In [ ]:

- Python Variables
- data types
- type casting
- print statement
    - end
    - sep
    - f string
    - format
- eval
- input
- round
- package information
- Basic codes
- Conditional statements
- **try-except**
- Functions
    - **with** out argument
    - **with** argument
    - default argumnet
    - local vs **global**
    - **return**
    - function **in** functions
- Loop
    - **for**
    - **while**

==========

- String
- list
- tuple
- dictionary

```
- set operations

- list comprehension

- lambda functions

- File handling
=====================================================
```

In [ ]:
```
## Strings
=================== PART-1==========================
- How to intialize the strings

- in built functions

    - type

    - print

    - len

    - max

    - min

    - reversed

    - sorted

- index operation

- concatenation

- mutable vs immutable

- slicing

===================== PART-2 =====================
string methods
```

Intialization

- strings represnt with single quotes
- strings represnt with double quotes
- strings represnt with triple quotes
- when you print a string , the output shows as with out quotes
- A string represnt with triple quotes : **Doc string**
- If you want highlite any word in entire string

- provide the entire string in double quotes and highlight the word with single quotes vice versa

In [1]:
```
str1='python'
```

In [2]:
```
type(str1)
```

Out[2]: str

In [3]:
```
str2="python"
str2
```

Out[3]: 'python'

In [4]:
```
type(str2)
```

Out[4]: str

doc string

In [9]:
```
str3="""hello python
many students not coming to lab
today come to their dreams"""

print(str3)
```

```
hello python
many students not coming to lab
today come to their dreams
```

In [10]:
```
str4="I like 'python'"
str4
```

Out[10]: "I like 'python'"

In [11]:
```
print(str4)
```

```
I like 'python'
```

In [12]:
```
str5='I like "python"'
str5
```

Out[12]: 'I like "python"'

In [13]:
```
print(str5)
```

```
I like "python"
```

some inbuilt functions

In [ ]:
```
max()
min()
len()
print("python")
type("python")
reveresd()
sorted()
```

In [19]:
```
str1='python '
```

In [20]:
```
max(str1)
# Because of ASCII
```

Out[20]: 'y'

In [21]:
```
for i in str1:
    print(i,ord(i))
```

```
p 112
y 121
t 116
h 104
o 111
n 110
  32
```

In [ ]:

In [ ]:
```
max= 121======'y'
min= 32 === ''
32,104,110,111,112,116,121
'' h   n   o   p   t   y (ascending order)
```

In [22]:
```
min(str1)
```

Out[22]: ' '

In [23]:
```
min('python')
```

Out[23]: 'h'

In [24]:
```
min('NAndu')
```

Out[24]: 'A'

In [25]:
```
str1='python'
len(str1)
```

Out[25]: 6

In [28]:
```
str2='  '
len(str2)
```

Out[28]: 2

In [29]:
```
print(str2)
```

In [30]:
```
print('python')
print(' python')
```

```
python
 python
```

In [ ]:
```
max

min

len
```

sorted

- sorted means sorting the letters based on ascii number
- there two possible sortings avialble
  - ascending : small to high
  - descending: high to small

In [31]:
```
str1='python'
sorted(str1) # by default ascending order
```

Out[31]:
```
['h', 'n', 'o', 'p', 't', 'y']
```

- when you apply the shift tab
- there is some arguments will be available
- Focus on two arguments
  - iterable
  - reverse=False
- because reverse is a default argument by default acsending order is coming
- if you want to change the order then change the default parameter value

In [33]:
```
sorted(str1,reverse=True)
```

Out[33]:
```
['y', 't', 'p', 'o', 'n', 'h']
```

In [ ]:
```
sorted(iterable=str1,reverse=False)  # Fail
sorted(iterable=str1,False) # Fail
sorted(str1,False) # Fail
sorted(str1,reverse=True) # Working
```

In [ ]:
```
sorted(str1,reverse=True)
```

- we should not allowed to use iterable argument name while provide the value
- we should use the argument names while provide the value after '/'
- we should not use the argument names before '/'

        - iterable argument name is there before '/'
        - so do not use iterable name

    - any function indicates * means
    - you can use any variable after *
    - after * there are two arguments are there
        - key
        - reverse
    - you can use both
    - you can use anyone
    - you no need to use anything

In [ ]:
```
sorted('hello') # Works
sorted(iterable='hello') # Fail
```

In [34]:
```
sorted('hello')
```

Out[34]:
```
['e', 'h', 'l', 'l', 'o']
```

In [35]:
```
sorted('hello',reverse=True)
```

Out[35]:
```
['o', 'l', 'l', 'h', 'e']
```

In [ ]:
```
# ascii value it is giving the order
```

In [36]:
```
sorted('hello',key=len) # you will not understand now
```

Out[36]:
```
['h', 'e', 'l', 'l', 'o']
```

In [38]:
```
sorted('hello',reverse=True,key=len)
```

Out[38]:
```
['h', 'e', 'l', 'l', 'o']
```

In [37]:
```
def add(a=100,b=200):
    print(a+b)

add(a=1000,b=2000)
add(b=3000,a=1000)
add(b=5000)
add(a=6000)
add()
```

```
3000
4000
5100
6200
300
```

reveresd

In [ ]:
```
vscode ===== we dont have markdown
if you want to write any information
then we can use triple quotes
```

In [41]:
```
str1='python'
ans=reversed(str1)
```

In [42]:
```
type(ans)
```

Out[42]:
```
reversed
```

In [43]:
```
ans=reversed(str1)
# the output is stored a memory location
# whenever you want to see the output
# use a list or for loop
```

Out[43]:
```
<reversed at 0x16e9eb70040>
```

In [44]:
```
str1='python'
ans=reversed(str1)
for i in ans:
    print(i)
```

```
n
o
h
t
y
p
```

In [46]:
```
ans=reversed(str1)
list(ans)
```

Out[46]:
```
['n', 'o', 'h', 't', 'y', 'p']
```

concatenation

In [ ]:
```
str1='hello'
str2='python'
str1+str2
str1*str2
str1-str2
str1/str2
```

In [47]:
```
str1+str2
```

Out[47]:
```
'python  '
```

In [48]:
```
str1-str2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[48], line 1
----> 1 str1-str2

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

In [49]:
```
str1/str2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[49], line 1
----> 1 str1/str2

TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

In [50]:
```
str1*str2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[50], line 1
----> 1 str1*str2

TypeError: can't multiply sequence by non-int of type 'str'
```

In [ ]:
```
TypeError: unsupported operand type(s) for -: 'str' and 'str'
TypeError: unsupported operand type(s) for /: 'str' and 'str'
TypeError: can't multiply sequence by non-int of type 'str'
```

In [51]:
```
'python' * 2
```

Out[51]:
```
'pythonpython'
```

In [52]:
```
'python'+'python'
```

Out[52]:
```
'pythonpython'
```

index

In [53]:
```
str1='python'
#-6    -5    -4    -3     -2    -1
#'p'   'y'   't'   'h'   'o'   'n'
#0      1     2     3     4     5
```

In [55]:
```
str1[0],str1[-6]
```

Out[55]:
```
('p', 'p')
```

In [ ]:

In [ ]:

In [ ]: