BASIC PYTHON SYNTAX

1) BACK SLASH means Continuation sign

In [5]:
```
print ("hello\
        python")
```
```
hello       python
```

2) TIPLE QUOTES - to produce text art , docs string

In [24]:
```
print("""(\_/)
(•x•)
(<▯ )""")
```
```
(\_/)
(•x•)
(<▯ )
```

3) STRING INSIDE THE QUOTE

In [10]:
```
print ('hello world')
```
```
hello world
```

In [11]:
```
print ('python's world')
```
```
  File "<ipython-input-11-ed3777defdc2>", line 1
    print ('python's world')
                   ^
SyntaxError: invalid syntax
```

Above error is showing that python is detecting english syntax which is 's , for rectifying it we can put double quote as shown below.

In [12]:
```
print ("python's world")
```
```
python's world
```

4) ESCAPE SEQUENCE OF STRING - \n (line seperator) , \t (produce space), \a (bell/alarm sound) and solution to produce outcome without error with single quote using back slash as shown below :

In [13]:
```
print ("hello\npython")
```
```
hello
python
```

In [14]:
```
print ("hello\tpython")
```
```
hello   python
```

In [17]:
```
print ("hello\apython")
```
```
hello▯python
```

In [19]:
```
print('python\'s world')
```
```
python's world
```

5) FORMATED OUTPUT - uses to run multiple statement as shown below:

In [23]:
```
name = "Nishant"
marks = "80"
age = "25"
print ("the name of person is",name , "marks is", marks,"age is", age)
```
```
the name of person is Nishant marks is 80 age is 25
```

if we dont want use so many commas as showing in above code we can replace it with as %s (string), %f(float) , %d(integer) and put relative order like name , marks as shown below:

In [31]:
```
name = "Nishant"
marks = 80.567
age = 25
print("the name of person is %s marks is %f age is %d"%(name,marks,age))
```
```
the name of person is Nishant marks is 80.567000 age is 25
```

Now if we want to put space then we can use this (%10 which means 10 times space this is for example we can use as per our priority like %4s , %5s) and for reducing and increase decimal value we can put .2 , .3, .4, .5 in %f as shown below

In [40]:
```
name = "Nishant"
marks = 80.567
age = 25
print("the name of person is %10s marks is %8.2f age is %7d"%(name,marks,age))
```
```
the name of person is    Nishant marks is    80.57 age is      25
```

uses of f string as show below if dont want to remmeber above code ,but make sure we should use above 3.4 version python

In [42]:
```
name = "Nishant"
marks = 80.567
age = 25
print(f"the name of person is {name} marks is {marks} age is {age}")
```
```
the name of person is Nishant marks is 80.567 age is 25
```

PYTHON VARIABLE

Variable means linking the data to a name

According to a data type ,the interpreter reserves the memory

variable refers to the memory location that contains the data.

for example a=10 , here = is assignment operator Rules 1)if, def , for cannot be use as variablr because they are reserved variables. 2)variable can contains lower and upper case , numbers and underscore.

example : a=10 , A=10 but both variable are different with each other bcoz python is case sensitive.

3)variable cannot be start with number : 10a = 20 which is not a variable 4)variable assigned to data by using the assignment operator which symbolize = this 5)python assignment statement x=y=z=10

In [1]:
```
print ("x=y=z=10")
```
```
x=y=z=10
```

Types of operators

Arithmetic Operators 1)* = *exponent- performs exponential(powers) calculation on operators 2)*= multiplication 3)/ = division 4)% = modulus/remainder 5)+ = addition 6)- substraction 7)//- floor divison means after performing the divison it returns the lower integer value as result

In [3]:
```
5**5
```
Out[3]:
```
3125
```

In [4]:
```
20+10
```
Out[4]:
```
30
```

In [5]:
```
30-40
```
Out[5]:
```
-10
```

In [6]:
```
25/5
```
Out[6]:
```
5.0
```

above code in divison we are getting float value because in python 3 if we divide int/int it will give us float value always

In [9]:
```
10/3
```
Out[9]:
```
3.3333333333333335
```

In [10]:
```
10%3
```
Out[10]:
```
1
```

In [11]:
```
10//3
```
Out[11]:
```
3
```

COMPARISON OPERATORS = ITS ALWAYS RETURNS THE VALUE IN FORM OF BOOLEAN VALUE MEANS TRUE AND FALSE, which are symbolize as double equal to == less than < greater than > less than equal to <= greater than equal to >= not equal to !=

In [14]:
```
a=23
b=25
a==b
```
Out[14]:
```
False
```

In [15]:
```
c=23
a==c
```
Out[15]:
```
True
```

In [16]:
```
c<a
```
Out[16]:
```
False
```

In [17]:
```
c<=a
```
Out[17]:
```
True
```

In [18]:
```
a>b
```
Out[18]:
```
False
```

In [19]:
```
a<b
```
Out[19]:
```
True
```

In [20]:
```
a>=b
```
Out[20]:
```
False
```

In [21]:
```
b>=a
```
Out[21]:
```
True
```

In [22]:
```
c<=a
```
Out[22]:
```
True
```

In [23]:
```
c>=a
```
Out[23]:
```
True
```

In [24]:
```
a!=b
```
Out[24]:
```
True
```

In [25]:
```
a!=c
```
Out[25]:
```
False
```

BITWISE OPERATORS = BITWIESE OR OPERATORS | BITWISE AND OPERATORS &

In [26]:
```
a=240
b=1
a|b
```
Out[26]:
```
241
```

LOGICAL OPERATORS (OR, AND , NOT)

OR = IF ANY SIDE IS TRUE THEN 'OR OPERATORS' RETURNS TRUE AND = IF BOTH SIDE IS TRUE THEN 'AND OPERATORS' RETURNS TRUE NOT = IF THE CONDITION IN THE NOT OPERATORS RETURNS TRUE, THEN THE NOT OPERATORS MAKE IT FALSE , VICE VERSA

In [33]:
```
a= 5
b=25
a>20
```
Out[33]:
```
False
```

In [31]:
```
a>20 or 10>9
```
Out[31]:
```
True
```

In [32]:
```
a>20 and 10>9
```
Out[32]:
```
False
```

In [35]:
```
M
```
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-35-f3e8788da83b> in <module>
----> 1 M

NameError: name 'M' is not defined
```

In [36]:
```
10>9 and 20>M
```
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-36-9b6b05d7a0cf> in <module>
----> 1 10>9 and 20>M

NameError: name 'M' is not defined
```

IN BOTH ABOVE CODE WE ARE GETTING ERROR BECAUSE M IS NOT DEFINED YET .

In [37]:
```
10<9 and 20>M
```
Out[37]:
```
False
```

SO HERE IN ABOVE CODE , PYTHON IS DETECTING FALSE IN FIRST PLACE SO IT WILL ALWAYS GIVE OUTPUT FLASE WHEN LEFT SIDE GIVE ERROR

In [38]:
```
10>9
```
Out[38]:
```
True
```

In [40]:
```
not 10>9
```
Out[40]:
```
False
```

above code is a example NOT operator which is inverting the result.

MEMBERSHIP OPERATOR in = if the specified operand is found in the sequence then the in operator returns true otherwise false not =if the specified operand is not found in the sequence then the in operator returns true otherwise false

In [44]:
```
str1 ="pythonprogramming"
"a" in str1
```
Out[44]:
```
True
```

In [45]:
```
"x" in str1
```
Out[45]:
```
False
```

In [46]:
```
"a" not in str1
```
Out[46]:
```
False
```

In [47]:
```
"x" not in str1
```
Out[47]:
```
True
```

IDENTITY OPERATOR Is = if two variables refer to the same memory location , then the is operator returns true , otherwise false is not = if two variables refer to the same memory location , then the is not operator returns true , otherwise true

In [48]:
```
a=10
b=10
a==b
```
Out[48]:
```
True
```

In [49]:
```
a is b
```
Out[49]:
```
True
```

In [50]:
```
id(a)
```
Out[50]:
```
140717966467760
```

In [51]:
```
id(b)
```
Out[51]:
```
140717966467760
```

above code is giving true outcome becuase both variable have same memory

In [54]:
```
x=2.5
y=2.5
x==y
```
Out[54]:
```
True
```

In [55]:
```
x is y
```
Out[55]:
```
False
```

In [56]:
```
id(x)
```
Out[56]:
```
2902500853904
```

In [57]:
```
id(y)
```
Out[57]:
```
2902500854832
```

now outcome is false because both x and y have different memory

-5 to 256 concept , if the integer lies between this range then the outcome or identity will be true otherwise false

In [58]:
```
a = -6
b = -6
a is b
```
Out[58]:
```
False
```

In [59]:
```
a = 150
b = 150
```