In [68]:

```
!python --version
```

```
Python 3.7.3
```

In [126]:

```
# Example on int;
x = 8
print(x, type(x))
```

```
8 <class 'int'>
```

In [127]:

```
print(x + 1) # Addition
print(x - 1) # Subtraction
print(x * 2) # Multiplication
print(x ** 2) # Exponentiation
```

```
9
7
16
64
```

In [128]:

```
print(x)
x *= 2
print(x)
```

```
8
16
```

In [129]:

```
# Example on float;
y = 5.5
print(type(y))
print(y, y + 1, y * 2, y ** 2)
```

```
<class 'float'>
5.5 6.5 11.0 30.25
```

In [130]:

```
t, f = True, False
print(type(t))
```

```
<class 'bool'>
```

In [131]:

```
print(t and f) # Logical AND;
print(t or f) # Logical OR;
print(not t) # Logical NOT;
print(t != f) # Logical XOR;
```

```
False
True
False
True
```

In [75]:

```
h = 'hello' # String literals can use single quotes
w = "world" # or double quotes; it does not matter
print(h, "Len=", len(h))
```

```
hello Len= 5
```

In [132]:

```
hw = h +' '+ w # String concatenation
print(hw)
```

```
hello world
```

In [133]:

```
print(h+w)
```

```
helloworld
```

In [134]:

```
hw77 = '{} {} {}'.format(h, w, 77) # string formatting
print(hw77)
```

```
hello world 77
```

In [135]:

```
s = "hello"
print(s.capitalize()) # Capitalize starting char
print(s.upper()) # Convert a string to uppercase; prints "HELLO"
print(s.rjust(10)) # Right-justify a string, padding with spaces
print(s.center(10)) # Center a string, padding with spaces
print(s.replace('l','(ell)'))#Replace all instances of 1 substring with another
print(' world '.strip()) # Strip leading and trailing whitespace
```

```
Hello
HELLO
     hello
  hello
he(ell)(ell)o
world
```

In [136]:

```
xs = [5, 3, 7] # Create a list
print(type(xs))
print(xs)
print(xs[1])
print(xs[-1])
```

```
<class 'list'>
[5, 3, 7]
3
7
```

In [137]:

```
xs[2] = 'bat' # Lists can contain elements of different types
print(xs)
```

```
[5, 3, 'bat']
```

In [138]:

```
xs.append('ball') # Add a new element to the end of the list
print(xs)
```

```
[5, 3, 'bat', 'ball']
```

In [139]:

```
x = xs.pop() # Remove and return the last element of the list
print(x,'\t', xs)
```

```
ball     [5, 3, 'bat']
```

In [140]:

```
animals = ['cat', 'dog', 'monkey']
for ani in animals:
    print(ani)
```

```
cat
dog
monkey
```

In [141]:

```
animals = ['cat', 'dog', 'monkey']
for idx, ani in enumerate(animals):
    print('{}: {}'.format(idx +1, ani))
```

```
1: cat
2: dog
3: monkey
```

In [142]:

```
print(range(5))
```

```
range(0, 5)
```

In [143]:

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

```
0
1
2
3
4
```

In [144]:

```
list(range(5))
```

Out[144]:

```
[0, 1, 2, 3, 4]
```

In [145]:

```
list(range(1,6))
```

Out[145]:

```
[1, 2, 3, 4, 5]
```

In [146]:

```
list(range(2,11,2))
```

Out[146]:

```
[2, 4, 6, 8, 10]
```

In [147]:

```
nums = list(range(5)) # range is a built-in func. that creates a list of ints
print(nums)
print(nums[2:4]) # Get a slice from index 2 to 4(exclusive); prints "[2, 3]"
print(nums[2:]) # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print(nums[:2]) #Get a slice from the start to index 2(exclusive);prints"[0,1]"
print(nums[:]) # Get a slice of the whole list; prints ["0, 1, 2, 3, 4]"
print(nums[:-1]) # Slice indices can be negative; prints ["0, 1, 2, 3]"
print(nums[1::2])
nums[2:4] = [8, 9] # Assign a new sublist to a slice of list
print(nums) # Prints "[0, 1, 8, 9, 4]"
```

```
[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
[0, 1]
[0, 1, 2, 3, 4]
[0, 1, 2, 3]
[1, 3]
[0, 1, 8, 9, 4]
```

In [148]:

```
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x**2)
    print(squares)
```

```
[0]
[0, 1]
[0, 1, 4]
[0, 1, 4, 9]
[0, 1, 4, 9, 16]
```

In [149]:

```
nums = [0, 1, 2, 3, 4]
squares = [x**2 for x in nums]
print(squares)
```

```
[0, 1, 4, 9, 16]
```

In [150]:

```
# Only list of even squared numbers
nums = [0, 1, 2, 3, 4]
even_squares = [x**2 for x in nums if x%2 == 0]
print(even_squares)
```

```
[0, 4, 16]
```

In [151]:

```
str='human'
letters = []
for ch in str:
    letters.append(ch)
print(letters)
```

```
['h', 'u', 'm', 'a', 'n']
```

In [152]:

```
print([ch for ch in 'human'])
```

```
['h', 'u', 'm', 'a', 'n']
```

In [153]:

```
d = {'cat': 'cute', 'dog': 'furry'} # Create a new dictionary with some data
print(d['cat']) # Get an entry from a dictionary; prints "cute"
print('cat' in d) # Check if a dictionary has a given key; prints "True"
```

```
cute
True
```

In [154]:

```
d = {'cat': 'cute', 'dog': 'furry'}
d['cat']= 'flexible' # Updates the value, if key is present;
print(d)
```

```
{'cat': 'flexible', 'dog': 'furry'}
```

In [155]:

```
d['fish'] = 'wet' # Set an entry in a dictionary, if key is not present;
print(d)
```

```
{'cat': 'flexible', 'dog': 'furry', 'fish': 'wet'}
```

In [156]:

```
d.keys()
```

Out[156]:

```
dict_keys(['cat', 'dog', 'fish'])
```

In [157]:

```
d.values()
```

Out[157]:

```
dict_values(['flexible', 'furry', 'wet'])
```

In [158]:

```
d.items()
```

Out[158]:

```
dict_items([('cat', 'flexible'), ('dog', 'furry'), ('fish', 'wet')])
```

In [159]:

```
print(d['monkey']) # KeyError: 'monkey' not a key of d
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-159-72598f4378e1> in <module>
----> 1 print(d['monkey']) # KeyError: 'monkey' not a key of d

KeyError: 'monkey'
```

In [160]:

```
print(d)
print(d.get('monkey')) # Get an element with a default;
print(d.get('monkey', 'NotAvai')) # Get an element with a default; prints "NA"
print(d.get('fish', 'NA')) # Get an element with a default; prints "wet"
```

```
{'cat': 'flexible', 'dog': 'furry', 'fish': 'wet'}
None
NotAvai
wet
```

In [161]:

```
del d['fish'] # Remove an element from a dictionary
print(d.get('fish', 'NA')) # "fish" is no longer a key; prints "N/A"
```

```
NA
```

In [162]:

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A {} has {} legs'.format(animal, legs))
```

```
A person has 2 legs
A cat has 4 legs
A spider has 8 legs
```

In [163]:

```
nums = [0, 1, 2, 3, 4, 5]
num_to_square = {x: x ** 2 for x in nums}
print(num_to_square)
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [164]:

```
nums = [0, 1, 2, 3, 4, 5]
even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}
print(even_num_to_square)
```

```
{0: 0, 2: 4, 4: 16}
```

In [165]:

```
animals = {'cat', 'dog'}
print('cat' in animals) # Check if an element is in a set; prints "True"
print('fish' in animals) # prints "False"
```

```
True
False
```

In [166]:

```
animals.add('fish') # Add an element to a set
print('fish' in animals)
print(animals)
print(len(animals)) # Number of elements in a set;
```

```
True
{'dog', 'fish', 'cat'}
3
```

In [167]:

```
print(len(animals))
animals.add('cat') # Adding an element that is already in the set does nothing
animals.add('dog')
print(animals)
print(len(animals))
animals.remove('cat') # Remove an element from a set
print(len(animals))
```

```
3
{'dog', 'fish', 'cat'}
3
2
```

In [168]:

```
animals = {'cat', 'dog', 'fish', 'monkey', 'hen'}
for idx, animal in enumerate(animals):
    print('{}: {}'.format(idx + 1, animal))
```

```
1: cat
2: monkey
3: fish
4: dog
5: hen
```

In [169]:

```
from math import sqrt
s = {int(sqrt(x)) for x in range(20)}
print(s)
print(type(s))
```

```
{0, 1, 2, 3, 4}
<class 'set'>
```

In [170]:

```
l = [int(sqrt(x)) for x in range(20)]
print(l)
print(type(l))
```

```
[0, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4]
<class 'list'>
```

In [171]:

```
t = (1, 2, 3) # Create a tuple
print(type(t))
print(t)
```

```
<class 'tuple'>
(1, 2, 3)
```

In [172]:

```
# Tuple in Dict
d = {(x, x+1): x+11 for x in range(5)} # Create a dictionary with tuple keys
print(d)
print(type(d))
```

```
{(0, 1): 11, (1, 2): 12, (2, 3): 13, (3, 4): 14, (4, 5): 15}
<class 'dict'>
```

In [173]:

```
t=(2, 3)
print(d[t])
print(type(t))
```

```
13
<class 'tuple'>
```

In [174]:

```
# Tuple in Set
s={(1,2), 3, 4, (5,6), 7, 8}
print(s)
print(type(s))
```

```
{(1, 2), 3, 4, 7, (5, 6), 8}
<class 'set'>
```

In [175]:

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'
for x in [-1, 0, 1]:
    print(sign(x))
```

```
negative
zero
positive
```

In [176]:

```
def add(n1, n2):
    print(n1," ",n2)
    print("Sum =",n1+n2)
```

In [177]:

```
add() # Error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-177-75d5be5ef31a> in <module>
----> 1 add() # Error

TypeError: add() missing 2 required positional arguments: 'n1' and 'n2'
```

In [121]:

```
add(10) #10
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-121-271ea1858236> in <module>
----> 1 add(10) #10

TypeError: add() missing 1 required positional argument: 'n2'
```

In [122]:

```
add(10,20) #30
```

```
10   20
Sum = 30
```

In [180]:

```
class Faculty:
    def __init__(self, loc="Hyd"): # Constructor
        self.location=loc # Create an Instance Variable
    ins_name="MRITS" # Static Varialbe
    def assign(self, id, na='N/A'): # Instance Methods1
        self.idno=id
        self.name=na # Instance Variables
    def display(self): # Instance Methods2
        print("IDNO:", self.idno)
        print("NAME:", self.name)
        print("INSTITUTION:", Faculty.ins_name)
        print("LOCATION:", self.location)
```

In [181]:

```
f1=Faculty() # Object created
f1.assign(101,"Raj") # Calling method
f1.display()
```

```
IDNO: 101
NAME: Raj
INSTITUTION: MRITS
LOCATION: Hyd
```

In [182]:

```
f2=Faculty("Blr")
f2.assign(102)
f2.display()
```

```
IDNO: 102
NAME: N/A
INSTITUTION: MRITS
LOCATION: Blr
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: