In [ ]:

```
# How to create a string?
-> enclosing characters inside a single quote or double quotes
```

In [2]:

```
mystring = 'Hello'
print(mystring)


mystring = "Hello"
print(mystring)


mystring = '''Hello'''
print(mystring)
```

```
Hello
Hello
Hello
```

In [3]:

```
mystring = """Hello"""
print(mystring)
```

```
Hello
```

In [ ]:

```
# How to access characters in a string?

# indexing & range of characters using slicing
# index start from 0
```

In [4]:

```
mystring = "Hello"
```

In [5]:

```
mystring[0]
```

Out[5]:

```
'H'
```

In [6]:

```
mystring[-1]
```

Out[6]:

```
'o'
```

In [7]:

```
 mystring[15]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-7-8502467dc6db> in <module>
----> 1 mystring[15]

IndexError: string index out of range
```

In [ ]:

```
# How to change or delete a string ?
```

In [ ]:

```
# strings are immutable
```

In [8]:

```
mystring = "Hello"

mystring[4] = 's'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-eba11f523fb0> in <module>
      1 mystring = "Hello"
      2
----> 3 mystring[4] = 's'

TypeError: 'str' object does not support item assignment
```

In [9]:

```
del mystring
```

In [10]:

```
print(mystring)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-fe4c199fc922> in <module>
----> 1 print(mystring)

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

In [ ]:

```
# String Operations
```

In [ ]:

```
# Concatenation
```

In [14]:

```
s1 = "Hello"
s2 = " LetsUpgrade"
```

In [15]:

```
print(s1 + s2)
```

```
Hello LetsUpgrade
```

In [16]:

```
print(s1 * 3)
```

```
HelloHelloHello
```

In [17]:

```
print(s1 * s2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-17-173519bce8ba> in <module>
----> 1 print(s1 * s2)

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

In [ ]:

```
String Methods
```

In [19]:

```
print(dir(str))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__'
, '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '
__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__ne
w__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__'
, '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith',
'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal',
'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', '
join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replac
e', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith
', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

In [20]:

```
"Hello".upper()
```

Out[20]:

```
'HELLO'
```

In [21]:

```
"Hello".lower()
```

Out[21]:

```
'hello'
```

In [22]:

```
"This is letsupgrade session".split()
```

Out[22]:

```
['This', 'is', 'letsupgrade', 'session']
```

In [23]:

```
"hello".split()
```

Out[23]:

```
['hello']
```

In [ ]:

```

```

In [ ]:

```
# Python Objects
```

In [24]:

```
lst = [1,2,3]
```

In [25]:

```
lst.count(2)
```

Out[25]:

```
1
```

In [26]:

```
# Object
```

In [28]:

```
print(type([]))
```

```
<class 'list'>
```

In [29]:

```
print(type(()))
```

```
<class 'tuple'>
```

In [ ]:

```
# class

class is user defined objects & class is like blueprint
```

In [30]:

```
class Sample:  # capital letter
    pass


# instance of sample
x = Sample() # refrence to our new instance

print(type(x)) # function
```

```
<class '__main__.Sample'>
```

In [ ]:

```
# Attributes

syntax :
    self.attributes = something
```

In [ ]:

```
Special Method:

    __init__()
```

In [31]:

```
class Dog:
    def __init__(self, breed):
        self.breed = breed

sam = Dog(breed='Lab')
frank = Dog(breed='Huskie')
```

In [ ]:

```
# __init__ = this is special method
```

In [ ]:

```
# def __init__(self, breed) - object has been created
```

In [ ]:

```
# self.breed = breed refrence to the instance objects
```

In [32]:

```
sam.breed
```

Out[32]:

```
'Lab'
```

In [33]:

```
frank.breed
```

Out[33]:

```
'Huskie'
```

In [34]:

```
class Dog:

    # class object attribute
    species = 'mammal'

    def __init__(self, breed, name):
        self.breed = breed
        self.name = name
```

In [37]:

```
sam = Dog('Lab', 'Sam')
```

In [38]:

```
sam.name
```

Out[38]:

```
'Sam'
```

In [39]:

```
sam.species
```

Out[39]:

```
'mammal'
```

In [ ]:

```
# Creating methods in a class
```

In [44]:

```
class Circle:

    pi = 3.14

    def __init__(self, radius=1):
        self.radius = radius
        self.area = radius * radius * Circle.pi

    # Methiodof resetting Radius
    def setRadius(self, new_radius):
        self.radius = new_radius
        self.area = new_radius * new_radius * self.pi


    def getCircumference(self):
        return self.radius * self.pi * 2


c = Circle()


print("Radius is. ", c.radius)
print("Area is: ", c.area)
print("Circumstance is ", c.getCircumference())
```

```
Radius is.  1
Area is:  3.14
Circumstance is  6.28
```

In [45]:

```
c.setRadius(2)
print("Radius is. ", c.radius)
print("Area is: ", c.area)
print("Circumstance is ", c.getCircumference())
```

```
Radius is.  2
Area is:  12.56
Circumstance is  12.56
```

In [ ]:

```

```

In [49]:

```
# generator function for the cube of numbers ( power of 3)
def gencubes(n):
    for num in range(n):
        yield num ** 3
```

In [50]:

```
for x in gencubes(10):
    print(x)
```

```
0
1
8
27
64
125
216
343
512
729
```

In [51]:

```
def genfibn(n):

    a = 1
    b = 1
    for i in range(n):
        yield a
        a, b = b, a+b
```

In [52]:

```
for num in genfibn(10):
    print(num)
```

```
1
1
2
3
5
8
13
21
34
55
```

In [ ]:

```
# next() & iter()
```

In [53]:

```
def simple_gen():
    for x in range(3):
        yield x
```

In [54]:

```
g = simple_gen()
```

In [55]:

```
print(next(g))
```

```
0
```

In [56]:

```
print(next(g))
```

```
1
```

In [57]:

```
print(next(g))
```

```
2
```

In [58]:

```
print(next(g))
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-58-1dfb29d6357e> in <module>
----> 1 print(next(g))

StopIteration:
```

In [59]:

```
s = 'hello'

for i in s:
    print(i)
```

```
h
e
l
l
o
```

In [60]:

```
next(s)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-60-61c30b5fe1d5> in <module>
----> 1 next(s)

TypeError: 'str' object is not an iterator
```

In [61]:

```
s_iter = iter(s)
```

In [62]:

```
next(s_iter)
```

Out[62]:

```
'h'
```

In [63]:

```
next(s_iter)
```

Out[63]:

```
'e'
```

In [64]:

```
next(s_iter)
```

Out[64]:

```
'l'
```

In [65]:

```
next(s_iter)
```

Out[65]:

```
'l'
```

In [ ]:

```

```

In [66]:

```
next(s_iter)
```

Out[66]:

```
'o'
```

In [ ]: