Python Introduction

Pyhtin is the programming language .Computer has two parts Software and hard ware.Hardware are mouse,keyboard (touchable) and software are programming language(untouchable) Interpreter and compiler:Interpreter means if you write a program it excite line by line ,if error occurs then it will not execute the next line of code.But for compiler if you write a program it will execute the code in one go and print the results Python is the interpreter language

In [125]:
```
#PYTHON IS CASE SENSITIVE
```

Number System conversion in Python

In programming world we use two system Binary system and Decimal system.Other than that we use Octal and Hexa Decimalsystem

when we use all these specially with the physical address ,link-local IP

1)Decimal mean base 10(0-9)

2)Binary means base 2(0-1)...bits

3)Octal base 8(0-7)

4)HexaDecimal base 16(0-9 & a -f)

In [20]:
```
bin(25)#bin is the function.convert decimal to binaary
```

Out[20]:
```
'0b11001'
```

In [ ]:
```
```

how to convert a decimal to binary

In [22]:
```
bin(25)
```
Out[22]:
```
'0b11001'
```

```
#how to convert a decimal to binary--so it mathematically below
2|25
2|12-->1
2|6-->0
2|3-->0
 |1-->1

#how to read this 11001
#so see in the above answer 0b means it is binary and 11001 is exactly same
```

how to convert a binary to decimal

```
0b11001
<-------------
 2^4 2^3  2^2 2^1 2^0  =16(1) + 8(1) + 4(0) + 2(0) +1(1) =25
```

how to convert oct,hex,oxf

In [27]:
```
oct(25)# in the result it will show 0o zero O which mean oct
```
Out[27]:
```
'0o31'
```

In [28]:
```
hex(25)
```
Out[28]:
```
'0x19'
```

In [29]:
```
0xf#HexaDecimal base 16(0-9 & a -f)
```
Out[29]:
```
15
```

1)Datatype

Numeric Data(interger & float)

In [6]:
```
#python can print more the one value(integer and floating values)
print(1000,4.56)
```
```
1000 4.56
```

Text Data(String value)

In [10]:
```
print("python")#string are always in interverted comman even though if it the number so then number will
#become string data
```
```
python
```

In [8]:
```
print("124")#it is also string because it is with comma
```
```
124
```

Number and text data(AlphaNumeric Data)

In [11]:
```
print("qbc123")#Alpha numeric data
```
```
qbc123
```

2)Print Statement

print Signature print() (sep-seperator default option as well as end/n default option)

In [12]:
```
print()#click shift and tab together between the bracket
#print has  no limit to print value print.....

print("Pakistan","Python")#comma tell interpreter that it has 2 values with gap between
#as sep " " by default python has
```

```

Pakistan Python
```

sep-seperator default option

In [25]:
```
print("Pakistan","Python" ,sep ="")
```

```
PakistanPython
```

In [14]:
```
print("Pakistan","Python" ,sep ="###")
```

```
Pakistan###Python
```

end\n default option(Escape character \n)

In [16]:
```
print("hello World1")
print("hello World2")
print("hello World3")

#result will show it will print in next line?why
#because "default end \n------ escape charater of python"
```

```
hello World1
hello World2
hello World3
```

In [26]:
```
print("hello World1",end ="\n\n")
print("hello World2",end ="\n\n")
print("hello World3")


#result will show it give two line gap
```

```
hello World1

hello World2

hello World3
```

In [23]:
```
#now here we are asssigning end option without \n so we basically not
#using default option so it will all in one line
print("hello World1",end = "")
print("hello World2",end = "")
print("hello World3")
```

```
hello World1hello World2hello World3
```

"\t"-python Escape character

In [27]:
```
print("hello World1",end ="\t")
print("hello World2",end ="\t")
print("hello World3")
```

```
hello World1    hello World2    hello World3
```

3) Variable

Variable is the place holder ,that holds/stores some value that may change and this value save in our system memory.If we want the stored value we call it through the variable identifier. Variable name Rules:

1)Can't enclose it in quotation marks

2)Can't have any spaces in it.

3)Can't be a number or begin with a number but we can use number like language_1

3.1)Variable for Numeric(Integer & Numeric)

In [77]:
```
a = 10# giving spaces between equal sign is style choice and recommended
#but not legal requirement of python

print(a)
# a is the varaible name(identifier)
# = assignment variable
# is the value that is stored in this varable a here 10 is Integer.
```

```
10
```

In [42]:
```
a = 12.0
print(12.0)
# a is the variable name
# = assignment variable
# floating varible
```

```
12.0
```

Here you will notice the it has printed new value of a by overwriting.So the variable has not its own data type but the value that is stored in it has data type.Let discuss about the data type of the value.

In [43]:
```
#check data type
type(a)
```

Out[43]:
```
float
```

In [55]:
```
#Check data type
b = 10
print(b)
type(b)
```

```
10
```

Out[55]:
```
int
```

String

3.2) Variable for String

In [50]:
```
language = "Python"
#here "a" is the variable name (identifier)
# = is the assignment variable
#python is the value that is stored in this variable a.Also here python is the string.
```

In [54]:
```
#check data type
print(language)
type(language)
```

```
Python
```

Out[54]:
```
str
```

In [53]:
```
language_1 = "Python123"  #using _ is the prefeerred while wririting ame
print(language_1)
type(language_1)
```

```
Python123
```

Out[53]:
```
str
```

multiline string

In [57]:
```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

Strings are Arrays

In [58]:
```
a = "Hello, World!"
print(a[1])
```

```
e
```

Slicing

In [59]:
```
b = "Hello, World!"
print(b[2:5])
```

```
llo
```

Negative Indexing

In [60]:
```
b = "Hello, World!"
print(b[-5:-2])
```

```
orl
```

String Length

In [61]:
```
a = "Hello, World!"
print(len(a))
```

```
13
```

String Methods

String Methods.(The strip() method removes any whitespace from the beginning or the end)

In [62]:
```
a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"
```

```
Hello, World!
```

String Methods.(The lower() method returns the string in lower case)

In [64]:
```
a = "HELLO, World!"
print(a.lower())
```

```
hello, world!
```

String Methods.The upper() method returns the string in upper case)

In [65]:
```
a = "Hello, World!"
print(a.upper())
```

```
HELLO, WORLD!
```

String Methods.The replace() method replaces a string with another string:

In [67]:
```
a = "Hello, World!"
print(a.replace("H", "J"))
```

```
Jello, World!
```

String Methods.The split() method splits the string into substrings if it finds instances of the separator:

In [69]:
```
a = "Hello, World!"
print(a.split(",")) #basically hello ,world was one string now two string
```

```
['Hello', ' World!']
```

Check String

(To check if a certain phrase or character is present in a string, we can use the keywords in or not in.)

In [70]:
```
txt = "The rain in Spain stays mainly in the plain"
x = "ain" in txt
print(x)
```

```
True
```

In [71]:
```
txt = "The rain in Spain stays mainly in the plain"
x = "ain" not in txt
print(x)
```

```
False
```

String Concatenation

In [73]:
```
#Merge variable a with variable b into variable c:
a = "Hello"
b = "World"
c = a + b
print(c)
```

```
HelloWorld
```

In [74]:
```
#To add a space between them, add a " ":
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

```
Hello World
```

String Format

In [75]:
```
#As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:
age = 36
txt = "My name is John, I am " + age
print(txt)

#it will give u error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-75-4403ad1734a6> in <module>
      1 #As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:
      2 age = 36
----> 3 txt = "My name is John, I am " + age
      4 print(txt)
      5 

TypeError: can only concatenate str (not "int") to str
```

```
But we can combine strings and numbers by using the format() method!

The format() method takes the passed arguments, formats them, and
The format() method takes the passed arguments, formats them, and
places them in the string where the placeholders {} are:
```

Use the format() method to insert numbers into strings:

In [76]:
```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

```
My name is John, and I am 36
```

The format() method takes unlimited number of arguments, and are placed into the respective placeholders:

In [77]:
```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

```
I want 3 pieces of item 567 for 49.95 dollars.
```

You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

In [78]:
```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

```
I want to pay 49.95 dollars for 3 pieces of item 567.
```

Escape Character

In [79]:
```
#You will get an error if you use double quotes inside a string that is surrounded by double quotes:

txt = "We are the so-called "Vikings" from the north."
```

```
  File "<ipython-input-79-9a22edd7efc5>", line 3
    txt = "We are the so-called "Vikings" from the north."
                                       ^
SyntaxError: invalid syntax
```

In [82]:
```
#To fix this problem, use the escape character \":
#The escape character allows you to use double quotes when you normally would not be allowed:

txt = "We are the so-called \"Vikings\" from the north."
txt
```

Out[82]:
```
'We are the so-called "Vikings" from the north.'
```

title

4)Maths Operations(+,- ,*,/ ---- operaters-Familiar variable)

in python during addition,subtraction,multiplication,division if one value is float then answer will be in float

In [59]:
```
#common operartor (there are 4 common operater with variable or without variable)
#operator without variable
print(1+2)
```

```
3
```

In [60]:
```
#operator with variable
a = 1
b = 2
print(a+b)
```

```
3
```

In [61]:
```
#operator with variable
a = 1
b = 2
c = a + b
print(c)
```

```
3
```

4.1)Subtraction

In [63]:
```
value_1 = 100
value_2 =  10.5
difference = value_2 -value_1
print(difference)
#note here the results shows in float because in python during addition,subtraction,multiplication,division
#value is float then answer will be in float
```

```
-89.5
```

4.2)Addition

In [64]:
```
value_1 = 100
value_2 =  10.5
sum = value_2  + value_1
print(sum)
```

```
110.5
```

4.3)Division

In [65]:

```
value_1 = 100
value_2 = 100
division =  value_2/value_1
print(division)
#note here the results shows in float because  python does division with
#float values by default so result is in float.
#python does floating point division
```

```
1.0
```

In [70]:

```
value_1 = 100.0
value_2 = 100
division =  value_2/value_1
print(division)
#note here the results shows in float because  python does division
#with float values by default so result is in float.
#python does floating point division
```

```
1.0
```

In [71]:

```
value_1 = 100
value_2 = 100
division =  value_2//value_1 #integer divsion
print(division)
#note here the results shows in integer because  // discard
#the fractional part of the answer like it discard .0
```

```
1
```

In [72]:

```
value_1 = 105
value_2 = 23
division =  value_2//value_1 #integer divsion or floor division
print(division)#result will discard the decimal part
```

```
0
```

In [73]:
```
value_1 = 105
value_2 = 23
division =  value_2/value_1
print(division)#result will include the decimal part as we use / division
```

```
0.21904761904761905
```

4.4)multiplication

In [68]:
```
#multiplication
value_1 = 100
value_2 = 100
multiplication =  value_2*value_1
print(multiplication)
```

```
10000
```

In [69]:
```
value_1 = 100.0
value_2 = 100
multiplication =  value_2*value_1
print(multiplication)
```

```
10000.0
```

5)Variable Names legal and illegal

Reserved Words:

There are some reserved words in python and they are keep increasing with new version of 3.7

For Example and,false,print,return,while,for,class,break,def,del,if,else,import,while,raise,and,as (many more words are adding)

We dont need to memorize the names if we use python will automatically give you and error.

A varaible name contain lowercase,uppercase,number and underscore but becareful python is case sensitive language.

6)Math Expressions (Unfamiliar operators)-(CONTD TO A11)

6.1)Modulus Operator(%):

It divided one number with the other number and show only reminder.

Also if one number is evenly divided by the other number than it will show no reminder.

Also if we are dividing small number with big number then answer will be the small number as a reminder.

In [80]:
```
what_ever_left_1 = 10 % 4
print(what_ever_left_1)
```
```
2
```

In [81]:
```
what_ever_left_2 = 25 % 5
print(what_ever_left_2)
```
```
0
```

In [84]:
```
what_ever_left_2 = 6 % 25
print(what_ever_left_2)
```
```
6
```

6.2)Change value of a variable by some number using addition,mutliplication,divsion and subtraction

In [105]:
```
#addition
age = 10
age = age + 1
print(age)
```
```
11
```

In [91]:
```
#another way of writing above code
age_1 = 23
age_1 += 1
print(age_1)
```
```
24
```

In [93]:
```
#multiplication

age_1 = 10
age_1 = age_1 * 2
print(age_1)
```
```
20
```

In [95]:
```
#another way of writing above code
age_1 = 10
age_1 *= 2
print(age_1)
```
```
20
```

In [97]:
```
#Subtraction
weight = 23
weight = weight - 2
print(weight)
```
```
21
```

In [98]:
```
#another way of writing above code
weight = 23
weight -=2
print(weight)
```
```
21
```

In [99]:
```
#Divsion
weight = 23
weight = weight / 2
print(weight)
```
```
11.5
```

In [100]:
```
#another way of writing above code
weight = 23
weight /=2
print(weight)
```
```
11.5
```

7)Math Expression :Eliminating Ambiguity

Python uses the standard order of operations.

That is, mathematical expressions are evaluated in the following order (as PEMDAS)

PEMDAS: parenthesis,exponental,multiplication,division,addition,subtraction

In [102]:
```
result_computation = ((2*2)*4 +2)
print(result_computation)
```
```
18
```

8)Variable concatenation:

Combine two strings with + sign is called concatenation.

Concatenation is not for done with Numeric Data

In [106]:
```
student_name = "Mark"
father_name = "Tony"
#concatenate both variables
print(student_name+father_name)
```

```
MarkTony
```

In [104]:
```
var_2 = "Blue" + "Colour" + 2
print(var_2)
#it will gives u the error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-104-b8eb708d820e> in <module>
----> 1 var_2 = "Blue" + "Colour" + 2
      2 print(var_2)

TypeError: can only concatenate str (not "int") to str
```

9)IF -STATEMENTS(Test for a condition)

In [111]:
```
a = 100
b = 200
# we want to use logical operator
if a<b:
    #indention help to read the code block of if in this case
    print("a is less than b")
    print("Hello a")
    print("Hello b")
```

```
a is less than b
Hello a
Hello b
```

In [113]:
```
a = 100
b = 200
# we want to use logical operator
if a>b:
    #indention help to read the code block of if in this case
    print("a is less than b")
    print("Hello a")
    print("Hello b")
# this block has not been executed because it doesnot satify the situation.
```

In [115]:
```
a = 100
b = 200
if a>b:
    #indention help to read the code block of if in this case
    print("a is less than b")
#now indent block has been out
print("Help python")
```

```
Help python
```

10)Comparision Operators (==,!=, <,>,<=,>=)

Equality Operator ==

1)We can use equality operator varaible with a string

2)we can use equality operator variable with a numeric

3)We can use eequality operator variable with varaiable

4)We can use equality operator variable with math expression

```
#examples of the above statements
if full_name == "Mark" + "Tony":
if student_id == 12344:
if father_name == mark + tony:
if x+y == a+b:
```

A-11) Extra -Python Operators (New as well Already covered above for revision)

Python Operators Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

1)Arithmetic operators

2)Assignment operators

3)Comparison operators

4)Logical operators

5)Identity operators

6)Membership operators

7)Bitwise operators

1)Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

In [49]:
```
#how to paste the image into the python file

#![title](Capture.png)#always keep this thing in mind the it
#should be in markdown as in the same working directory
#like pytjon file is
```

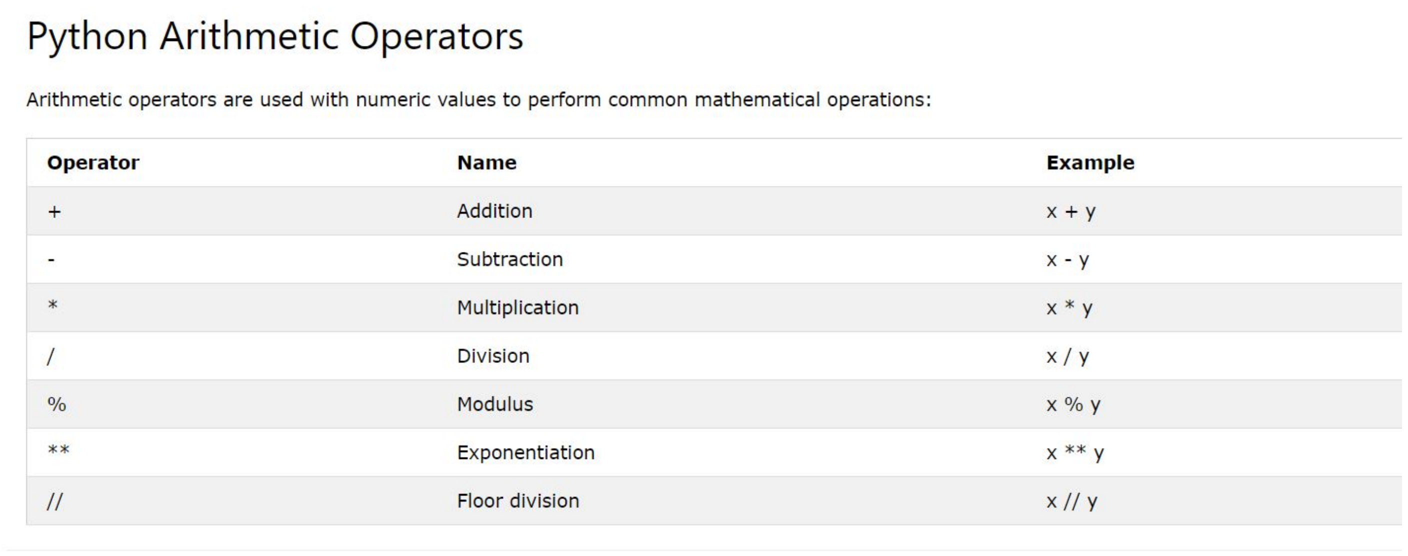
title

In [1]:
```
10/ #divsion
```
Out[1]:
```
3.3333333333333335
```

In [2]:
```
10//3#floor divion or integer division... it show number before decimal from the result
```
Out[2]:
```
3
```

In [3]:
```
10%3#modulus it only show remainder in dvision
```
Out[3]:
```
1
```

In [47]:
```
#how to transfer the image in the python file -method 2
from IPython.display import Image
Image(filename = "Capture.png",width =700 ,height = 700)
```
Out[47]:



Python Arithmetic Operators

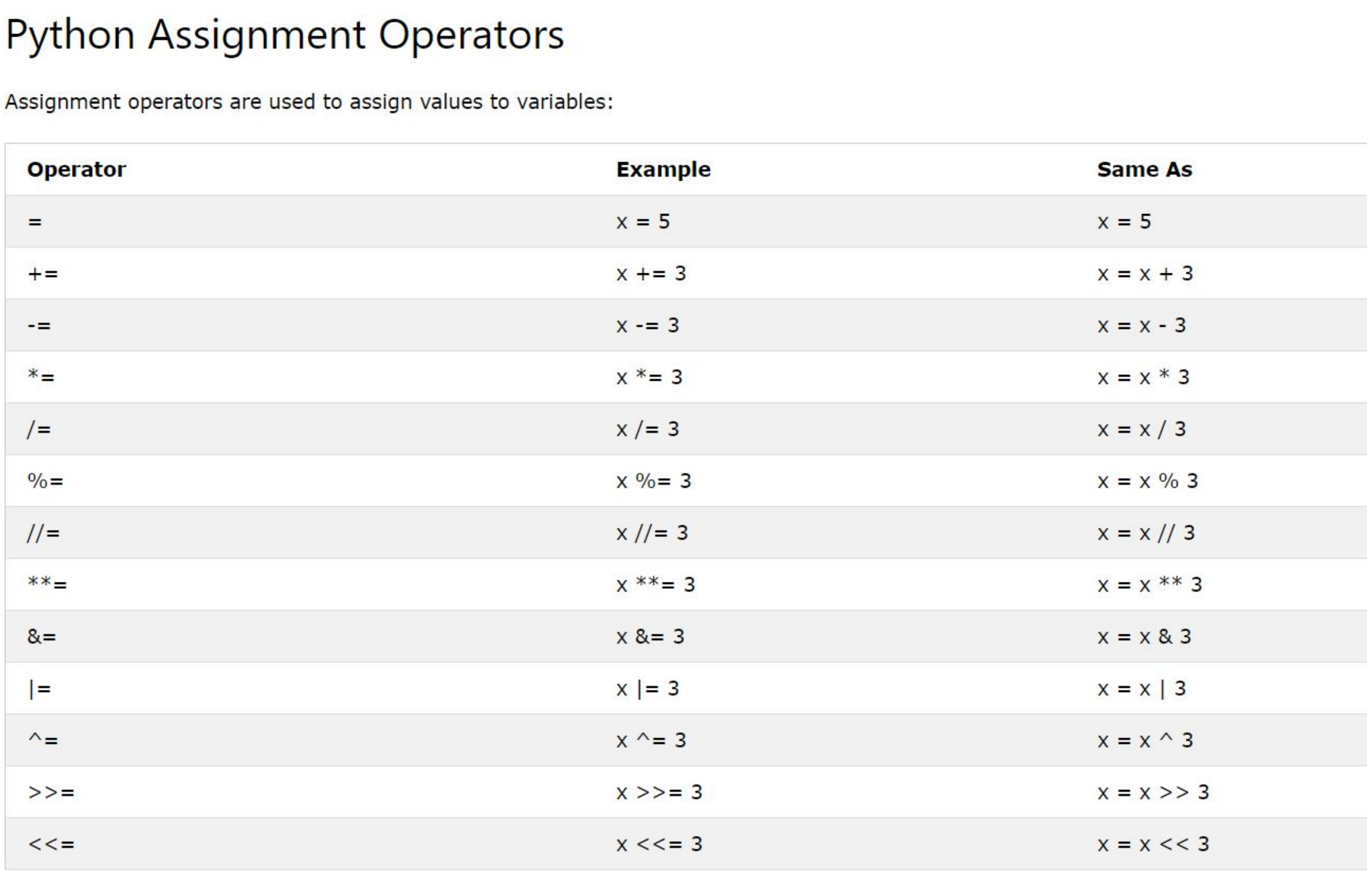
Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

In [ ]:

In [53]:
```
#how to transfer the image in the python file -method 2
from IPython.display import Image
Image(filename = "Capture1.JPG",width =900 ,height = 900)
```

Out[53]:



Python Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

title

title

11)Else and Elif Statements(test condition)

1)if statement is not statisfy the condition then execute the elif statement ,if not satisfy elif then else.

2)if statement is stafying then code will not read elif and else part of code

3)we don't define any condition in else

In [135]:
```
#here if statement is not statisfy the condition then execute the else statement ,
#if statement is stafying so it print it otherwise else


species = "dog"
if species == "cat":
    print("Yes it is the cat")
else:
    print("Nopes,it is not a cat")
```

```
Nopes,it is not a cat
```

In [144]:
```
#here if,elif,else statement
donut_condition = "bad"
if  donut_condition == "fresh":#here you can see you have used "== "sign with if else statement
     print("buy = 10")
elif donut_condition =="Fresh and cheap":
     print("buy = 20")
else:
    print("buy = 0")
#here it will not print any thing
```

```
buy = 0
```

In [143]:
```
donut_condition = "Fresh and cheap"
if  donut_condition == "fresh":
    print("buy = 10")
elif donut_condition =="Fresh and cheap":
    print("buy = 20")
else:
    print("buy = 0")
```

```
buy = 20
```

12)Testing sets of Conditions

1)"and" means satisfy all given conditions.

2)"or" means satisfy any given condition.

3)Combination of "or" & "and" togethor when you want to create ambiguities.

In [150]:
```
#"and" set of conditions------Satisfying all given conditions

weight = 310
time = 5

if weight > 300 and time < 6:
    print("status = try to recruit him")
```

```
status = try to recruit him
```

In [151]:
```
#"and" set of conditions

weight = 200
time = 5

if weight > 300 and time < 6:
    print("status = try to recruit him")
#It will not print any thing as it is not satisfying both conditions
```

In [152]:
```
# "or" set of conditions-------satisfying "any" given condition.

weight = 200
time = 5

if weight > 300 or time < 6:
    print("status = try to recruit him")
#result will show it will print value as it is satifying atleast one situation.
```

```
status = try to recruit him
```

In [153]:
```
weight = 200
time = 7

if weight > 300 or time < 6:
    print("status = try to recruit him")
#it will not print any thing  as it is not satifying "any" "given condition.
```

In [156]:
```
#combined conditions of "or" & "and"
""""sfefefjeriofheriof
jnfjnfjkernfjkerkgn
"""""

age =19
resident = "UK"
if (age < 25 and age > 18) or resident == "UK":
    print("Hire for our team")
#here it is satifying any condition
```

```
Hire for our team
```

In [157]:
```
age = 19
resident = "UK"
if (age < 25 and age > 18) and resident == "UK":
    print("Hire for our team")
```

```
Hire for our team
```

In [158]:
```

age = 19
resident = "UK"
if age > 65 or (age < 21 and resident == "UK"):
    print("Good job")
```

```
Good job
```

13)If statements nested

In [231]:
```
x = 10
y = 111
df = 14
fg = 14
gh =34
rj =32
hj = 43
if x <= y:
    if df == fg:
        print("g =j")
    elif gh == ek:
        print("df =rj")
    else:
        print("hj")
else:
    print("hello World")
```

```
g =j
```

In [165]:

```
x = 10
y = 10
df = 14
fg = 20
gh =34
ek = 30
if x == y:
    if df == fg:
        print("g =j")
    elif gh == ek:
        print("df =rj")
    else:
        print("hj")
else:
    print("hello World")
```

```
hj
```

In [171]:

```
good_weather = "123"
pleasent = "123"
rain = "34"
mild ="45"
heavy ="90"
if good_weather == pleasent:
    if rain == heavy:
        print("wow")
    elif rain == mild:
        print("okay")
    else:
        print("interesting")

else:
    print("bad")
```

```
interesting
```

14)Comments

Comments are the lines of the text in our code that python ignores.

Comments are for human not for machine.

1)single comment can be commented by using # symbol

2)Multi/paragraph comments can be commented by triplequote.

In [177]:
```
# this is the single line comment

#this is multi line comment
#this is the multi line comment

"""this is the paragraph comment
in which a programmmer can write detail of the code for her/his understanding ,not for machine
"""
```

Out[177]:
```
'this is the paragraph comment\nin which a programmmer can write detail of the code for her/his understandi
ng ,not for machine\n'
```

14.1)User Input

In [ ]:
```
input()python provides the key word
```

15-19)Lists :

1)A list is the data structure in python that is mutable or changeable ,orderedsequence of elements.

2)Each element inside the list is callled "item".so bunch of values in a single variables.That can be identified by using "index". In List the first value has the index "0","1" and so on(if we read the list form left to right.... but if we read the list from right to left index will start from "-1"

3)list can be define with square bracket .[] and every value is seperated by ","

4)list can have mixed data type like string,float,integer but array has only one datatype.

5)We can do multiple operations with the list..like

(i)Access value through index

(ii)Add new value/element(end of the list as well as the middle of the list)

(iii)Find index of a value in the list

(iv)Slicing the elements from list,

(v)Deleting and removing the elements from the list(last,middle,first)

(vi)Popping element from the list.

(vii)Making copies of the list

In [311]:
```
#list variable is cities and elements are the countries name.
cities = ["Altantic","Chicago","UK","NY"]
#index       0          1        2(from left to right)
#index       -3         -2       -1(from right to left)
print(cities)
```
```
['Altantic', 'Chicago', 'UK', 'NY']
```

In [312]:
```
#how to acces the member/element/value of thr list
cities[0]
```
Out[312]:
```
'Altantic'
```

In [313]:
```
cities[1]
```
Out[313]:
```
'Chicago'
```

In [314]:
```
cities[2]
```
Out[314]:
```
'UK'
```

In [315]:
```
cities[-1]
```
Out[315]:
```
'NY'
```

In [316]:
```
#len of the list
len(cities)
```
Out[316]:
```
4
```

In [221]:
```
fruits =[]#empty list
len(fruits)
```
Out[221]:
```
0
```

append() adding new values in the empty list ,also for adding/appending values in the end of the existing list

In [222]:
```
#adding member in the empty list-----(always use () with function)
#after writing fruits put . and then press"tab button)
#append function
fruits.append("apple")
fruits
```
Out[222]:
```
['apple']
```

In [225]:
```
#append ()function always append in the end of the list.
fruits.append("orange")
fruits
#if we keep run this code it will keep append orange in the end as I  ran this code thrice thats why orange
```
Out[225]:
```
['apple', 'orange', 'orange', 'orange']
```

insert() for adding/appending value in the specific index of the existing list

In [226]:
```
#insert()----it ask what element has to be added in which location.

fruits.insert(2,"mango")
fruits
```
Out[226]:
```
['apple', 'orange', 'mango', 'orange', 'orange']
```

extend()for adding/appending multiple values in the list

```
In [227]: fruits.extend(["pineapple","banana"])
          fruits

Out[227]: ['apple', 'orange', 'mango', 'orange', 'orange', 'pineapple', 'banana']
```

count()for counting the number of specific element in the list

```
In [228]: fruits.count("orange")
          #this is showing orange is 3 times in the list of fruits

Out[228]: 3
```

index() to find the index of the value/element in the list

```
In [229]: fruits.index("pineapple")

Out[229]: 5
```

clear() it clear all the element/value in the list and it will clear it permanaetly

```
In [299]: fruits.clear()
          fruits

Out[299]: []
```

```
In [300]: #As now our list is clesr now so we are making new list
          fruits= ["apple","cherry",'blueberry']
          fruits

Out[300]: ['apple', 'cherry', 'blueberry']
```

copy() it copy by value that means

In [301]:
```
#by using copy()
fruits_2 =fruits.copy()# by value it make seperate copy
fruits_2
```
Out[301]:
```
['apple', 'cherry', 'blueberry']
```

In [302]:
```
#by only reference
fruits_3 =fruits#by refernce only  fruits refernce  but not the value
fruits_3
```
Out[302]:
```
['apple', 'cherry', 'blueberry']
```

In [ ]:
```

```

In [303]:
```
#lets check what is the differnece between copy() or just only refernce
#with copy() it will work for append is called by pass by refernce
#with refence it will not work for append is called pass by refernce
fruits.append("newfruit")
fruits_3#as fruits_3 is refering fruits so main list is referencing with sp fruit_3 and fruits has same outp
```
Out[303]:
```
['apple', 'cherry', 'blueberry', 'newfruit']
```

In [304]:
```
#now lets check fruits_2 that is the copy of the fruit it has no impact as it is seperate copy
fruits_2
```
Out[304]:
```
['apple', 'cherry', 'blueberry']
```

In [ ]:
```

```

del() is the statement ,,it will require index to delete permanetly

In [305]:
```
#removing an item from the list through del () we use index number
del fruits[1]
fruits
```

Out[305]:
```
['apple', 'blueberry', 'newfruit']
```

remove() is the function .it requre value to remove item from the list

In [306]:
```
fruits.remove("newfruit")
fruits
```

Out[306]:
```
['apple', 'blueberry']
```

pop() it will remove the value through without index & index both but not remove permanetly so we can save that deleted value.

In [322]:
```
cities = ["Altantic","Chicago","UK","NY","KSA"]
```

In [323]:
```
#popped without index
popped_city = cities.pop()
print(f"This city is popped from list{popped_city}")
print(f"This remaining cities are {cities}")

#pop mean it will remove the last item in the list or we can say pop last item from the list...like stacked
```

```
This city is popped from listKSA
This remaining cities are ['Altantic', 'Chicago', 'UK', 'NY']
```

In [324]:
```
#popped with index
popped_city = cities.pop(0)
print(f"This city is popped from list{popped_city}")
print(f"This remaining cities are {cities}")
```

```
This city is popped from listAltantic
This remaining cities are ['Chicago', 'UK', 'NY']
```

sort()

In [325]:
```
cities
```

Out[325]:
```
['Chicago', 'UK', 'NY']
```

In [326]:
```
cities.sort(reverse = True) #descending order sorting
```

In [327]:
```
cities
```

Out[327]:
```
['UK', 'NY', 'Chicago']
```

reserve()

In [328]:
```
cities.reverse()
cities
```

Out[328]:
```
['Chicago', 'NY', 'UK']
```

Slicing

In [332]:
```
#index       -5      -4        -3       -2        -1
students = ["Ali","Faisal","Saleem","Hamza","Kashif"]
#index       0       1         2        3        4

#it means list has two index positive as well as negative.
```

In [335]:
```
#it means list has two index positive as well as negative.
print(students[0])
print(students[-5])
```

```
Ali
Ali
```

In [336]:
```
#students[start:end+1]----if u want fasial and saleem----always see start from left to right
students[1:3]#slicing from the list is the list
```

Out[336]:
```
['Faisal', 'Saleem']
```

In [338]:
```
#----if u want fasial and saleem-----always see start from left to right
students[-4:-2]
```

Out[338]:
```
['Faisal', 'Saleem']
```

In [339]:
```
#we are not giving ant start and end so it will print full list
students[:]
```

Out[339]:
```
['Ali', 'Faisal', 'Saleem', 'Hamza', 'Kashif']
```

In [340]:
```
#if we want to print first 3 elements of the list
students[:3]
```

Out[340]:
```
['Ali', 'Faisal', 'Saleem']
```

In [342]:
```
#if we want to print elements after 3rd elememt
students[3:]
```

Out[342]:
```
['Hamza', 'Kashif']
```

In [343]:
```
#try something different using positive as well as negative slicing both togethor
#keep this thing in mind slicing always read from left to right

students[1:-1]
```
Out[343]:
```
['Faisal', 'Saleem', 'Hamza']
```

In [344]:
```
students[2:-3]
#it will print empty list as it start from 2 and after reaching 4 it will stop as it cannot read from right
```
Out[344]:
```
[]
```

In [345]:
```
#slicing with steps

nums = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
#index= 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
```

In [348]:
```

#first we want to select odd number from the list
 #we will define start index and end index and steps how much we want
nums[0:20:2]
#    (start: (end : (steps)
#   index)   index)
```
Out[348]:
```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

In [350]:
```
nums[1:20:2]
```
Out[350]:
```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

In [351]:
```
nums[::5]#start from beginning(:) end (:) step (5)
```
Out[351]:
```
[1, 6, 11, 16]
```

Change Item Value

In [85]:

```
#To change the value of a specific item, refer to the index number:

thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

```
['apple', 'blackcurrant', 'cherry']
```

Check if Item Exists

In [86]:

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

```
Yes, 'apple' is in the fruits list
```

List Length

In [87]:

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

```
3
```

Join Two Lists

In [88]:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

```
['a', 'b', 'c', 1, 2, 3]
```

In [89]:
```
#ANOTHER WAY OF ADDFING A LIST TOGETHOR THRUGH FOR LOOP
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

for x in list2:
  list1.append(x)

print(list1)
```

```
['a', 'b', 'c', 1, 2, 3]
```

In [ ]:

20)Tuples

Tuples are just like python list except that they are immutabel(unchangeable-like roll number of the student,patient file number).

You can add,delete,and change elements after the creation of the creation of tuples instance.

You can pick a particular element from a tuple the same way you pick an element of a list.

Like from the list tuple index from the 0

Tuple define with () brackets

You cannot add,delete ,remove from the tuple

If we dont use any bracket then it is also the tuple

In [39]:
```
#tuple with bracket
tuple = "a","b",1,"3"
tuple[0]
```

Out[39]:
```
'a'
```

```
In [353]:  #tuple with bracket
           tuple = ("a","b",1,"3")
           tuple[0]

Out[353]:  'a'

In [354]:  tuple[-1]

Out[354]:  '3'

In [355]:  tuple[:]

Out[355]:  ('a', 'b', 1, '3')

In [357]:  tuple[:2]#: is the start

Out[357]:  ('a', 'b')

In [358]:  len(tuple)

Out[358]:  4

In [359]:  del tuple[1]

           #it will give u error as you cannot add,delete ,remove from the tuple

           ---------------------------------------------------------------------------
           TypeError                                 Traceback (most recent call last)
           <ipython-input-359-fa7f8ab7ad51> in <module>
           ----> 1 del tuple[1]

           TypeError: 'tuple' object doesn't support item deletion
```

tuple has only two function count and index

```
In [8]:  tuple =(1,2,3,4,5,5,2,3)
          tuple.count(4)

Out[8]:   1
```

```
In [9]:  tuple.count(5)

Out[9]:   2
```

```
In [10]: tuple.index(3)

Out[10]:  2
```

20.1)Set

set use the concept of the hash

In set we can not do the indexing as it has no proper sequence. set doesnot support dupplicate value

In set which doesnot make sequence but deffiently we can pop ,remove,update

We need to go to the detail of the set in later

https://www.w3schools.com/python/python_sets.asp (https://www.w3schools.com/python/python_sets.asp)

```
In [12]: s ={12,45,65,78,43,67,34,23}# dont think its result is in the sequential order
         s

Out[12]: {12, 23, 34, 43, 45, 65, 67, 78}
```

```
In [14]: s ={12,45,65,34,23,12,78,78,43,67,34,2,983}
         s

Out[14]: {2, 12, 23, 34, 43, 45, 65, 67, 78, 983}
```

```
In [ ]:
```

21) for loop

Python provide for loop to iterate over a sequence (e.g.a list,a tuple and dictionary).Ths is a very handy tool when u traverse all/few elements of a sequence without having to worry about the number of elements in the sequence.

loop start and then end like terminate.

But if we want to terminate the loop "break-keyword" is used we can do by using "if" statement. so it will break the loop or loop terminate permanently.

but if we want to skip iteration of some element from the loop we use "continue keyword" - that will skip that part of the loop and continue to iteratetill the end or given condition.

In [360]:

```
#if we want to print the name 5 times.
print("John")
print("John")
print("John")
print("John")
print("John")
```

```
John
John
John
John
John
```

In [1]:

```
# above-it quite lengthy to do like this so we use loop to  iterate my name 5 times.
#use of "for loop"

for a in range(5):
    print(a)
```

```
0
1
2
3
4
```

In [ ]:
```
#for a-variable in range-function(you need to define how much time ypu need tp run the loop):
   #print("John")
```

In [3]:
```
for a in range(5):
    print("John")
```

```
John
John
John
John
John
```

In [4]:
```
for a in range(5):
    print(a,"John")
```

```
0 John
1 John
2 John
3 John
4 John
```

In [6]:
```
#HW TO PRINT NUMBER FROM 0-9
for num in range(10):
    print(num)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [7]:
```
#NOW WE WANT TO PRINT THE NUMBER FROM 1-10
for num in range(1,11,1):#here 1 is the starting number ,10 is the end number and 1 step
    print(num)
```

```
1
2
3
4
5
6
7
8
9
```

In [8]:
```
#how to print even number through for loop
for num in range(2,11,2):
    print(num)
```

```
2
4
6
8
10
```

In [9]:
```
#similarly print odd number through for loop
for num in range(1,11,2):
    print(num)
```

```
1
3
5
7
9
```

In [16]:

```python
#if we want to print the number in reverse order

for num in range(10,0,-1):
    print(num)
```

```
10
9
8
7
6
5
4
3
2
1
```

In [17]:

```python
#if we want to print the number in reverse order

for num in range(10,-1,-1):
    print(num)
```

```
10
9
8
7
6
5
4
3
2
1
0
```

In [18]:

```
#if we want to print the odd number in reverse order

for num in range(11,1,-2):
    print(num)
```

```
11
9
7
5
3
```

In [20]:

```
#if we want to print the even number in reverse order

for num in range(10,1,-2):
    print(num)
```

```
10
8
6
4
2
```

In [28]:

```
#if u want to print is whole list by 6 times as number of the elemenets in list are 6
countries = ["UK","USA","KSA","PK","UAE","GERMANY"]

for a in countries:
    print(countries)
```

```
['UK', 'USA', 'KSA', 'PK', 'UAE', 'GERMANY']
['UK', 'USA', 'KSA', 'PK', 'UAE', 'GERMANY']
['UK', 'USA', 'KSA', 'PK', 'UAE', 'GERMANY']
['UK', 'USA', 'KSA', 'PK', 'UAE', 'GERMANY']
['UK', 'USA', 'KSA', 'PK', 'UAE', 'GERMANY']
['UK', 'USA', 'KSA', 'PK', 'UAE', 'GERMANY']
```

In [29]:
```
#if u want to print the countries after every single iteration.
countries = ["UK","USA","KSA","PK","UAE","GERMANY"]

for a in countries:
    print(a)
```

```
UK
USA
KSA
PK
UAE
GERMANY
```

In [30]:
```
#if u want to print the countries after every single iteration.
countries = ["UK","USA","KSA","PK","UAE","GERMANY"]

for a in countries:
    print(f"The country under consideration is {a}")
```

```
The country under consideration is UK
The country under consideration is USA
The country under consideration is KSA
The country under consideration is PK
The country under consideration is UAE
The country under consideration is GERMANY
```

In [32]:
```
#Similarly if u want to print the element of the list through for loop
list = [11,22,33,44,55]
for a in list:
    print(a)
```

```
11
22
33
44
55
```

In [33]:
```
#Similarly if u want to print the element of the list through for loop
for a in [11,22,33,44,55]:# we define the list in the for loop
    print(a)
```

```
11
22
33
44
55
```

For Looping through string

In [35]:
```
#if we want to print the character in the single word

country = "Pakistan"

for char in country:
    print(char)
#where it with deal with the single string it will take char one by one
```

```
P
a
k
i
s
t
a
n
```

In [38]:
```
#if we have two countries

countries = "Pakistan","Canada","Oman"

for string in countries:
    print(string)
```

```
Pakistan
Canada
Oman
```

Reference:https://www.geeksforgeeks.org/python-programming-language/ (https://www.geeksforgeeks.org/python-programming-language/)

In [ ]:

In [ ]: