FIRST WORK SHEET

PYTHON BASIC

keywords / Reserved Words

In [1]:
```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'fo
r', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'no
t', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

above are the keywords which we should write in same format to run the code

Identifier

Name given to entity like class, function and variables.

Example:

In [2]:
```
a=2
```

In [3]:
```
print(a)
```

```
2
```

In [4]:
```
X1=10
print(X1)
```

```
10
```

In [5]:
```
Y_=50
print(Y_)
```

```
50
```

Rules in identifier:
Identifier can't start with Numerical. Ex:- 1var=10.
Identifier can't use any special character except underscore(_).
Reserverd words or Keywords, we can't use as a Identifier.

Comment (Readable)

Comment just used to identify something
used to make note on particular code line

Single Line Comment (#)

Example:

In [7]:
```
a=2  #a is variable
print(a)
```

```
2
```

Multi Line Comment (""""XXXXX""")

Example:

In [15]:
```
"""First
class
revision"""          #where, \n in output is next line
```

Out[15]:
```
'First \nclass\nrevision'
```

Indentation (4space/Tab):

Conditional, Statement, if, for - indentation used

Example:

In [17]:
```
X=2
if X==2:
    print('X is equal to 2')
```

```
X is equal to 2
```

In [22]:
```
X=3
Y=9
if Y==9:
    print('Y is twice of X')
```

```
Y is twice of X
```

Statement

Single Line Statement

Example:

In [23]:
```
a=10 #single line statement or coding
print(a)
```

```
10
```

Multi Line Statement (end of the line should include \ )

Example:

In [24]:
```
a=1+2\
+3+4\
+5+6
a
```

Out[24]:
```
21
```

Docstrings

It's explaining the functionality of function

Example:

In [25]:
```
def square(num):
    """square function will return the square of a number"""
    return num**2
```

In [26]:
```
def cube(num):
    """cube function will return the thrice of a number"""
    return num**3
```

In [29]:
```
square(16)
```

Out[29]:
```
256
```

In [30]:
```
cube(16)
```

Out[30]:
```
4096
```

format to see Docstrings

short key - function name + dot + double underscore + doc + double underscore

{function name . __ doc __ }

In [31]:
```
square.__doc__
```

Out[31]:
```
'square function will return the square of a number'
```

In [32]:
```
cube.__doc__
```

Out[32]:
```
'cube function will return the thrice of a number'
```

Variable

In [33]:
```
a=5
b=5
c=a+b

print(c)
```

```
10
```

Memory location of variables
print(id(identifier))

Example:

In [34]:
```
print(id(a))
print(id(b))
print(id(c))
```

```
140721219605416
140721219605416
140721219605576
```

Data type

int, float, bool/boolean, complex

Integer
integers not in decimals and it doesn't mean positive or negative.

Float
any decimal number is float.

Bool
true or False.

Complex

real + imaginary.

In [43]:
```
a=2
b=-2
c=0.2
d=2+2J
```

In [36]:
```
type(a)
```

Out[36]: int

In [37]:
```
type(b)
```

Out[37]: int

In [44]:
```
type(c)
```

Out[44]: float

In [46]:
```
type(d)
```

Out[46]: complex

String ('X')

Can't edit string. if any mistake made, delete the cell/string.

In [49]:
```
Avatar='sullys stick together'
```

In [50]:
```
type(Avatar)
```

Out[50]: str

In [52]:
```
len(Avatar) #including space it will count
```

Out[52]: 21

Indexing [X]

In python always starts with Zero and square bracket should use to run code.

-21.....-1

S U L L Y S S T I C K T O G E T H E R

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Forward indexing '0' to '20'.

In [53]:
```
Avatar[2]
```

Out[53]: '1'

In [54]:
```
Avatar[6]
```

Out[54]: ' '

Slicing [X]

Slicing the word from the sentence, end+1

Example:

In [55]:
```
print(Avatar)
```

```
sullys stick together
```

In [56]:
```
Avatar[7:21]
```

Out[56]: 'stick together'

In [57]:
```
Avatar[0:5]
```

Out[57]: 'sully'

In [58]:
```
Avatar[8:12]
```

Out[58]: 'tick'

In [59]:
```
Avatar[0:]
```

Out[59]: 'sullys stick together'

String Concatenation(combined)

In [62]:
```
a1='sullys'
b1='stick'
c1='together'
```

In [63]:
```
print(a1+b1+c1)
```

```
sullysstricktogether
```

In [64]:
```
print(a1,b1,c1)
```

```
sullys stick together
```

In [65]:
```
print(a1+' '+b1+ ' '+c1)
```

```
sullys stick together
```

String membership

in or not in the membership operator.

In [66]:
```
print(Avatar)
```

```
sullys stick together
```

In [67]:
```
print('sullys' in Avatar)
```

```
True
```

In [68]:
```
print('sulyls' in Avatar)
```

```
False
```

String partitioning

In [69]:
```
print(Avatar)
```

```
sullys stick together
```

In [86]:
```
Avatar.capitalize()  #capitalized starting letter of the sentence.
```

Out[86]:
```
'Sullys stick together'
```

In [88]:
```
Avatar.casefold()  #remain same.
```

Out[88]:
```
'sullys stick together'
```

In [92]:
```
Avatar.count('s')   #count of 3
```

Out[92]:
```
3
```

In [93]:
```
Avatar.encode()
```

Out[93]:
```
b'sullys stick together'
```

In [95]:
```
Avatar.endswith('stick')
```

Out[95]:
```
False
```

In [97]:
```
Avatar.endswith('together')
```

Out[97]:
```
True
```

In [98]:
```
Avatar.expandtabs()
```

Out[98]:
```
'sullys stick together'
```

In [99]:
```
Avatar.partition('stick')
```

Out[99]:
```
('sullys ', 'stick', ' together')
```

In [ ]:
```

```