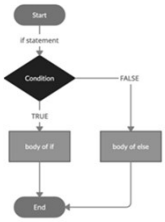
LESSON THREE

- Conditional Statements (Parts 2+3)
- Exercises 9 – 12
- Nested conditional statements
- Math & String functions
- Combined conditional statements
- Exercises 13 – 14

Syntax (IF Else Statement)

- If the condition defined is false the statement or block of statements after (else) will be executed. An indentation should also be added after the else.

```
age = int(input("Enter your age: "))
if (x >= 18):
    print("You can get your driving license")
else:
    print("Sorry you must be 18 or older")
```



Example 1

main.py

```
num=int(input("Input Number: "))
if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

Condition is true

Shell

```
Input Number: 1
Positive or Zero
>
```

main.py

```
num=int(input("Input Number: "))
if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

Condition is false

Shell

```
Input Number: -2
Negative number
>
```

EXERCISE NINE

In no more than 6 lines of code, create a program to ask the user to input two different numbers, and print which number is greater:

Notes: use the if else statements

```
1. num1=int(input("Input the first number: "))
2. num2=int(input("Input the first number: "))
3. if (num1>num2):
4.     print(num1, ">", num2)
5. else:
6.     print(num2 , ">", num1)
```

main.py | Run | Shell

```
1 num1=int(input("Input the first number: "))
2 num2=int(input("Input the first number: "))
3 if (num1>num2):
4     print(num1, ">", num2)
5 else:
6     print(num2 , ">", num1)
```

```
Input the first number: 3
Input the first number: 20
20 > 3
>
```

EXERCISE TEN

Create a program to check if a word includes a specific letter, as shown in the shell, if the condition is true print that it includes the letter if its false print that it doesn't:

Notes: use the if statement, and string functions

Shell | Clear

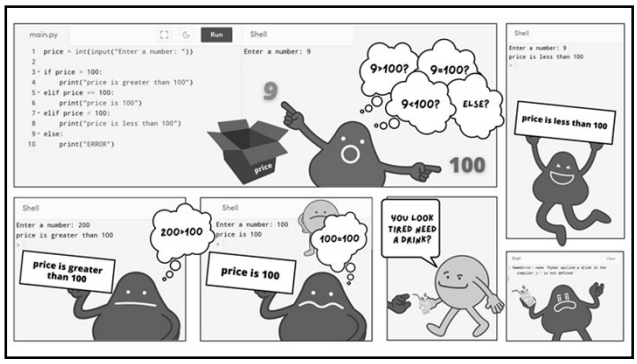
```
Input the letter to search for:
```

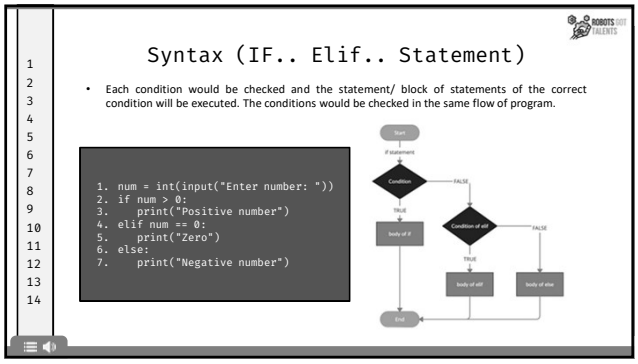
```
1. letter=input("Input the letter to search for: ")
2. word=input("Enter a word: ")
3. check=(word.count(letter))
4. if(check>0):
5.     print(word, "includes the letter", letter)
6. else:
7.     print(word, "does not include the letter", letter)
```

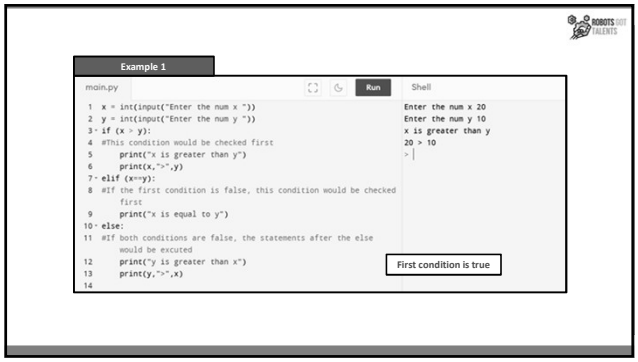
main.py | Run | Shell

```
1 letter=input("Input the letter to search for: ")
2 word=input("Enter a word: ")
3 check=(word.count(letter))
4 if(check>0):
5     print(word, "includes the letter", letter)
6 else:
7     print(word, "does not include the letter",
          letter)
```

```
Input the letter to search for: r
Enter a word: python
python does not include the letter r
>
```


main.py
Run
Shell
price = int(input("Enter a number: "))
if price > 100:
print("price is greater than 100")
elif price == 100:
print("price is 100")
elif price < 100:
print("price is less than 100")
else:
print("ERROR")
Enter a number: 9
9
price
100
9>100?
9=100?
9<100?
ELSE?
Enter a number: 9
price is less than 100
price is less than 100
Enter a number: 200
price is greater than 100
200>100
price is greater than 100
Enter a number: 100
price is 100
100=100
price is 100
YOU LOOK TIRED NEED A DRINK?


Syntax (IF.. Elif.. Statement)
Each condition would be checked and the statement/ block of statements of the correct condition will be executed. The conditions would be checked in the same flow of program.
1. num = int(input("Enter number: "))
2. if num > 0:
3. print("Positive number")
4. elif num == 0:
5. print("Zero")
6. else:
7. print("Negative number")
Start
if statement
Condition
FALSE
TRUE
body of if
Condition of elif
FALSE
TRUE
body of elif
body of else
End


Example 1
main.py
Run
Shell
x = int(input("Enter the num x "))
y = int(input("Enter the num y "))
if (x > y):
#This condition would be checked first
print("x is greater than y")
print(x,">",y)
elif (x==y):
#If the first condition is false, this condition would be checked first
print("x is equal to y")
else:
#If both conditions are false, the statements after the else would be excuted
print("y is greater than x")
print(y,">",x)
Enter the num x 20
Enter the num y 10
x is greater than y
20 > 10
First condition is true

Example 1

main.py

```
x = int(input("Enter the num x "))
y = int(input("Enter the num y "))
if (x > y):
#This condition would be checked first
    print("x is greater than y")
    print(x,">",y)
elif (x==y):
#If the first condition is false, this condition would be checked first
    print("x is equal to y")
else:
#If both conditions are false, the statements after the else would be excuted
    print("y is greater than x")
    print(y,">",x)
```

Shell

```
Enter the num x 4
Enter the num y 4
x is equal to y
>
```

Second condition is true

Example 1

main.py

```
x = int(input("Enter the num x "))
y = int(input("Enter the num y "))
if (x > y):
#This condition would be checked first
    print("x is greater than y")
    print(x,">",y)
elif (x==y):
#If the first condition is false, this condition would be checked first
    print("x is equal to y")
else:
#If both conditions are false, the statements after the else would be excuted
    print("y is greater than x")
    print(y,">",x)
```

Shell

```
Enter the num x 5
Enter the num y 20
y is greater than x
20 > 5
>
```

First & second conditions are false

Example 2

main.py

```
var1 = 1+2j
if (type(var1) == int):
    print("Type of the variable is Integer")
elif (type(var1) == float):
    print("Type of the variable is Float")
elif (type(var1) == complex):
    print("Type of the variable is Complex")
elif (type(var1) == bool):
    print("Type of the variable is Bool")
elif (type(var1) == str):
    print("Type of the variable is String")
elif (type(var1) == tuple):
    print("Type of the variable is Tuple")
elif (type(var1) == dict):
    print("Type of the variable is Dictionaries")
elif (type(var1) == list):
    print("Type of the variable is List")
else:
    print("Type of the variable is Unknown")
```

Shell

```
Type of the variable is Complex
>
>
>
```

EXERCISE ELEVEN

Create a program to print the RGB value of the color chosen by the user from the colors below:

- White RGB code = #FFFFFF
- Blue RGB code = #0000FF
- Red RGB code = #FF0000
- Green RGB code = #00FF00
- Gray RGB code = #808080
- Yellow RGB code = #FFFF00

```
print("RGB COLOR CODE GENERATOR: ")
color=input("Choose a color from White, Blue, Red, Green, Gray
    and Yellow: ")
color=color.lower()
if(color=="white"):
    print("White RGB code = #FFFFFF")
elif(color=="blue"):
    print("Blue RGB code = #0000FF")
elif(color=="red"):
    print("Red RGB code = #FF0000")
elif(color=="green"):
    print("Green RGB code = #00FF00")
elif(color=="gray"):
    print("Gray RGB code = #808080")
elif(color=="yellow"):
    print("Yellow RGB code = #FFFF00")
```

Example 3

main.py

```
score=int(input("Enter your score: "))
if(score>=90):
    print("A+")
elif(score>=80):
    print("A")
elif(score>=65):
    print("B")
elif(score>=50):
    print("C")
else:
    print("F")
```

Run

Shell

```
Enter your score: 60
C
>
```

EXERCISE_TWELVE

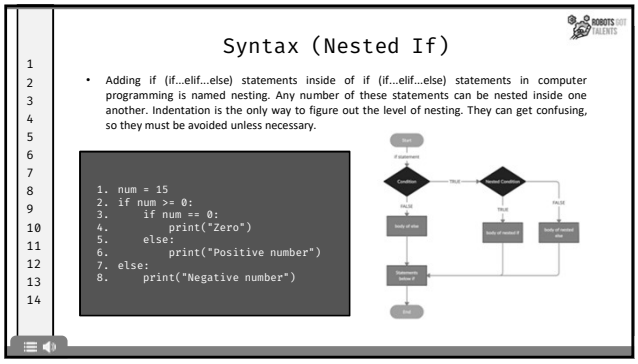
Create a program to decide whether the user can get driver license, or not:

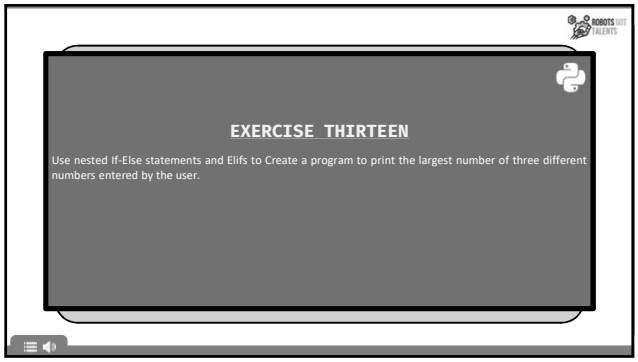
- National Driver License >18 years
- International Driver License >21 years
- Professional Driver License >30 years

Calculate the years left to get any non-valid driving licenses

```
age=int(input("Enter your age: "))
if(age<18):
    print("Sorry you can not get a driving license")
    print("You have",18-age, "years left to get a national license")
    print(",",21-age, "years left to get an international license")
    print("and",30-age ,"years left to get a professional license")
elif(age<21):
    print("You can get a national driving license")
    print(",",21-age, "years left to get an international license")
    print("and",30-age ,"years left to get a professional license")
elif(age<30):
    print("You can get an international & national driving license")
    print("and",30-age ,"years left to get a professional license")
elif(age>=30):
    print("You can get a national, international and professional driving license")
```


main.py
Run
Shell
age=int(input("Enter your age: "))
if(age<18):
print("Sorry you can not get a driving license")
print("You have",18-age, "years left to get a national license")
print(",",21-age, "years left to get an international license")
print("and",30-age ,"years left to get a professional license")
elif(age<21):
print("You can get a national driving license")
print(",",21-age, "years left to get an international license")
print("and",30-age ,"years left to get a professional license")
elif(age<30):
print("You can get an international & national driving license")
print("and",30-age ,"years left to get a professional license")
elif(age>=30):
print("You can get a national, international and professional driving license")
Enter your age: 20
You can get a national driving license
, 1 years left to get an international license
and 10 years left to get a professional license


Syntax (Nested If)
Adding if (if...elif...else) statements inside of if (if...elif...else) statements in computer programming is named nesting. Any number of these statements can be nested inside one another. Indentation is the only way to figure out the level of nesting. They can get confusing, so they must be avoided unless necessary.
1. num = 15
2. if num >= 0:
3. if num == 0:
4. print("Zero")
5. else:
6. print("Positive number")
7. else:
8. print("Negative number")


EXERCISE THIRTEEN
Use nested If-Else statements and Elifs to Create a program to print the largest number of three different numbers entered by the user.

```
num1=int(input("Input the first number: "))
num2=int(input("Input the second number: "))
num3=int(input("Input the third number: "))
if(num1>num2):
    if(num1>num3):
        print(num1,"is the greatest number")
    else:
        print(num3,"is the greatest number")
elif(num2>num1):
    if(num2>num3):
        print(num2,"is the greatest number")
    else:
        print(num3,"is the greatest number")
elif(num3>num1):
    if(num3>num2):
        print(num3,"is the greatest number")
    else:
        print(num2,"is the greatest number")
```

main.py

Run

```
num1=int(input("Input the first number: "))
num2=int(input("Input the second number: "))
num3=int(input("Input the third number: "))
if(num1>num2):
    if(num1>num3):
        print(num1,"is the greatest number")
    else:
        print(num3,"is the greatest number")
elif(num2>num1):
    if(num2>num3):
        print(num2,"is the greatest number")
    else:
        print(num3,"is the greatest number")
elif(num3>num1):
    if(num3>num2):
        print(num3,"is the greatest number")
    else:
        print(num2,"is the greatest number")
```

Shell

```
Input the first number: 3
Input the second number: 50
Input the third number: 18
50 is the greatest number
>
```

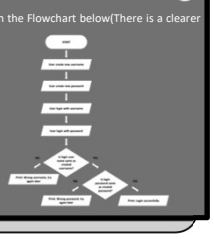
EXERCISE FOURTEEN

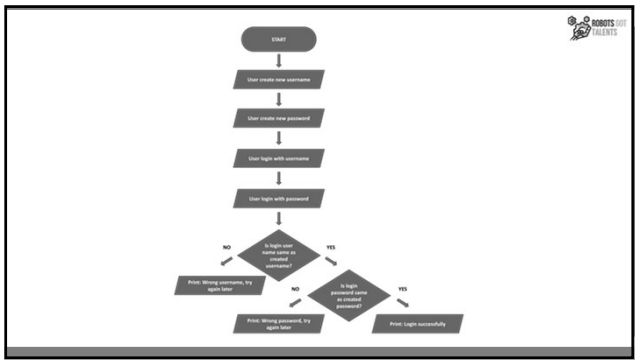
Use nested If-else statements to create a Login System as shown in the Flowchart below(There is a clearer image in the next page):

Shell

Clear

```
CREATE ACCOUNT:
Create username:
```






```
1. print("CREATE ACCOUNT:")
2. cname=input("Create username: ")
3. cpass=input("Create password: ")
4. print("ACCOUNT CREATED\n####################")
5. print("\nLOGIN: ")
6. uname=input("Enter your username: ")
7. upass=input("Enter your password: ")
8. if(uname==cname):
9.     if(upass==cpass):
10.         print("Login successfully")
11.     else:
12.         print("Wrong password, try again later")
13. else:
14.     print("Wrong username, try again later")
```

main.py

Run

```
1  print("CREATE ACCOUNT:")
2  cname=input("Create username: ")
3  cpass=input("Create password: ")
4  print("ACCOUNT CREATED\n####################")
5  print("\nLOGIN: ")
6  uname=input("Enter your username: ")
7  upass=input("Enter your password: ")
8  if(uname==cname):
9      if(upass==cpass):
10         print("Login successfully")
11     else:
12         print("Wrong password, try again later")
13 else:
14     print("Wrong username, try again later")
```

Shell

```
CREATE ACCOUNT:
Create username: John202
Create password: 123456
ACCOUNT CREATED
##################

LOGIN:
Enter your username: John202
Enter your password: 123456
Login successfully
>
```

Combined conditional statements

```
#The (and) keyword is a logical operator, and is used to combine conditional statements
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")

#The (or) keyword is a logical operator, and is used to combine conditional statements
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

Example 1

main.py

```
print("Create Account: ")
uname=input("Create new username: ")
cpass=int(input("Create new password: "))
print("\nAccount Created\n")
name=input("Enter username:  ")
ipass=int(input("Create password: "))
if(name==uname and ipass==cpass):
    print("Welcome (:")
else:
    print("Username or password is wrong")
```

Shell

```
Create Account:
Create new username: Youssef
Create new password: 20222023
Account Created

Enter username: Youssef
Create password: 20222023
Welcome (:
```

main.py

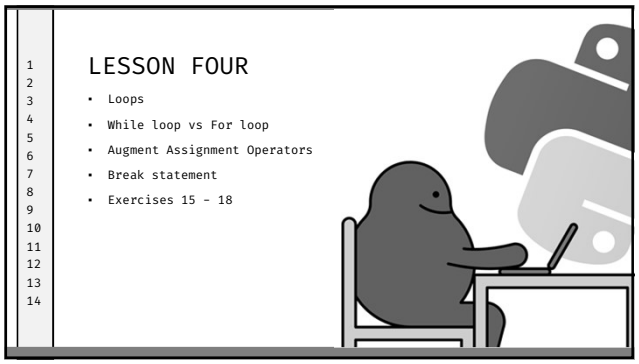
```
print("Create Account: ")
uname=input("Create new username: ")
cpass=int(input("Create new password: "))
print("\nAccount Created\n")
name=input("Enter username:  ")
ipass=int(input("Create password: "))
if(name==uname and ipass==cpass):
    print("Welcome (:")
else:
    print("Username or password is wrong")
```

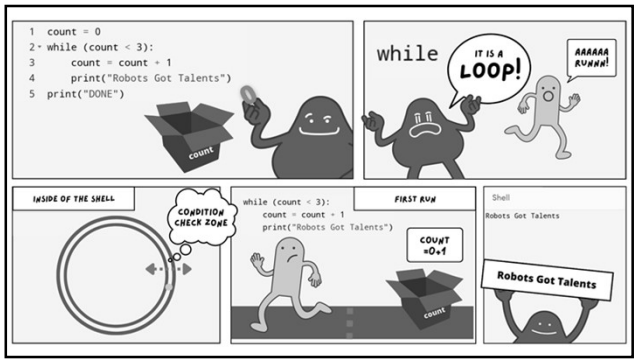
Shell

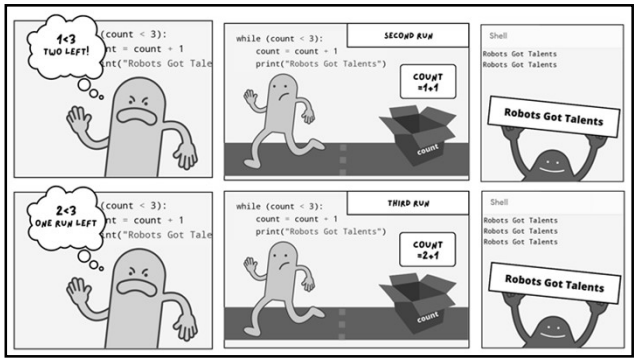
```
Create Account:
Create new username: Youssef
Create new password: 20222023
Account Created

Enter username: Yousef
Create password: 202213
Username or password is wrong
```

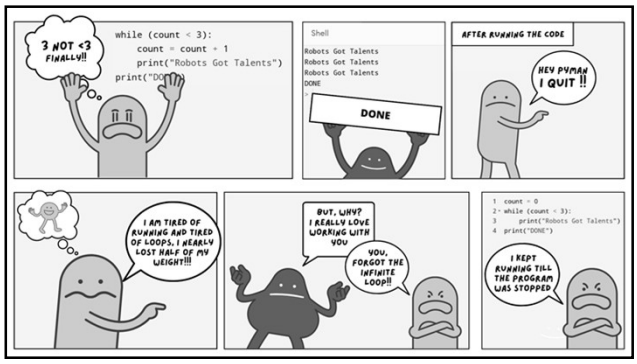
Python Fundamentals

QUIZ THREE


LESSON FOUR
• Loops
• While loop vs For loop
• Augment Assignment Operators
• Break statement
• Exercises 15 – 18


count = 0
while (count < 3):
count = count + 1
print("Robots Got Talents")
print("DONE")
count
while
IT IS A LOOP!
AAAAAA RUNNN!
INSIDE OF THE SHELL
CONDITION CHECK ZONE
FIRST RUN
while (count < 3):
count = count + 1
print("Robots Got Talents")
COUNT =0+1
count
Shell
Robots Got Talents
Robots Got Talents


1<3 TWO LEFT!
SECOND RUN
while (count < 3):
count = count + 1
print("Robots Got Talents")
COUNT =1+1
count
Shell
Robots Got Talents
Robots Got Talents
Robots Got Talents
2<3 ONE RUN LEFT
THIRD RUN
while (count < 3):
count = count + 1
print("Robots Got Talents")
COUNT =2+1
count
Shell
Robots Got Talents
Robots Got Talents
Robots Got Talents
Robots Got Talents



LOOPS (For & While)

The for and while loops in Python are used to iterate over a sequence. The statements inside of the loop would repeat until the loop condition is no longer satisfied. Python relies on indentation (whitespace at the beginning of a line) to define scope in the code.

For Loop

```
for count in range (0,3):
    count = count + 1
    print("Robots Got Talents")
#Robots Got Talents prints 3 times
```

While Loop

```
count = 0
while (count < 3):
    count = count + 1
    print("Robots Got Talents")
#Robots Got Talents prints 3 times
```

Syntax (While)

```
count = 0
while(count<=10):
    print(count)
    count = count+1
#counts from 0 to 10
```

```
r = 1
while(r<5):
    print(":")
    r = r+1
#prints (: 4 times
```

Notes:

- Use logical conditions for while conditions [>=, <=,>,<,==,!=)
- If the while block consists of a single statement then we can declare the entire loop in a single line
- The while loop executes the block until a condition is satisfied. When the condition becomes false, the statement immediately after the loop is executed.
- An else statement can run a block of code once when the condition of the while loop no longer is true.

Example 1

main.py

```
count = 0
while(count<10):
  print(count*2)
  count = count+1
```

Shell

```
0
2
4
6
8
10
12
14
16
18
```

Example 2

main.py

```
rep = 0
while(rep<3):
  print(":")
  rep=rep+1
```

Shell

```
:
:
:
```

Example 3

main.py

```
count=4
while(count>=0):
    print(count)
    count=count-1
```

Shell

```
4
3
2
1
0
```

Example 4

main.py

```
count=1
s = 0
while(count<=4):
    num=int(input("Input num: "))
    s=s+num
    count=count+1
print("sum=",s)
```

Shell

```
Input num: 10
Input num: 2
Input num: 1
Input num: 3
sum= 16
```

Useful Augmented Assignment Operators

```
x+=5  # x=x+5
x-=5  # x=x-5
x*=5  # x=x*5
x/=5  # x=x/5
x%=5  # x=x%5
```

Example 5

main.py

```
count=1
s = 0
while(count<=4):
    num=int(input("Input num: "))
    s=s+num
    count+=1
print("sum=",s)
```

Shell

```
Input num: 10
Input num: 2
Input num: 2
Input num: 3
sum= 17
```

EXERCISE FIFTEEN

In no more than 5 lines of code create a program to count from one to the number inputted by the user:

Notes:

- Use while Loop

EXERCISE FIFTEEN SOLUTION

```
1. num=int(input("Enter the number: "))
2. count=0
3. while (count<num):
4.     count=count+1
5.     print(count)
```

main.py | Run | Shell

```
num=int(input("Enter the number: "))
count=0
while (count<num):
    count=count+1
    print(count)
```

```
Enter the number: 50
1
2
3
4
```

EXERCISE SIXTEEN

Create an account signup and login system, where the user can type in the username or password incorrect for no more than 3 times.

Notes:

- The user should know how many trails left

Shell

```
CREATE ACCOUNT:
Create username:
```

```
print("CREATE ACCOUNT:")
cname=input("Create username: ")
cpass=input("Create password: ")
print("ACCOUNT CREATED\n####################")
print("\nLOGIN: ")
t=3
while(t>=0):
    uname=input("Enter your username: ")
    upass=input("Enter your password: ")
   if(uname==cname and upass==cpass):
                print("Login successfully")
        t=-1
    else:
        print("Wrong username or password, try again")
        print(t,"trials left")
        t=t-1
```

```
main.py                                                    Run        Shell
print("CREATE ACCOUNT:")
cname=input("Create username: ")
cpass=input("Create password: ")
print("ACCOUNT CREATED\n####################")
print("\nLOGIN: ")
t=3
while(t>=0):
    uname=input("Enter your username: ")
    upass=input("Enter your password: ")
    if(uname==cname and upass==cpass):
        print("Login successfully")
        t=-1
    else:
        print("Wrong username or password, try again")
        print(t,"trials left")
        t=t-1

CREATE ACCOUNT:
Create username: John12
Create password: 1234
ACCOUNT CREATED
###################

LOGIN:
Enter your username: JIg12
Enter your password: 1234
Wrong username or password, try again
3 trials left
Enter your username: John12
Enter your password: 12345
Wrong username or password, try again
2 trials left
Enter your username: John12
Enter your password: 1234
Login successfully
```

Syntax (Break statement)

- The (break) statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop. If the break statement is inside a nested loop (loop inside another loop), the break statement will terminate the innermost loop.

```
n = 5
while n > 0:
    n -= 1
    if n == 2:
        break
    print(n)
print('Loop ended.')
```

Example 1

main.py

```
trail=0
print("CREATE ACCOUNT:")
cname= input("Create new username: ")
cpass= int(input("Create new password: "))
while (trail<=3):
    print("\nLOGIN: ")
    iname= input("Enter username: ")
    ipass= int(input("Enter password: "))
    if (iname==cname and ipass==cpass):
        print("Welecome (:\n")
        print("Login Successful")
        break
    else:
        print("\nwrong username or password")
        print("Please Try again ",3-trail, "trail/s left")
        trail=trail+1
```

Run

Shell

```
CREATE ACCOUNT:
Create new username: Angela
Create new password: 10203040
LOGIN:
Enter username: Angela
Enter password: 20130
wrong username or password
Please Try again  3 trail/s left

LOGIN:
Enter username: Angla
Enter password: 10204030
wrong username or password
Please Try again  2 trail/s left

LOGIN:
Enter username: Angla
Enter password: 10203040
wrong username or password
Please Try again  1 trail/s left

LOGIN:
Enter username: Angela
Enter password: 10203040
Welecome (:

Login Successful
>
```

Example 1

main.py

```
trail=0
print("CREATE ACCOUNT:")
cname= input("Create new username: ")
cpass= int(input("Create new password: "))
while (trail<=3):
    print("\nLOGIN: ")
    iname= input("Enter username: ")
    ipass= int(input("Enter password: "))
    if (iname==cname and ipass==cpass):
        print("Welecome (:\n")
        print("Login Successful")
        break
    else:
        print("\nwrong username or password")
        print("Please Try again ",3-trail, "trail/s left")
        trail=trail+1
```

Run

Shell

```
CREATE ACCOUNT:
Create new username: Angela
Create new password: 10203040
LOGIN:
Enter username: Angela
!Enter password: 21203
wrong username or password
Please Try again  3 trail/s left

LOGIN:
Enter username: Angela
Enter password: 10203040
Welecome (:

Login Successful
>
```

Example 1

main.py

```
trail=0
print("CREATE ACCOUNT:")
cname= input("Create new username: ")
cpass= int(input("Create new password: "))
while (trail<=3):
    print("\nLOGIN: ")
    iname= input("Enter username: ")
    ipass= int(input("Enter password: "))
    if (iname==cname and ipass==cpass):
        print("Welecome (:\n")
        print("Login Successful")
        break
    else:
        print("\nwrong username or password")
        print("Please Try again ",3-trail, "trail/s left")
        trail=trail+1
```

Run

Shell

```
CREATE ACCOUNT:
Create new username: Angela
Create new password: 10203045
LOGIN:
Enter username: Angela
Enter password: 10203045
Welecome (:

Login Successful
>
```

EXERCISE_SEVENTEEN

Fix all the errors in the program below, it should only print numbers divisible by 3 lower than the number inputted by the user.

main.py | Run | Shell

```
num=int(input("Input a number: "))
count=0
while(num>=count):
    if(num%3==0):
        print(num)
    count=count+1
```

```
Input a number: 90
90
90
90
90
90
90
```

```
1. num=int(input("Input a number: "))
2. while(num>=0):
3.     if(num%3==0):
4.         print(num)
5.     num=num-1
```

main.py | Run | Shell

```
num=int(input("Input a number: "))
while(num>=0):
    if(num%3==0):
        print(num)
    num=num-1
```

```
Input a number: 30
30
27
24
21
18
```

EXERCISE_EIGHTEEN

In no more than 5 lines of code, create a program to print all even numbers lower than the inputted number by the user.

Notes:

- Use % modules

```
num=int(input("Input a number: "))
while(num>=0):
    if(num%2==0):
        print(num)
    num=num-1
```

main.py | Run | Shell

```
num=int(input("Input a number: "))
while(num>=0):
    if(num%2==0):
        print(num)
    num=num-1
```

```
Input a number: 8
8
6
4
2
0
>
```

Syntax (For) – numbers & strings

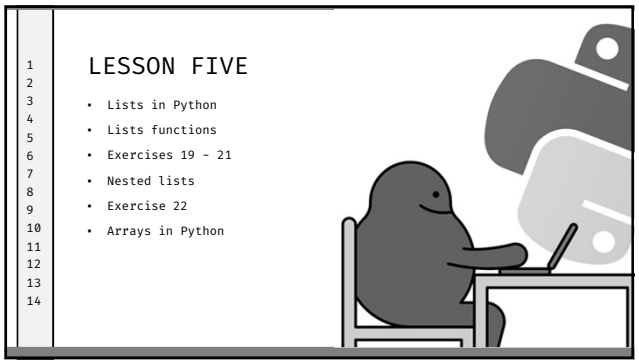
```
for x in range(1, 4):
    print(x)
#counts from 1 to 3
```

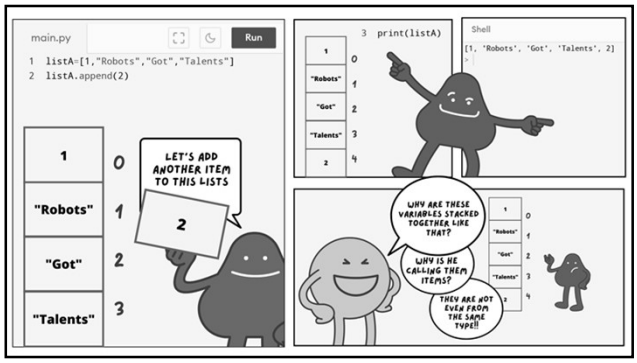
```
for x in range(1, 5):
    print(":")
#prints (: 4 times
```

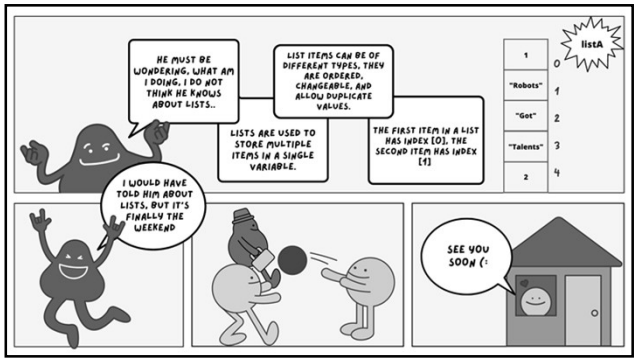
```
for x in range(0, 31, 5):
    print(x)
#prints numbers from 0 to 30 adding 5 (0,5,10,15,20..)
```

Python Fundamentals

QUIZ FOUR


LESSON FIVE
• Lists in Python
• Lists functions
• Exercises 19 – 21
• Nested lists
• Exercise 22
• Arrays in Python


main.py
Run
listA=[1,"Robots","Got","Talents"]
listA.append(2)
1
"Robots"
"Got"
"Talents"
0
1
2
3
LET'S ADD ANOTHER ITEM TO THIS LISTS
2
print(listA)
Shell
[1, 'Robots', 'Got', 'Talents', 2]
WHY ARE THESE VARIABLES STACKED TOGETHER LIKE THAT?
WHY IS HE CALLING THEM ITEMS?
THEY ARE NOT EVEN FROM THE SAME TYPE!!


HE MUST BE WONDERING, WHAT AM I DOING, I DO NOT THINK HE KNOWS ABOUT LISTS...
LIST ITEMS CAN BE OF DIFFERENT TYPES. THEY ARE ORDERED, CHANGEABLE, AND ALLOW DUPLICATE VALUES.
LISTS ARE USED TO STORE MULTIPLE ITEMS IN A SINGLE VARIABLE.
THE FIRST ITEM IN A LIST HAS INDEX [0], THE SECOND ITEM HAS INDEX [1]
listA
1
"Robots"
"Got"
"Talents"
2
0
1
2
3
4
I WOULD HAVE TOLD HIM ABOUT LISTS, BUT IT'S FINALLY THE WEEKEND
SEE YOU SOON (:

Lists

List are ordered sequence of items (Variables), starting from 0 to n (number of items) and separated by a comma. A list could include items of different types including strings, integers, Boolean and lists. Like variables there are functions for lists capable of changing, determining the order and contents of items in the list.

- Lists are used to store multiple items in a single variable.
- List items are ordered, changeable, and allow duplicate values.
- List items are indexed, the first item has index [0], the second item has index [1]
- There are four types of arrays in Python: Lists, Tuples, Sets, and Maps

```
#Lists

listA = ["Python", "C", "Java"]
listB = [1, 3, 7, 9, 5]
listC = [True, False, True]
listD = [1,"RGT",True,5,"Python"]
print(listA)
print(len(listB))
print(listC[2])
```

```
#Lists

list1 = [True , 2 , 3 , "Python" , "C" , "Java"]
list2= ["Robots" , "Got" , "Talents" , "Python" , "Curriculum"]
print(list1)        #Outputs: [True, 2, 3, 'Python', 'C', 'Java']
print(len(list2))   #Outputs: 4
print(list2[3])     #Outputs: 'Python'
list1[4] = "C+"
#Change item in index 4 to "C+"
List3=[list1, list2] #Nested List
List3[0][1] = 1 #Change item 1 in list1
#len() is an example of list functions
```

List1	
True	0
2	1
3	2
"Python"	3
"C"-"C+"	4
"Java"	5

List2	
"Robots"	0
"Got"	1
"Talents"	2
"Python"	3
"Curriculum"	4

Syntax (Lists)

```
list1 = [True , 2 , 3 , "Python" , "C" , "Java"] #Creating lists
list2= ["Robots" , "Got" , "Talents" , "Python" , "Curriculum"]
```

Changing Items

```
odd = [2, 4, 6, 8]
odd[0] = 1    # change the 1st item
odd[1:4] = [3, 5, 7] # change 2nd to 4th items
```

Lists with print() Function

```
Lits1 = [1 , 2 , True , "RGT" , "Python"]
print(list1) #Print all items in list
print(list1[3]) #Prints item in index 2 ('RGT')
```

Example 1

main.py

```python
Rlist = ['R', 'G', 'T', 2022 ]
# first item
print(Rlist[0])
print(Rlist[3])
Nlist = ["PYTHON", [2, 0, 2, 3]]
# Nested indexing
print(Nlist[0][1])
print(Nlist[1][2])
```

Shell

```
R
2022
Y
2
>
```

Example 2

main.py

```python
even = [1, 3, 4, 6, 8]
even[0] = 2    # change the 1st item
even[1:4] = [4, 6, 8] # change 2nd to 4th items
even[4] = 10
print(even)
```

Shell

```
[2, 4, 6, 8, 10]
>
```

```
Useful lists functions
append()  #Adds an element at the end of the list
clear()   #Removes all the elements from the list
count()   #Returns the number of elements with the specified value
extend()  #Add the elements of a list, to the end of the current list
index()   #Returns the index of the first element with the specified value
insert()  #Adds an element at the specified position
pop()     #Removes the element at the specified position
remove()  #Removes the first item with the specified value
reverse() #Reverses the order of the list
sort()    #Sorts the list
```

Example 3

main.py

```python
# Appending and Extending lists in Python
odd = [1, 3, 5]
odd.append(7) #Add 7 to the existing list index 3
print(odd)
odd.extend([9, 11, 13]) #Add 9,11,13 to the existing list
print(odd)
```

Shell

```
[1, 3, 5, 7]
[1, 3, 5, 7, 9, 11, 13]
>
```

Example 4

main.py

```python
# Deleting list items
my_list = ['p', 'r', 'o', 'b', 'l', 'e', 'm']
# delete one item
del my_list[2]
print(my_list)
# delete multiple items
del my_list[1:5]
print(my_list)
# delete the entire list
del my_list
```

Shell

```
['p', 'r', 'b', 'l', 'e', 'm']
['p', 'm']
>
```

EXERCISE_NINETEEN

In no more than 9 lines of code. Create a list with the letters of the word "python", Print the full characters in the list and the length of the list, then the each character individually in a single line.

Notes:

- Use while loop

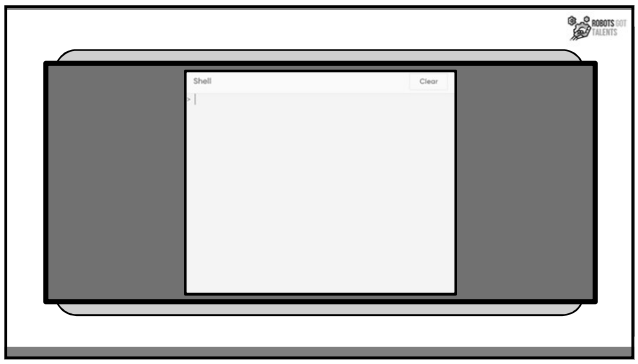
```
PList=["p","y","t","h","o","n"]
print(PList)
L=len(PList)
print(L)
n=0
while(n<L):
    print(PList[n])
    n=n+1
```

EXERCISE_TWENTY

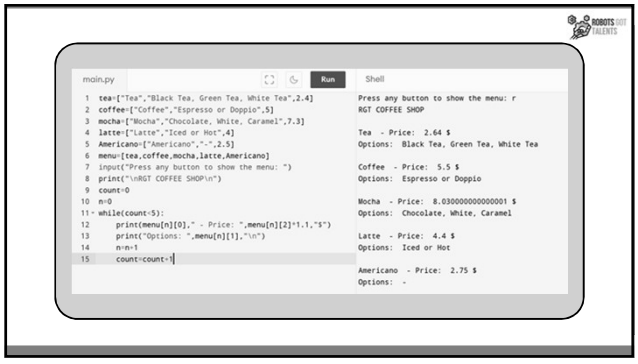
Using nested lists create a coffee shop menu with at least 5 products, each product should have its own name, options, and price. The price of all products should be multiplied by 1.1 (service & dine-in fees). The menu should print when the user input any button. Build the menu, so each product is easy to modify.

Notes:

- Use while loop
- An example is included in the next page



```
tea=["Tea","Black Tea, Green Tea, White Tea",2.4]
coffee=["Coffee","Espresso or Doppio",5]
mocha=["Mocha","Chocolate, White, Caramel",7.3]
latte=["Latte","Iced or Hot",4]
Americano=["Americano","-",2.5]
menu=[tea,coffee,mocha,latte,Americano]
input("Press any button to show the menu: ")
print("\nRGT COFFEE SHOP\n")
count=0
n=0
while(count<5):
    print(menu[n][0]," - Price: ",menu[n][2]*1.1,"$")
    print("Options: ",menu[n][1],"\n")
    n=n+1
    count=count+1
```



Example 5

main.py

```
#user input items of list
# creating an empty list
lst = []
# number of elements as input
n = int(input("Enter number of elements : "))
# iterating till the range
i=0
while i <n:
    ele = int(input("Enter element: "))
    lst.append(ele)
    i=i+1
print(lst)
```

Shell

```
Enter number of elements : 4
Enter element: 6
Enter element: 3
Enter element: 9
Enter element: 12
[6, 3, 9, 12]
```

Example 6

main.py

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

Shell

```
3
```

EXERCISE TWENTY-ONE

Create an empty list that only accepts even numbers inputted by the user. No more items can be included in the list after the user enter the number 0 (The list should not include the number 0).

Notes:

- Use the append and remove functions

```
Elist=[]
i=1
while(i!=0):
    i=int(input("Enter a number: "))
    if(i%2==0):
        Elist.append(i)
Elist.remove(0)
print(Elist)
```

main.py

```
Elist=[]
i=1
while(i!=0):
    i=int(input("Enter a number: "))
    if(i%2==0):
        Elist.append(i)
Elist.remove(0)
print(Elist)
```

Shell

```
Enter a number: 9
Enter a number: 2
Enter a number: 3
Enter a number: 5
Enter a number: 6
Enter a number: 8
Enter a number: 10
Enter a number: 0
[2, 6, 8, 10]
```

Nested Lists

A list in Python can include multiple lists, these are called nested lists. To create a nested list in Python simply create a list then put one or more lists (as elements) in that list.

- Nested Lists can work with lists functions
- Each element in a nested list can be accessed, changed, and removed

```
Studentsdata = [["John",19,1,3.2], ["Sophia",20,2,3.8], ["Jared",21,3,2.8]]
''' The list (Studentsdata) contains 3 elements which are lists each nested list
includes 4 elements (name, age, year, GPA) '''
```

Accessing Elements in nested lists:

```
Studentsdata[1][3] = 3.6 # Sophia's GPA changed to 3.6
#List: [['John', 19, 1, 3.2], ['Sophia', 20, 2, 3.6], ['Jared', 21, 3, 2.8]]
```

EXERCISE TWENTY-TWO

Create the table below:

Then edit the records for apples, bananas, and orange individually to be as shown below.

Apple Price = 2.2,Banana Available = True
Orange price = 1.4

- Print the list after the modifications

Notes:

- Use nested lists

Product	ID	Price	Available
Apple	1243	2	True
Banana	1245	1	False
Orange	1244	1.5	True
Kiwi	1234	2	True
Pineapple	1230	4	True
Mango	1256	3	True

```
FProducts = [["Apple",1243,2,True],["Banana",1245,1,False],
["Orange",1244,1.5,True],["Kiwi",1234,2,True],["Pineapple",
1230,4,True],["Mango",1256,3,True]]
FProducts[0][2]=2.2
FProducts[1][-1]=True
FProducts[2][2]=1.4
print(FProducts)
```

List	Tuple	Set	Dictionary
Lists are represented by [ ]	Tuples are represented by ( )	Sets are represented by { }	Dictionaries are represented by { }
List allows duplicate elements	Tuple allows duplicate elements	Set will not allow duplicate elements	Set will not allow duplicate elements and dictionary doesn't allow duplicate keys.
List can use nested among all	Tuple can use nested among all	Set can use nested among all	Dictionary can use nested among all
List is mutable i.e we can make any changes in list.	Tuple is immutable i.e we can not make any changes in tuple	Set is mutable i.e we can make any changes in set. But elements are not duplicated.	Dictionary is mutable. But Keys are not duplicated.
List is ordered	Tuple is ordered	Set is unordered	Dictionary is ordered (Python 3.7 and above)

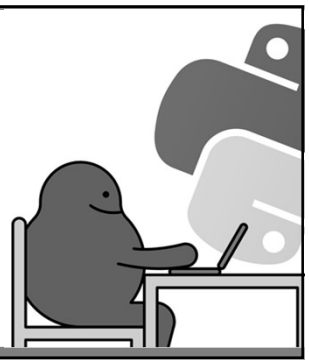
List	Tuple	Set	Dictionary
List can be created using **list()** function	Tuple can be created using **tuple()** function.	Set can be created using **set()** function	Dictionary can be created using **dict()** function.
List is mutable i.e we can make any changes in list.	Tuple is immutable i.e we can not make any changes in tuple	Set is mutable i.e we can make any changes in set. But elements are not duplicated.	Dictionary is mutable. But Keys are not duplicated.
List is ordered	Tuple is ordered	Set is unordered	Dictionary is ordered (Python 3.7 and above)
Creating an empty list l=[]	Creating an empty Tuple t=()	Creating a set a=set() b=set(a)	Creating an empty dictionary d={}
Example: [1, 2, 3, 4, 5]	Example: (1, 2, 3, 4, 5)	Example: {1, 2, 3, 4, 5}	Example: {1, 2, 3, 4, 5}

Python Fundamentals

QUIZ FIVE

LESSON SIX

- Creating functions
- Exercises 23 – 24
- Libraries and Modules
- Installing Libraries
- Understanding Libraries



Creating Function

A function is a block of organized, reusable code that is used to perform a single, related action, when it is called. A function is defined using the def keyword and called by typing the function name followed by parenthesis (). Functions

```
#Creating Functions

#Defining a function to print RGT every time its called
def RGT():
    print("Robots Got Talents")

# Calling the function
RGT()
print("Hello There")
```

Syntax (Functions)

```
#Defining functions with no parameters
def my_function():
  print("Hello World")
  #The rest of the function should be indented after the (:)
```

```
#Defining functions with parameters
def addf (num1: int, num2: int) -> int:
    #def function_name(parameter: data_type) -> return_type:
    num3 = num1 + num2
    return num3
```

```
"""Calling function with no parameters"""
my_function()
```

```
#Calling function with parameters
num1, num2 = 5, 15
ans = addf(num1, num2)
print("Answer = ", ans)
```

Example 1

main.py

```
def addf (num1: int, num2: int) -> int:
    #def function_name(parameter: data_type) -> return_type:
    num3 = num1 + num2
    return num3
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the first number: "))
ans = addf(num1, num2)
print("Answer = ", ans)
```

Shell

```
Enter the first number: 10
Enter the first number: 35
Answer =  45
>
```

Example 2

main.py

```
def evenOdd(x):
    if (x % 2 == 0):
        print("even")
    else:
        print("odd")

evenOdd(2)
evenOdd(3)
```

Shell

```
even
odd
>
```

EXERCISE TWENTY-THREE

Create a function that multiply two numbers together in the program, ask the user to input two numbers, then multiply 5 and 4.

Notes:

- Create a function with parameters

```
1. def mul(a, b):
2.       return a * b
3.
4. n1=int(input("Input number 1: "))
5. n2=int(input("Input number 1: "))
6. total=mul(n1, n2)
7. print(total)
8. print(mul(5, 4))
```

main.py

```
def mul(a, b):
    return a * b

n1=int(input("Input number 1: "))
n2=int(input("Input number 1: "))
total=mul(n1, n2)
print(total)
print(mul(5, 4))
```

Shell

```
Input number 1: 2
Input number 1: 7
14
20
>
```

EXERCISE TWENTY-FOUR

Create a function that great the user by the mentioned name in the program.

Notes:

- Create a function with parameters

```
def greetmebyname(name):
    return "Hello "+name

p=greetmebyname("Youssef")
print(p)
```

main.py | Run | Shell

```
def greetmebyname(name):
    return "Hello "+name

p=greetmebyname("Youssef")
print(p)
```

```
Hello Youssef
>
```

Python Libraries & Modules

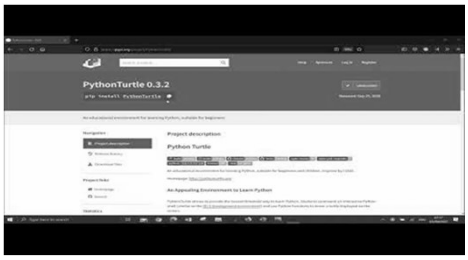
A Python library is a collection of related modules. It contains bundles of code/functions that can be used repeatedly in different programs. As of 2022 there are over 137,000 Python Libraries, there are libraries for most of the things you can think of; drawing objects, creating Databases, sending messages, text to speech and even creating advanced mathematical calculations, data visualization and machine learning.

- To use libraries, you will need to use a Python desktop IDE, In this lesson we will use the official Python 3.9 IDLE (Download). You can use any other Integrated Developed Environment as Pycharm, or Visual Studio.

Installing & Using Python Library:

1. After installing the Python IDLE/Python 3.9, open (cmd) command prompt
2. Type in (pip install then the library name)
3. Open the Python IDLE, create a new file then type in (import then the library name)

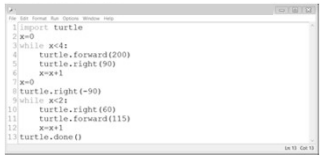
Installing Python Libraries



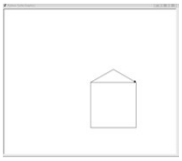
Understanding Libraries Functions

Most Python libraries would have a Documentation website or details about the library and its functions the pypi.org website. If a library does not have enough resources, you can look for examples or projects built using this library.

- The library we will be using in this lesson in named turtle. Which is Python library that enables users to create pictures and shapes by providing them with a virtual canvas.



```
import turtle
x=0
while x<4:
    turtle.forward(200)
    turtle.right(90)
    x=x+1
x=0
turtle.right(-90)
while x<2:
    turtle.right(60)
    turtle.forward(115)
    x=x+1
turtle.done()
```



Python Turtle Documentation Sample:

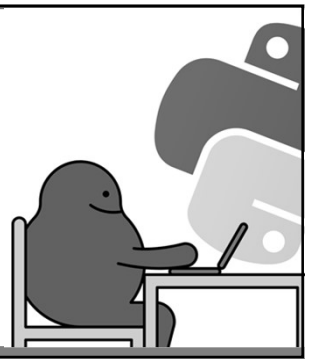
Turtle()	()- None	Creates and returns a new turtle object
forward()	()- amount	Moves the turtle forward by the specified amount
backward()	()-amount	Moves the turtle backward by the specified amount
right()	()-angle	Turns the turtle clockwise
left()	()-angle	Turns the turtle counterclockwise
penup()	()-None	Picks up the turtle's Pen
pendown()	()-None	Puts down the turtle's Pen
up()	()-None	Picks up the turtle's Pen
down()	()-None	Puts down the turtle's Pen
color()	()-Color name	Changes the color of the turtle's pen
fillcolor()	()-Color name	Changes the color of the turtle will use to fill a polygon

Python Fundamentals

QUIZ SIX

LESSON SEVEN

- Python Turtle documentation
- Exercises 25 – 27
- Introducing bonus projects



Python Turtle Documentation Sample:

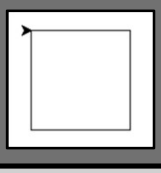
Turtle()	()- None	Creates and returns a new turtle object
forward()	()- amount	Moves the turtle forward by the specified amount
backward()	()-amount	Moves the turtle backward by the specified amount
right()	()-angle	Turns the turtle clockwise
left()	()-angle	Turns the turtle counterclockwise
penup()	()-None	Picks up the turtle's Pen
pendown()	()-None	Puts down the turtle's Pen
up()	()-None	Picks up the turtle's Pen
down()	()-None	Puts down the turtle's Pen
color()	()-Color name	Changes the color of the turtle's pen
fillcolor()	()-Color name	Changes the color of the turtle will use to fill a polygon

Example 1

```
import turtle
tur=turtle.Turtle()
tur.penup()
tur.setpos(-20,40)
tur.pendown()
tur.pensize(10)
tur.pencolor("cyan")
tur.forward(100)
tur.backward(100)
tur.right(90)
tur.forward(100)
tur.left(90)
tur.forward(100)
tur.backward(100)
tur.right(90)
tur.forward(100)
tur.left(90)
tur.forward(100)
turtle.done()
```

EXERCISE_TWENTY-FIVE

Draw this shape using Python Turtle in no more than 7 lines:

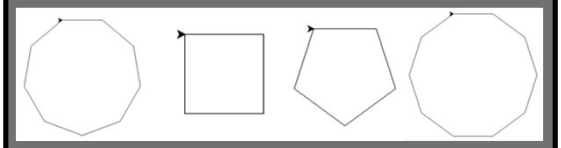


```
1. import turtle
2. x=0
3. while x<4:
4.     turtle.forward(100)
5.     turtle.right(90)
6.     x=x+1
7. turtle.done()
```

```
1.  import turtle
2.  x=0
3.  while x<4:
4.      turtle.forward(100)
5.      turtle.right(90)
6.      x=x+1
7.  turtle.done()
```

EXERCISE TWENTY-SIX

Create a program to draw a shape with the inputted number of angles.

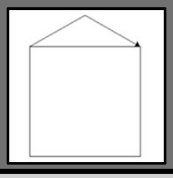


```
import turtle
i=int(input("Input number of angles: "))
angle=360/i
x=0
while x<i:
    turtle.forward(100)
    turtle.right(angle)
    x=x+1
turtle.done()
```

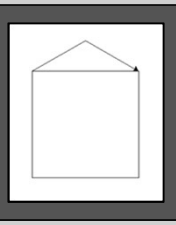
```
import turtle
i=int(input("Input number of angles: "))
angle=360/i
x=0
while x<i:
    turtle.forward(100)
    turtle.right(angle)
    x=x+1
turtle.done()
```

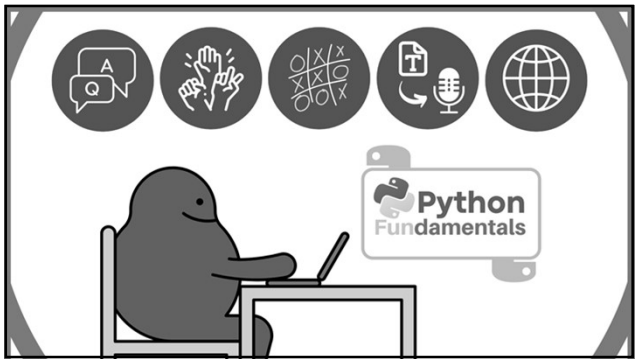
EXERCISE TWENTY-SEVEN

Draw this shape using Python Turtle library:



```
import turtle
x=0
while x<4:
    turtle.forward(200)
    turtle.right(90)
    x=x+1
x=0
turtle.right(-90)
while x<2:
    turtle.right(60)
    turtle.forward(115)
    x=x+1
turtle.done()
```





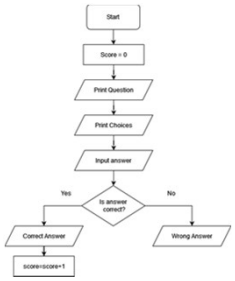
PROJECT 1 – QUIZ

Welcome to your first project the Quiz, throughout this project we will move through most of the covered lessons in the course. At anytime you can use the topic selector to revise any previous lesson.

STEP ONE: Now let's think of how automated quizzes wo

- A question appears on the screen
- The user type in or choose an answer
- If the answer is correct points are added to the score
- After finishing all questions grades, and/or results appears on the screen

Now try drawing a simple flowchart for this project.



STEP TWO: Use the flowchart drawn to create a single question quiz.

- The first thing to do is to create the variable score and give it the value 0
- The second thing is printing the question and the choices
- Then ask the user for input (answer)
- Check if the answer is correct
- If the answer is correct increase the core by 1

STEP THREE: Creating the code

```
score=0
#score variable created and given value 0
print("What is Python?")
print("(a)-programming language, (b)-scripting language, (c)-human language")
#Question and answer printed
answer=input("Answer: ")
#Answer chosen by the user
if(answer=='a'):
#The correct answer is 'a'.
    print("correct answer")
    score=score+1
else:
    print("wrong answer")
```

STEP FOUR: Adding more questions

```
score=0
#Q1
print("What is Python?")
print("(a)-programming language, (b)-scripting language, (c)-human language")
answer=input("Answer: ")
if(answer=='a'):
    print("correct answer\n")
    score=score+1
else:
    print("wrong answer")
Q2
print("When was Python developed?")
print("(a)-1980, (b)-1991, (c)-1982")
answer=input("Answer: ")
if(answer=='b'):
    print("correct answer\n")
    score=score+1
else:
    print("wrong answer")
```

STEP FIVE: Creating a variable for correct answer, to show when user choose the wrong one

```
score=0
print("What is Python?")
print("(a)-programming language, (b)-scripting language, (c)-human language")
answer=input("Answer: ")
c_ans='a'  #new variable
if(answer==c_ans):
    print("correct answer")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n") #new line
print("When was Python developed?")
print("(a)-1980, (b)-1991, (c)-1982")
answer=input("Answer: ")
c_ans='b'
if(answer== c_ans):
    print("correct answer\n")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n")
```

STEP SIX: add more questions & print the final score

```
#Question logic
print("When was Python developed?") #question
print("(a)-1980, (b)-1991, (c)-1982") #choices
answer=input("Answer: ") #ask for answer
c_ans='b' #correct answer
if(answer== c_ans): #questition & answer logic
    print("correct answer\n")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n")
```

```
#Print score
print("Score = ",score)
```

STEP SEVEN: Finalizing the code & improving the user experience

```
score=0
name=input("Enter your name: ")
print("\nQUIZ\n")
print("What is Python?")
print("(a)-programming language, (b)-scripting language, (c)-human
  language")
answer=input("Answer: ")
c_ans='a'
if(answer==c_ans):
    print("correct answer")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n")
print("When was Python developed?")
print("(a)-1980, (b)-1991, (c)-1982")
answer=input("Answer: ")
c_ans='b'
if(answer== c_ans):
    print("correct answer\n")
    score=score+1
```

```
21.else:
22.     print("wrong answer")
23.     print("The correct answer is ",c_ans,"\n")
24.print("A procedure used for solving a problem or performing a
   computation:")
25.print("(a)-programming language, (b)-programming, (c)-algorithm")
26.answer=input("Answer: ")
27.c_ans='c'
28.if(answer==c_ans):
29.     print("correct answer")
30.     score=score+1
31.else:
32.     print("wrong answer")
33.     print("The correct answer is ",c_ans,"\n")
34.print("Which of the following is not a High Level Programming Language")
35.print("(a)-Java, (b)-Python, (c)-Machine Code")
36.answer=input("Answer: ")
37.c_ans='c'
38.if(answer==c_ans):
39.     print("correct answer")
40.     score=score+1
41.else:
42.     print("wrong answer")
```

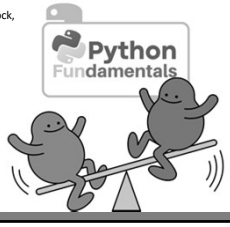
```
43.    print("The correct answer is ",c_ans,"\n")
44.print("[1,2,34,50,67,890,-49] This list include only:")
45.print("(a)-integers, (b)-floats, (c)-strings")
46.answer=input("Answer: ")
47.c_ans='a'
48.if(answer==c_ans):
49.     print("correct answer")
50.     score=score+1
51.else:
52.     print("wrong answer")
53.     print("The correct answer is ",c_ans,"\n")
54.print("This statement terminates the loop containing it.")
55.print("(a)-while, (b)-break, (c)-for")
56.answer=input("Answer: ")
57.c_ans='b'
58.if(answer==c_ans):
59.     print("correct answer")
60.     score=score+1
61.else:
62.     print("wrong answer")
63.     print("The correct answer is ",c_ans,"\n")
64.print("Score = ",score,"\nGreat Job",name)
```

PROJECT 2 – ROCK, PAPER, SCISSORS

Welcome to project 2, in this project we will build a rock, paper, scissors game that can be played in single player mode (user vs computer) and multiplayer mode (user vs user).

STEP ONE: Now let's think of how the game works, and how to win a round.

- Each player should choose one of the three (Rock, Paper or Scissors) at the sa]me time
- (Rock vs Scissors) Rock smashes scissors
- (Paper vs Rock) Paper covers rock
- (Scissors vs Paper) Scissors cuts paper
- (Same choice) draw



```
score=0
name=input("Enter your name: ")
print("\nQUIZ\n")
print("What is Python?")
print("(a)-programming language, (b)-scripting
language, (c)-human language")
answer=input("Answer: ")
c_ans='a'
if(answer==c_ans):
    print("correct answer")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n")
print("When was Python developed?")
print("(a)-1980, (b)-1991, (c)-1982")
answer=input("Answer: ")
c_ans='b'
if(answer== c_ans):
    print("correct answer\n")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n")
print("A procedure used for solving a problem or
performing a computation:")
print("(a)-programming language, (b)-programming,
(c)-algorithm")
answer=input("Answer: ")
c_ans='c'
if(answer==c_ans):
    print("correct answer")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n")
```

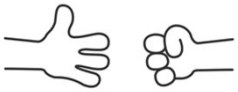
FULL CODE: Download Now

```
print("Which of the following is not a High Level
Programming Language")
print("(a)-Java, (b)-Python, (c)-Machine Code")
answer=input("Answer: ")
c_ans='c'
if(answer==c_ans):
    print("correct answer")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n")
print("[1,2,34,50,67,890,-49] This list include
only:")
print("(a)-integers, (b)-floats, (c)-strings")
answer=input("Answer: ")
c_ans='a'
if(answer==c_ans):
    print("correct answer")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n")
print("This statement terminates the loop containing
it:")
print("(a)-while, (b)-break, (c)-for")
answer=input("Answer: ")
c_ans='b'
if(answer==c_ans):
    print("correct answer")
    score=score+1
else:
    print("wrong answer")
    print("The correct answer is ",c_ans,"\n")
print("Score = ",score,"\nGreat Job",name)
```

STEP TWO: When playing with the computer the user can choose rock, paper, or scissors and the computer should choose a random one, and according to the choice of each player. Using your previous knowledge think how of how we could build that. For choosing a random action we will use the random python module (check functions suitable for the needed task in the module documentation).

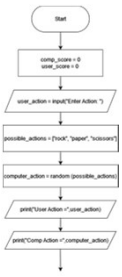
To sum up the game should work as following

- user choose rock, paper, or scissors
- computer choose random action from rock, paper, or scissors
- Conditions (finding the winner)
- Winner score increases by 1



Now try presenting this algorithm in a flowchart

STEP THREE: Drawing flowchart



STEP FOUR: Create the code for user action and computer action

```
#Importing libraries & modules
import random
import string
#User input
user_action = input("\nEnter a choice (rock, paper, scissors): ")
user_action=user_action.lower() #to ignore any uppercase letter inputted
#list for possible actions
possible_actions = ["rock", "paper", "scissors"]
#computer random input from possible actions list
computer_action = random.choice(possible_actions)
#printing the actions
print("\nYou chose ",user_action,", computer chose", computer_action,"\n")
```

STEP FIVE: Create the code for selecting the winner according the chosen actions

```
#creating scores variables
comp_score=0
user_score=0
#case of similar actions
if user_action == computer_action:
    print("Both players selected", user_action," It's a tie!")
```

```
#case user chose rock
elif user_action == "rock":
    if computer_action == "scissors":
        print("Rock smashes scissors! You win!")
        user_score=user_score+1
    else:
        print("Paper covers rock! You lose.")
        comp_score=comp_score+1
```

STEP FIVE: Create the code for selecting the winner according the chosen actions

```
#user chose paper
elif user_action == "paper":
    if computer_action == "rock":
        print("Paper covers rock! You win!")
        user_score=user_score+1
    else:
        print("Scissors cuts paper! You lose.")
        comp_score=comp_score+1
```

```
#case user chose scissors
elif user_action == "scissors":
    if computer_action == "paper":
        print("Scissors cuts paper! You win!")
        user_score=user_score+1
    else:
        print("Rock smashes scissors! You lose.")
        comp_score=comp_score+1
```

STEP SIX: Adding more rounds, after we build the procedure for a single round, we need to repeat that for more rounds (until the user choose to exit). To build that we would need to add all the previous program in a while loop, and move the variables to outside the loop.

```
1. import random
2. import string
3. user_score=0
4. comp_score=0
5. user_action=1
6. while(user_action != 'x'): #the game would stop whenever the user type in x
7. #below (in the body of the loop) is the existing code for scoring
8.     user_action = input("\nEnter a choice (rock, paper, scissors): ")
9.     user_action=user_action.lower()
10.    possible_actions = ["rock", "paper", "scissors"]
11.    computer_action = random.choice(possible_actions)
12.    print("\nYou chose ",user_action,", computer chose", computer_action,"\n")
13.    #choosing winner & scoring
14.    if user_action == computer_action:
15.        print(f"Both players selected {user_action}. It's a tie!")
16.    elif user_action == "rock":
17.        if computer_action == "scissors":
18.            print("Rock smashes scissors! You win!")
19.            user_score=user_score+1
20.        else:
21.            print("Paper covers rock! You lose.")
22.            comp_score=comp_score+1
23.    elif user_action == "paper":
24.        if computer_action == "rock":
25.            print("Paper covers rock! You win!")
26.            user_score=user_score+1
27.        else:
28.            print("Scissors cuts paper! You lose.")
29.            comp_score=comp_score+1
30.    elif user_action == "scissors":
31.        if computer_action == "paper":
32.            print("Scissors cuts paper! You win!")
33.            user_score=user_score+1
34.        else:
35.            print("Rock smashes scissors! You lose.")
36.            comp_score=comp_score+1
```

STEP SEVEN: Printing the score & choosing the winner

```
1. #This code should be placed after the loop, as it should occur when user type
   x
2. print“user score = ",user_score)
3. print“computer score = ",comp_score)
4. #user won
5. if(user_score>comp_score):
6.     print“You won, congratulations”)
7. #user lost
8. elif(user_score<comp_score):
9.     print(”Computer won, Good Luck next time”)
10. #draw
11. else:
12.    print“It’s a draw”)
```

STEP EIGHT: Improving the User experience & finishing the single-player code

```
1. import random
2. import string
3. #Show the name of the game, and how to exit at the beginning of the game
4. print("rock, paper, scissors game")
5. print("At anytime type in x to exit the code")
6. comp_score=0
7. user_score=0
8. user_action=1
9. while(user_action != 'x'):
10.    user_action = input("\nEnter a choice (rock, paper, scissors): ")
11.    user_action=user_action.lower()
12.    possible_actions = ["rock", "paper", "scissors"]
13.    computer_action = random.choice(possible_actions)
14.    print("\nYou chose ",user_action,", computer chose",
   computer_action,"\n")
15.    #choosing winner & scoring
16.    if user_action == computer_action:
17.        print(f"Both players selected {user_action}. It's a tie!")
```

```
    elif user_action == "rock":
        if computer_action == "scissors":
            print("Rock smashes scissors! You win!")
            user_score=user_score+1
        else:
            print("Paper covers rock! You lose.")
            comp_score=comp_score+1
    elif user_action == "paper":
        if computer_action == "rock":
            print("Paper covers rock! You win!")
            user_score=user_score+1
        else:
            print("Scissors cuts paper! You lose.")
            comp_score=comp_score+1
    elif user_action == "scissors":
        if computer_action == "paper":
            print("Scissors cuts paper! You win!")
            user_score=user_score+1
        else:
            print("Rock smashes scissors! You lose.")
            comp_score=comp_score+1
```

```
print("user score = ",user_score)
print("computer score = ",comp_score)
if(user_score>comp_score):
    print("You won, congratulations")
elif(user_score<comp_score):
    print("Computer won, Good Luck next time")
else:
    print("It's a draw")
```

```
import random
import string
print("rock, paper, scissors game")
print("At anytime type in x to exit the code")
comp_score=0
user_score=0
user_action=1
while(user_action != 'x'):
    user_action = input("\nEnter a choice (rock, paper, scissors): ")
    user_action=user_action.lower()
    possible_actions = ["rock", "paper", "scissors"]
    computer_action = random.choice(possible_actions)
    print("\nYou chose ",user_action,", computer chose", computer_action,"\n")
    if user_action == computer_action:
        print(f"Both players selected {user_action}. It's a tie!")
    elif user_action == "rock":
        if computer_action == "scissors":
            print("Rock smashes scissors! You win!")
            user_score=user_score+1
        else:
            print("Paper covers rock! You lose.")
            comp_score=comp_score+1
```

FULL CODE: Download Now

```
    elif user_action == "paper":
        if computer_action == "rock":
            print("Paper covers rock! You win!")
            user_score=user_score+1
        else:
            print("Scissors cuts paper! You lose.")
            comp_score=comp_score+1
    elif user_action == "scissors":
        if computer_action == "paper":
            print("Scissors cuts paper! You win!")
            user_score=user_score+1
        else:
            print("Rock smashes scissors! You lose.")
            comp_score=comp_score+1
print("user score = ",user_score)
print("computer score = ",comp_score)
if(user_score>comp_score):
    print("You won, congratulations")
elif(user_score<comp_score):
    print("Computer won, Good Luck next time")
else:
    print("It's a draw")
```

STEP NINE: To create a multiplayer game, you would need to add and modify a few lines of code.

```
#PART 1
import random
import string
print("rock, paper, scissors game")
print("At anytime type in x to exit the code")
user2_score=0
user1_score=0
user1_action=1
user2_action=1
```

```
#PART 2
#The game stops when any player type in x
while(user1_action or user2_action!= 'x'):
    user1_action = input("\nUSER 1 - Enter a choice (rock, paper, scissors): ")
    user1_action=user1_action.lower()
    #This loop enter 100 new lines to hide the action by player 1
    for n in range(0, 100):
        print("\n")
```

STEP NINE: To create a multiplayer game, you would need to add and modify a few lines of code.

```
    user2_action = input("\nUSER 2 - Enter a choice (rock, paper, scissors): ")
    user2_action=user2_action.lower()
    #This loop enter 100 new lines to hide the action by player 2
    for n in range(0, 100):
        print("\n")
    print("\nUser 1 chose ",user1_action,", User 2 chose",
  user2_action,"\n")
```

```
    if user1_action == user2_action:
        print(f"Both players selected {user1_action}. It's a tie!")
    elif user1_action == "rock":
        if user2_action == "scissors":
            print("Rock smashes scissors!")
            user1_score=user1_score+1
        else:
            print("Paper covers rock!")
            user2_score=user2_score+1
```

```
    elif user1_action == "paper":
        if user2_action == "rock":
            print("Paper covers rock!")
            user1_score=user1_score+1
        else:
            print("Scissors cuts paper!")
            user2_score=user2_score+1
    elif user1_action == "scissors":
        if user2_action == "paper":
            print("Scissors cuts paper!")
            user1_score=user1_score+1
        else:
            print("Rock smashes scissors!")
            user2_score=user2_score+1
print("user 1 score = ",user1_score) print("user 2 score = ",user2_score)
if(user1_score>user2_score):
    print("User 1 won, congratulations")
elif(user1_score<user2_score):
    print("User 1 won, congratulations")
else:
    print("It's a draw")
```

```
1.  import random
2.  import string
3.  print("rock, paper, scissors game")
4.  print("At anytime type in x to exit the code")
5.  user2_score=0 user1_score=0
6.  user1_action=1
7.  while(user1_action or user2_action!= 'x'):
8.      user1_action = input("\nUSER 1 - Enter a
    choice (rock, paper, scissors): ")
9.      user1_action=user1_action.lower()
10.     for n in range(0, 100):
11.         print("\n")
12.     user2_action = input("\nUSER 2 - Enter a
    choice (rock, paper, scissors): ")
13.     user2_action=user2_action.lower()
14.     for n in range(0, 100):
15.         print("\n")
16.     print("\nUser 1 chose
    ",user1_action,", User 2 chose",
    user2_action,"\n")
17.     if user1_action == user2_action:
18.         print("Both players selected
    {user1_action}. It's a tie!")
19.     elif user1_action == "rock":
20.         if user2_action == "scissors":
21.             print("Rock smashes scissors!")
22.             user1_score=user1_score+1
23.         else:
24.             print("Paper covers rock!")
25.             user2_score=user2_score+1
```

FULL CODE: Download Now

```
27.   elif user1_action == "paper":
28.       if user2_action == "rock":
29.           print("Paper covers rock!")
30.           user1_score=user1_score+1
31.       else:
32.           print("Scissors cuts paper!")
33.           user2_score=user2_score+1
34.   elif user1_action == "scissors":
35.       if user2_action == "paper":
36.           print("Scissors cuts paper!")
37.           user1_score=user1_score+1
38.       else:
39.           print("Rock smashes scissors!")
40.           user2_score=user2_score+1
41.print("user 1 score = ",user1_score)
42.print("user 2 score = ",user2_score)
43.if(user1_score>user2_score):
44.    print("User 1 won, congratulations")
45.elif(user1_score<user2_score):
46.    print("User 1 won, congratulations")
47.else:
48.    print("It's a draw")
```

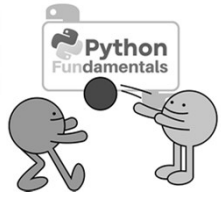
PROJECT 3 – TIC-TAC-TOE

Welcome to project 3, in this project we will build a tic tac toe multiplayer game.

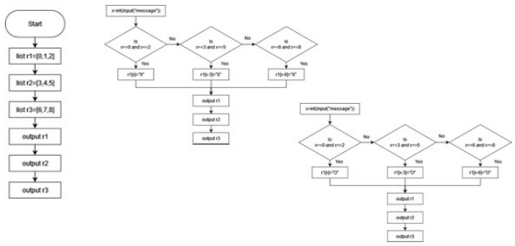
STEP ONE: To build a simpler game. For every slot in the X&O board/tic tac toe, we will give it a number, and when a user choose this number, it will either be replaced by a X or O

STEP TWO: Guess what should be used to build this Tic Tac Toe game slots. Try creating an algorithm for how the game should work.

```
[0, 1, 2]
[3, 4, 5]
[6, 7, 8]
```



STEP THREE: Drawing flowchart



STEP FOUR: Creating and printing the Tic-tac-toe field

```
r1=[0,1,2]
r2=[3,4,5]
r3=[6,7,8]
#A list is used for every row
print(r1)
print(r2)
print(r3)
```

STEP FIVE: X & O placement algorithms

```
x=int(input("Choose a number to add ( X )")) #Choose X placement
if(x>=0 and x<=2):
    r1[x]="X"
elif(x>=3 and x<=5):
    r2[x-3]="X"
elif(x>=6 and x<=8):
    r3[x-6]="X"
```

```
x=int(input("Choose a number to add ( X )")) #Choose O placement
if(x>=0 and x<=2):
    r1[x]="O"
elif(x>=3 and x<=5):
    r2[x-3]="O"
elif(x>=6 and x<=8):
    r3[x-6]="O"
```

STEP SIX: Create a game skip or restart

```
while(x or y!= 'r'):
    #Player 1 turn (X)
    x=int(input("Choose a number to add ( X )"))
    if(x>=0 and x<=2):
        r1[x]="X"
    elif(x>=3 and x<=5):
        r2[x-3]="X"
    elif(x>=6 and x<=8):
        r3[x-6]="X"
    print(r1)
    print(r2)
    print(r3)
  #Player 2 turn (O)
    x=int(input("Choose a number to add ( O )"))
    if(x>=0 and x<=2):
        r1[x]="O"
    elif(x>=3 and x<=5):
        r2[x-3]="O"
    elif(x>=6 and x<=8):
        r3[x-6]="O"
    print(r1)
    print(r2)
    print(r3)
```

STEP SEVEN: Improving the User experience & full code

```
print("Tic-tac-toe\npress n to close the game")
r1=[0,1,2]
r2=[3,4,5]
r3=[6,7,8]
print(r1)
print(r2)
print(r3)
x=y=0
while(x or y!= 'r'):
    x=int(input("Choose a number to add ( X )"))
    if(x>=0 and x<=2):
        r1[x]="X"
    elif(x>=3 and x<=5):
        r2[x-3]="X"
    elif(x>=6 and x<=8):
        r3[x-6]="X"
```

```
17.    print(r1)
18.    print(r2)
19.    print(r3)
20.    x=int(input("Choose a number to add ( O )"))
21.    if(x>=0 and x<=2):
22.        r1[x]="O"
23.    elif(x>=3 and x<=5):
24.        r2[x-3]="O"
25.    elif(x>=6 and x<=8):
26.        r3[x-6]="O"
27.    print(r1)
28.    print(r2)
29.    print(r3)
```

FULL CODE: Download Now

```
1. print("Tic-tac-toe\npress n to close the
   game")
2. r1=[0,1,2]
3. r2=[3,4,5]
4. r3=[6,7,8]
5. print(r1)
6. print(r2)
7. print(r3)
8. x=y=0
9. while(x or y!= 'r'):
10.     x=int(input("Choose a number to add (
   X )"))
11.     if(x>=0 and x<=2):
12.         r1[x]="X"
13.     elif(x>=3 and x<=5):
14.         r2[x-3]="X"
15.  elif(x>=6 and x<=8):
16.      r3[x-6]="X"
17.    print(r1)
18.    print(r2)
19.    print(r3)
20.    x=int(input("Choose a number to add ( O
   )"))
21.    if(x>=0 and x<=2):
22.        r1[x]="O"
23.    elif(x>=3 and x<=5):
24.        r2[x-3]="O"
25.    elif(x>=6 and x<=8):
26.        r3[x-6]="O"
27.    print(r1)
28.    print(r2)
29.    print(r3)
```

PROJECT 4 – TEXT TO SPEECH

Welcome to project 4, in this project we will build a text to speech program, which would narrate the text inputted by the user and/or save it as an .mp3 file.

STEP ONE: To build this program, we will need a few libraries and modules;

- **gTTS:** gTTS (Google Text-to-Speech), a Python library and CLI tool to interface with Google Translate text-to-speech.

Check documentations to find the useful functions

- **OS:** The OS module in Python provides functions for creating and removing a directory, fetching its contents, changing and identifying the current directory, etc. This module will be used to save media files.

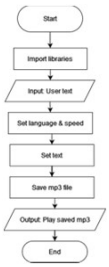
Check documentations to find the useful functions

STEP TWO: Check Example programs

```
#Example 1
import gtts
#Import the gtts library
from gtts import gTTS
#Import the gTTS module from the gtts library
import os
#Import the os module
mytext = "Hello World, Welecome to RGT"
#Function from gTTS
readme = gTTS(text=mytext, lang='en', slow=False)
#saving as file in mp3 format
readme.save("texttospeach.mp3")
#opening saved file
os.system("texttospeach.mp3")
```

STEP THREE: Draw a flowchart + Check more examples & Solve Errors

```
#Example 2
from gtts import gTTS
from playsound import playsound
text = " This is in english language"
var = gTTS(text = text,lang = 'en')
var.save('eng.mp3') playsound('.\eng.mp3')
```



STEP FOUR: Create final program & improve user experience.

FULL CODE: Download Now

```
#Importing libraries
import gtts
from gtts import gTTS
import os
#User input text
mytext = input("Input the text you want to read: ")
#gTTS function
readme = gTTS(text=mytext, lang='en', slow=False)
# .mp3 file saved
readme.save("filename.mp3")
# .mp3 file played
os.system("filename.mp3")
```

PROJECT 5 – TEXT ASSISTANT

Welcome to project 5, in this project we will build a text-based assistant which open websites, webpages, and web files with specific commands. This project is very customizable, so you can setup your very own commands, and actions.

STEP ONE: Check needed libraries and modules.

- **webbrowser:** The webbrowser Python module provides a high-level interface to allow displaying web-based documents (websites and webpages) to users.

Check documentations to find the useful functions

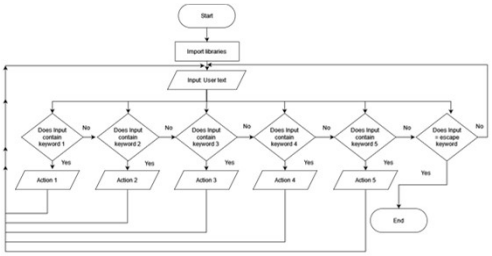
- **OS:** The OS module in Python provides functions for creating and removing a directory, fetching its contents, changing and identifying the current directory, etc.

Check documentations to find the useful functions

STEP TWO: Check Module functions/methods:

Useful methods	
webbrowser.open(url, new = 0, autoraise = true)	This is the main method where the web browser with the passed URL is opened and is displayed to the user. If the parameter "new" is 0 then URL is opened in the same browser and if it is 1 then URL is opened in another browser and if it's 2, then the page is opened in another tab.
webbrowser.open_new(url)	URL passed is opened in the a new browser if it is possible to do that, else it is opened in default one.
webbrowser.open_new_tab(url)	Opens new tab of passpage URL passed in the browser which is currently active.

STEP THREE: Create Algorithms flowchart



STEP FOUR: Open single webpage/website

```
# Import webbrowser Python module
import webbrowser
#URL
url = "https://www.robotsgottalents.com/"
#open url
webbrowser.open(url)
#Note that the website will open in your default browser
```

STEP FIVE: Check for keyword then open website.

```
import webbrowser
import string
ucommand=input("Enter your command: ")
ucommand=ucommand.lower()
if ("robots got talents" in ucommand):
    url = "https://www.robotsgottalents.com/"
    webbrowser.open(url)
```

STEP SIX: Check for keyword then search on Google

```
import webbrowser
import string
ucommand=input("Enter your command: ")
ucommand=ucommand.lower()
if ("search" in ucommand):
    s=input("Input what do you want to search for here: ")
    url = "https://www.google.com/search?q="+s
    webbrowser.open(url)
```

STEP SEVEN: Create final program & improve user experience.

```
#Import libraries
import webbrowser
import string
ucommand=0
while(ucommand != "x"):
    ucommand=input("Enter your command: ")
    ucommand=ucommand.lower()
    #Open RGT Website
    if ("robots got talents" in ucommand):
        url = "https://www.robotsgottalents.com/"
        webbrowser.open(url)
    #Search for something
    elif ("search" or "google" in ucommand):
        s=input("Input what do you want to search for here: ")
        url = "https://www.google.com/search?q="+s
        webbrowser.open(url)
```

STEP SEVEN: Create final program & improve user experience.

FULL CODE: Download Now

```
    #Open YouTube
    elif("youtube" or "video" in ucommand):
        url = "https://www.youtube.com/"
        webbrowser.open(url)
    #Open Facebook
    elif("facebook" in ucommand):
        url = "https://www.facebook.com/"
        webbrowser.open(url)
    #Add your own commands here
    print("Press x at anytime to exit the code")
```

COURSE STUDENT ZONE

This is Robots Got Talents Students Zone, here you will find all the course topics, activities and explanation guides, We wish you enjoy the course, Good Luck

Lesson One	Lesson Two
Lesson Three	Lesson Four
Lesson Five	Lesson Six
Lesson Seven	Bonus Projects