In [1]:

```
print("Good Morning")
```

```
Good Morning
```

In [135]:

```
greeting = 'Hello'
wishes = 'Good Morning'
hi = greeting + ' '+ wishes # Concat or combining of 2 variables
hi
```

Out[135]:

```
'Hello Good Morning'
```

In [136]:

```
len(hi) # Length of a string
```

Out[136]:

```
18
```

In [137]:

```
type(hi) # Type variable
```

Out[137]:

```
str
```

In [138]:

```
replace_val = hi.replace('Hello', 'Hola') # Replacing a string into another string
replace_val
```

Out[138]:

```
'Hola Good Morning'
```

In [14]:

```
c = '{} {}'.format(greeting,wishes) # Joining of 2 string
c
d = f'{greeting} {wishes}! Welcome!!' # Joining of 2 string using f string method availabe in
python 3.6 and above
d
```

Out[14]:

```
'Hello Good Morning! Welcome!!'
```

Lists [ ], Tuples( ) and Sets { }

In [53]:

```
course = ['DataScience', 'Python', 'R', 'SQL'] # List which is denoted with []
print(course)
len(course)
course[2:] # Indexing
```

```
['DataScience', 'Python', 'R', 'SQL']
```

Out[53]:

```
['R', 'SQL']
```

In [57]:

```
course.append('DBMS') # Insert @ the end of the list
course
```

Out[57]:

```
['DataScience', 'Python', 'R', 'SQL', 'DBMS', 'DBMS', 'DBMS']
```

In [59]:

```
course.insert(0,'Flask') # Insert @ the beginning of the list
course
```

Out[59]:

```
['Flask', 'Flask', 'DataScience', 'Python', 'R', 'SQL', 'DBMS', 'DBMS', 'DBMS']
```

In [55]:

```
old_course = ['Python','Pycharm']
new_course = ['Django', 'MongoDB']
new_course.extend(old_course) # Combibe both the lists - pass variable withot '' in extend
new_course
```

Out[55]:

```
['Django', 'MongoDB', 'Python', 'Pycharm']
```

In [60]:

```
old_course.remove('Pycharm') # Remove the item which is passed
old_course
```

Out[60]:

```
['Python']
```

In [98]:

```
sub = ['Eng', 'Eco', 'Stat', 'Phy','Phy'] # list
sub
```

Out[98]:

```
['Eng', 'Eco', 'Stat', 'Phy', 'Phy']
```

In [83]:

```
sub.pop() # Remove last item in the list - in this case it is Phy which is removed
sub
```

Out[83]:

```
['Eng', 'Eco', 'Stat']
```

In [87]:

```
sub.reverse() # Print list in reverse order
sub
```

Out[87]:

```
['Stat', 'Eco', 'Eng']
```

In [89]:

```
for index,subject in enumerate(sub, start=1): # To print index and subject together, o/p of index will be 012, instead if we
                                              #like to index start from 1 instead of 0 in for colu
pass argument start=1
    print(index,subject)
```

```
1 Stat
2 Eco
3 Eng
```

In [90]:

```
sub_str = ','.join(sub) # Turning a list into a string with ,
sub_str
```

Out[90]:

```
'Stat,Eco,Eng'
```

Tuples are immutable example is mentioned below

Tuples () are similar to list but the only differece is former cannot be modified where as latter can be modified

In [96]:

```
tuple_1 = ('Eng', 'Phy') # Tuple is represented as ()
tuple_2 = tuple_1
print(tuple_1)
(tuple_2)
tuple_1[0] = ('Kannada')
print(tuple_1) # Output will be tuple object does not support item assignment
```

```
('Eng', 'Phy')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-96-110de0d0e9dd> in <module>
      3 print(tuple_1)
      4 (tuple_2)
----> 5 tuple_1[0] = ('Kannada')
      6 print(tuple_1) # Output will be tuple object does not support item assignment

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

Sets { } - Union, Intersection, Difference

Sets - are values that are unordered and has no duplicate

Uses : 1. Remove duplicate value ; 2. Find out common elements b/n 2 sets ; 3. See whether a value is part of set or not

In [100]:

```
programmes = {'C', 'C++', 'Java', 'Python', 'Java'} # List with a duplicate value of Java. Sets are denoted as '{}'
programmes # Note in output duplicate value is removed and Java is printed only once istead of twice.(1st use)
```

Out[100]:

```
{'C', 'C++', 'Java', 'Python'}
```

In [102]:

```
print('C++' in programmes) # o/p-True Use no.3 (True or False)
print('MongoDB' in programmes) # - o/p False
```

```
True
False
```

In [105]:

```
new_programmes = {'Postgresql', 'MongoDB', 'SQL', 'Python'}
new_programmes
```

Out[105]:

```
{'MongoDB', 'Postgresql', 'Python', 'SQL'}
```

In [106]:

```
print (programmes.intersection(new_programmes)) # Print out common element in both the sets
```

```
{'Python'}
```

In [109]:

```
print(programmes.difference(new_programmes)) # Print difference element from programme comparing n
ew_programmes
```

```
{'C', 'C++', 'Java'}
```

In [110]:

```
print(programmes.union(new_programmes)) # Print all the elements in both the sets its like joining
of 2 sets without duplicates
```

```
{'C', 'SQL', 'Java', 'MongoDB', 'Postgresql', 'C++', 'Python'}
```

Dictionaries

Keys and Values: Keys are unique identifiers and values represent the data or information againt key

Example: Keys - name, age, course ; Values - john, 26, python, java

In dictionaries we can pass arguments like strings, list and intergers

In [111]:

```
student = {'name':'John', 'age':26, 'course':['Java', 'Python']} # name - String ; age - Integer ;
course - List
student
```

Out[111]:

```
{'name': 'John', 'age': 26, 'course': ['Java', 'Python']}
```

In [114]:

```
print(student['age']) # - Index method --> To get specific value from the dictionary
# OR
print(student.get('age')) # get method --> To get specific value from the dictionary. O/P remain s
ame in both cases.
```

```
26
26
```

In [115]:

```
print(student.get('phone'), 'Not Found!') # Print not found if item is not present. Default is non
e
```

```
None Not Found!
```

In [118]:

```
student.update({'name':'Peter', 'course':['MongoDB', 'Django']}) # To update student details
student   # original list of student has been changed
```

Out[118]:

```
{'name': 'Peter', 'age': 26, 'course': ['MongoDB', 'Django']}
```

In [119]:

```
del student['age']  # to delete certain info of a student
student  # age information is deleted, check output values
```

Out[119]:

```
{'name': 'Peter', 'course': ['MongoDB', 'Django']}
```

In [120]:

```
print(student.keys()) # print the key values like name, course
```

```
dict_keys(['name', 'course'])
```

In [121]:

```
print(student.values()) # print values of key
```

```
dict_values(['Peter', ['MongoDB', 'Django']])
```

In [126]:

```
for key,value in student.items(): # To see both key and values representing against each other
    print(key,value)
```

```
name Peter
course ['MongoDB', 'Django']
```

Conditionals : if, elif, else statement

In [127]:

```
language = 'python' # if you change language- python to java withot changing if condition o/p wil
be False
if language == 'python': # Using if condition
    print("True")
else:
    print("False")
```

```
True
```

In [128]:

```
language = 'C'
if language == 'C++':
    print("Language is C++")
```

```
elif language == 'C': # el if statment
    print('Language is C')
else:
    print('No results found!')
```

```
Language is C
```

Boolean : AND, OR, NOT

In [139]:

```
user = 'admin'
logged_in = True # Change it to False to see results

if user == 'admin' and logged_in:
    print('Welcome to admin Page')
else:
    print('Invalid Credentials')

if user == 'user' or logged_in: # here only logged_in is satisfied and printed true value. Require only 1 condtion get true
    print('Welcome to admin Page')
else:
    print('Invalid Credentials')
```

```
Welcome to admin Page
Welcome to admin Page
```

Loops and Iterations : Break and Continue

In [144]:

```
numbers = [1,2,3,4,5,6,4,7,8]
for num in numbers:
    print(num)
```

```
1
2
3
4
5
6
4
7
8
```

In [145]:

```
for num in numbers:
    if num == 3:
        print('Value Found')
        break   # Once 3 is found it will break there and won't proceed further
    print(num)
```

```
1
2
Value Found
```

In [146]:

```
for num in numbers:
    if num == 4:
        print('Value Found')
        continue # value 4 is found ! print Value found ! and continue iteration
    print(num)
```

```
1
2
3
Value Found
5
6
Value Found
7
8
```

Loop within a loop

In [148]:

```
nums = [1,2,3]
for num in nums:
    for letter in 'abc':
        print(num,letter)
```

```
1 a
1 b
1 c
2 a
2 b
2 c
3 a
3 b
3 c
```

In [150]:

```
for i in range(5):   # Print values of i within the range mentioned
    print(i)
```

```
0
1
2
3
4
```

In [151]:

```
for i in range(1,6): # 1 - to srart with 1 instead of 0 and 6 - to end at value 5 (since index 0 i
s 1 and index 6 is 5)
    print(i)
```

```
1
2
3
4
5
```

Slicing of strings

In [2]:

```
url = 'http://google.com'
url
```

Out[2]:

```
'http://google.com'
```

In [9]:

```
print( url[7:]) # if we want to print only google.com then slice string by index values. So index v
alue 7 is g till end
```

```
# OR
print (url[-10:])
```

```
google.com
google.com
```

In [6]:

```
url[-3:] # if we want to print only .com
```

Out[6]:

```
'com'
```

Comprehensions

In [14]:

```
nums = [1,2,3,4,5,6]

my_list = [] # if we want to print numbers in nums list into my_list

for n in nums:
    my_list.append(n)
print(my_list)

#OR

my_list = [n for n in nums]
print(my_list)
```

```
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6]
```

In [21]:

```
number = [1,2,3,4,5,6,7,8] # printing only even number in my_list

my_list = []

for n in number:
    if n % 2 == 0:
        my_list.append(n)
print(my_list)

#OR

my_list = [n for n in number if n%2==0]
print(my_list)
```

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

Dictionary Comprehension

In [38]:

```
Heros = ['Sharukh', 'Salman', 'Akshay']
Movies = ['HNY', 'Dabang', 'MOM']

mapp = zip(Heros,Movies)
mapped = set(mapp)

mapped # Print output in dictionary like { (hero,movie), (hero,movie)}

result = list(zip(*mapped)) # Print in list format [(Hero,hero), (movie,movie)]
result
```

Out[38]:

```
[('Akshay', 'Sharukh', 'Salman'), ('MOM', 'HNY', 'Dabang')]
```

In [45]:

```
# Generator Expression

num = [1,2,3,4]

def gen_func(x):
    for n in x:
        yield n*n

my_gen = gen_func(num)
for i in my_gen:
    print(i)
```

```
1
4
9
16
```

String Formatting - Advanced operations for dicts, lists, numbers and dates

In [57]:

```
person = {'name': 'Peter', 'age': 26} # Dictionary
sentence = 'My name is {} and Iam {} years old'.format(person['name'], person['age'])
sentence_1 = 'My name is {[name]}, and Iam {[age]} years old'.format(person,person) # Using index i
nside{}

print(sentence)
sentence_1
```

```
My name is Peter and Iam 26 years old
```

Out[57]:

```
'My name is Peter, and Iam 26 years old'
```

In [68]:

```
# Using Attributes

class Employee():

    def __init__(self,name,age):
        self.name = name
        self.age = age

Emp1 = Employee('John', 33)

emp_details = 'My name is {0.name} and Iam {0.age} years old'.format(Emp1) # we add 0.age/0.name
bcz - it is self.age/self.name

print (emp_details)

# Using Keyword Argument

emp_details1 = 'My name is {name} and Iam {age} years old'.format(name='John', age=33)
print(emp_details1)

person = {'name':'John', 'age':33} # if values are in dictionaly instead of using class use below m
ethod

person_name = 'My name is {name} and Iam {age} years old'.format(**person) # * - Key , ** - Values

print(person_name)
```

```
My name is John and Iam 33 years old
My name is John and Iam 33 years old
My name is John and Iam 33 years old
```

In [96]:

```
# Number formatting

for i in range(1,11):
    sentence = "The values are {:}".format(i)
    print(i)
print(sentence)  # To count the totla number of values
```

```
1
2
3
4
5
6
7
8
9
10
The values are 10
```

In [139]:

```
import datetime  # importing datetime

date = datetime.datetime(2019,10,4,12,46,40) # Printing date and time manually

print(date)

tday = datetime.date.today() # print today's date automatically

print(tday)

#Usually system print date in YYYY-MM-DD format. To change the format use below set of code

tday_new_format = '{:%B %d, %Y}'.format(tday) # %B - Month in full %d - date %Y - Year

print(tday_new_format)

# say if we want to print 04 October 2019 fell on Friday and is the 277 day of the year

tday_sentence = '{0:%d %B, %Y} fell on {0:%A} and is the {0:%j} day of the year'.format(tday) # %
A - Day and %j - number of day

print(tday_sentence)
```

```
2019-10-04 12:46:40
2019-10-04
October 04, 2019
04 October, 2019 fell on Friday and is the 277 day of the year
```

In [116]:

```
print(tday.year) # to print only year from the date

print(tday.month) # to print only month
```

```
2019
10
```

In [120]:

```
# There are 2 types of weekdays that system can understand 1. weekdays --> 0 - mon & 6 - Sun 2. is
oweekdays --> 1-mon & 7 - Sun

print(tday.weekday()) # (system understand 0 - Mon and 6 - Sun) In O/P 4 - means - Friday
```

```
print(tday.isoweekday()) # (system understand 1 - Mon and 7 - Sun) - O/P 5  -means - Friday
```

```
4
5
```

In [121]:

```
# timedelta function

timedelta = datetime.timedelta(days=7)
print(timedelta+tday) # today's date + 7days
```

```
2019-10-11
```

In [122]:

```
dt_today = datetime.date.today() # Print only date
dt_now = datetime.datetime.now() # Print date as well as time
dt_utcnow = datetime.datetime.utcnow() # print date and time in UTC

print(dt_today)
print(dt_now)
print(dt_utcnow)
```

```
2019-10-04
2019-10-04 13:05:01.992508
2019-10-04 07:35:01.992508
```

In [130]:

```
import pytz # timezone library (pytz = python timezone)

dt_utcnow = datetime.datetime.now(tz=pytz.utc)

print(dt_utcnow) # time zone @ GMT - Greenwhich meridian time

dt_india = dt_utcnow.astimezone(pytz.timezone('Asia/Kolkata'))

print(dt_india) # Time zone of India - Asia/Kolkata
```

```
2019-10-04 07:43:36.766548+00:00
2019-10-04 13:13:36.766548+05:30
```

In [133]:

```
#for tz in pytz.all_timezones: # to get all timezones available in library
    #print(tz)
```

File object - Reading and Writing files

In [173]:

```
with open('C:/Users/Syed Afzal/Desktop/test.txt', 'r') as f: # open file in read mode
    f_contents = f.read()  # read file
    print(f_contents) # print the content - entire content in the file wil be pritned
```

```
Syed Afzal
Welcome
How are you!!
```

In [187]:

```
with open('C:/Users/Syed Afzal/Desktop/test.txt', 'r') as f: # open file in read mode

    #f_contents = f.read(10) # 10 indicates print only 10 charecter from the file
```

```
    #print(f_contents)

    f_rline = f.readline() # read individual line from the list at a time

    print(f_rline)

    f_rlines = f.readlines() # read and print all the lines in the list

    print(f_rlines)
```

```
Syed Afzal

['Welcome\n', 'How are you!!']
```

In [188]:

```
# in readlines comman we notice the values are printing in different manner not in the same order
as in file. So to print
# the lines in file as it is we can write

with open('C:/Users/Syed Afzal/Desktop/test.txt') as f:
    for line in f:
        print(line, end="")
```

```
Syed Afzal
Welcome
How are you!!
```

In [6]:

```
with open('C:/Users/Syed Afzal/Desktop/test.txt', 'a') as f: # if you want to write in the
existing file use 'a' to add new line
    f.write("Hello! How are you")                            # new data will be printed with existi
g data
```

In [8]:

```
with open('C:/Users/Syed Afzal/Desktop/test.txt', 'w') as f: # if you use 'w' mode all the
existing data wil replaced with new
    f.write('Hello Syed Afzal \nHow are you!')    # existing data will be replaced with new data
which is writtne
```

In [9]:

```
# if we want to copy entire data from existing file into a new file then use command below:

with open('C:/Users/Syed Afzal/Desktop/test.txt', 'r') as rf: # original file - from where data is
copied should be in read mode
    with open('C:/Users/Syed Afzal/Desktop/test_copy.txt', 'w') as wf: # file into which data is co
pied should be in write mode
        for line in rf:
            wf.write(line)
```

Generating random numbers and data using random module

In [12]:

```
import random  # importing random library
```

In [13]:

```
values = random.random()  # generate random values greater than zero but less than 1
values
```

Out[13]:

```
0.9354073673624956
```

In [20]:

```
val = random.uniform(1,10) # generate random values greater than 1 but less than 10 and all re flo
ating point values
val
```

Out[20]:

```
2.8432364917949497
```

In [28]:

```
values = random.randint(1,6) # generata random integers including both the values 1 and 6
values
```

Out[28]:

```
5
```

In [33]:

```
greetings = ['Hello', 'Hi', 'Hola', 'Namaste', 'Salam']

wishes = random.choice(greetings) # Select random values from variable greetings

print(wishes+','+ 'Syed!')
```

```
Hi,Syed!
```

In [39]:

```
colors = ['Red', 'Green', 'Blue']

results = random.choices(colors, k=10) # Select random values and print 10 times in a single list
*K denote how many times

print(results)                          # CHOICES used to get multiple items

results_weight = random.choices(colors, weights=[50,40,10], k=5) # weights denote chances of repeat
ing a item

print(results_weight)
```

```
['Blue', 'Green', 'Red', 'Red', 'Blue', 'Green', 'Blue', 'Green', 'Red', 'Blue']
['Green', 'Red', 'Green', 'Green', 'Red']
```

In [65]:

```
# let us try a exmaple where in create address of random person using random module

person = {'first_name': ['john', 'peter', 'alex'], 'last_name':['doe', 'kumar', 'well'],
          'street_name':['Xyz', 'ABC', 'MNO'], 'fake_cities':['NY', 'DEL', 'HYD'], 'states':['Kar',
'Tel', 'TN']}
person
```

Out[65]:

```
{'first_name': ['john', 'peter', 'alex'],
 'last_name': ['doe', 'kumar', 'well'],
 'street_name': ['Xyz', 'ABC', 'MNO'],
 'fake_cities': ['NY', 'DEL', 'HYD'],
 'states': ['Kar', 'Tel', 'TN']}
```

In [149]:

```
for num in range (10):
    first = random.choice(person['first_name'])
```

```
    last = random.choice(person['last_name'])
    phone = f'{random.randint(100,999)}-555-{random.randint(1000,9999)}'
    street = random.choice(person['street_name'])
    city = random.choice(person['fake_cities'])
    state = random.choice(person['states'])
    zipcode = random.randint(100000,999999)
    address = f'{street} {city}\n{state}\n{zipcode}'
    email = f'{first}.{last}'+'@mail.com'
print(f'{first.upper()} {last.upper()}\n{phone}\n{address}\n{email}')
```

```
PETER KUMAR
497-555-4850
ABC DEL
TN
749845
peter.kumar@mail.com
```

Reading, Parsing and Writing CSV files using CSV Module

In [82]:

```
import csv # Importing CSV module
```

In [111]:

```
with open('C:/Users/Syed Afzal/Downloads/MOCK_DATA.csv') as csv_file:  # importing csv file
    csv_reader = csv.reader(csv_file)
    for line in csv_reader:
        print(line)
```

```
['id', 'first_name', 'last_name', 'email', 'gender']
['1', 'Madelena', 'Hammelberg', 'mhammelberg0@nydailynews.com', 'Female']
['2', 'Doralia', 'Brusin', 'dbrusin1@wired.com', 'Female']
['3', 'Deane', 'Crowch', 'dcrowch2@boston.com', 'Female']
['4', 'Hewe', 'Fruin', 'hfruin3@zdnet.com', 'Male']
['5', 'Caryn', 'Bailes', 'cbailes4@goo.ne.jp', 'Female']
['6', 'Christyna', 'Gabey', 'cgabey5@posterous.com', 'Female']
['7', 'Lynnea', 'Dodell', 'ldodell6@scientificamerican.com', 'Female']
['8', 'Koren', 'Moyler', 'kmoyler7@google.de', 'Female']
['9', 'Ivonne', 'Siddons', 'isiddons8@pcworld.com', 'Female']
['10', 'Kettie', 'Cleef', 'kcleef9@dell.com', 'Female']
['11', 'Babita', 'Bugg', 'bbugga@pcworld.com', 'Female']
['12', 'Idaline', 'Algore', 'ialgoreb@domainmarket.com', 'Female']
['13', 'Krystyna', 'Linge', 'klingec@seesaa.net', 'Female']
['14', 'Michele', 'Whorlow', 'mwhorlowd@state.tx.us', 'Male']
['15', 'Phillipp', 'Chorley', 'pchorleye@jigsy.com', 'Male']
['16', 'Arabella', 'Parzis', 'aparzisf@nationalgeographic.com', 'Female']
['17', 'Rodolfo', 'Cauldwell', 'rcauldwellg@irs.gov', 'Male']
['18', 'Sampson', 'Meneyer', 'smeneyerh@salon.com', 'Male']
['19', 'Rois', 'Padginton', 'rpadgintoni@liveinternet.ru', 'Female']
['20', 'Nealson', 'Vasler', 'nvaslerj@newyorker.com', 'Male']
```

In [148]:

```
with open('C:/Users/Syed Afzal/Downloads/MOCK_DATA.csv', 'r') as csv_file:
    csv_reader = csv.reader(csv_file)
    with open('C:/Users/Syed Afzal/Downloads/syed.csv', 'w') as syed_file:
        csv_writer = csv.writer(syed_file, delimiter = '-')
        for line in csv_reader:
            csv_writer.writerow(line)
```

In [112]:

```
with open ('C:/Users/Syed Afzal/Downloads/MOCK_DATA.csv','r') as csv_file:
    csv_reader = csv.DictReader(csv_file)  # Split data into first name, last name seperately as me
ntioned in list
    for line in csv_reader:
        print(line)
```

```
OrderedDict([('id', '1'), ('first_name', 'Madelena'), ('last_name', 'Hammelberg'), ('email'
```

```
OrderedDict([('id', '1'), ('first_name', 'Madelena'), ('last_name', 'Hammelberg'), ('email',
'mhammelberg0@nydailynews.com'), ('gender', 'Female')])
OrderedDict([('id', '2'), ('first_name', 'Doralia'), ('last_name', 'Brusin'), ('email',
'dbrusin1@wired.com'), ('gender', 'Female')])
OrderedDict([('id', '3'), ('first_name', 'Deane'), ('last_name', 'Crowch'), ('email',
'dcrowch2@boston.com'), ('gender', 'Female')])
OrderedDict([('id', '4'), ('first_name', 'Hewe'), ('last_name', 'Fruin'), ('email',
'hfruin3@zdnet.com'), ('gender', 'Male')])
OrderedDict([('id', '5'), ('first_name', 'Caryn'), ('last_name', 'Bailes'), ('email',
'cbailes4@goo.ne.jp'), ('gender', 'Female')])
OrderedDict([('id', '6'), ('first_name', 'Christyna'), ('last_name', 'Gabey'), ('email',
'cgabey5@posterous.com'), ('gender', 'Female')])
OrderedDict([('id', '7'), ('first_name', 'Lynnea'), ('last_name', 'Dodell'), ('email',
'ldodell6@scientificamerican.com'), ('gender', 'Female')])
OrderedDict([('id', '8'), ('first_name', 'Koren'), ('last_name', 'Moyler'), ('email',
'kmoyler7@google.de'), ('gender', 'Female')])
OrderedDict([('id', '9'), ('first_name', 'Ivonne'), ('last_name', 'Siddons'), ('email',
'isiddons8@pcworld.com'), ('gender', 'Female')])
OrderedDict([('id', '10'), ('first_name', 'Kettie'), ('last_name', 'Cleef'), ('email',
'kcleef9@dell.com'), ('gender', 'Female')])
OrderedDict([('id', '11'), ('first_name', 'Babita'), ('last_name', 'Bugg'), ('email',
'bbugga@pcworld.com'), ('gender', 'Female')])
OrderedDict([('id', '12'), ('first_name', 'Idaline'), ('last_name', 'Algore'), ('email',
'ialgoreb@domainmarket.com'), ('gender', 'Female')])
OrderedDict([('id', '13'), ('first_name', 'Krystyna'), ('last_name', 'Linge'), ('email',
'klingec@seesaa.net'), ('gender', 'Female')])
OrderedDict([('id', '14'), ('first_name', 'Michele'), ('last_name', 'Whorlow'), ('email',
'mwhorlowd@state.tx.us'), ('gender', 'Male')])
OrderedDict([('id', '15'), ('first_name', 'Phillipp'), ('last_name', 'Chorley'), ('email',
'pchorleye@jigsy.com'), ('gender', 'Male')])
OrderedDict([('id', '16'), ('first_name', 'Arabella'), ('last_name', 'Parzis'), ('email',
'aparzisf@nationalgeographic.com'), ('gender', 'Female')])
OrderedDict([('id', '17'), ('first_name', 'Rodolfo'), ('last_name', 'Cauldwell'), ('email',
'rcauldwellg@irs.gov'), ('gender', 'Male')])
OrderedDict([('id', '18'), ('first_name', 'Sampson'), ('last_name', 'Meneyer'), ('email',
'smeneyerh@salon.com'), ('gender', 'Male')])
OrderedDict([('id', '19'), ('first_name', 'Rois'), ('last_name', 'Padginton'), ('email',
'rpadgintoni@liveinternet.ru'), ('gender', 'Female')])
OrderedDict([('id', '20'), ('first_name', 'Nealson'), ('last_name', 'Vasler'), ('email',
'nvaslerj@newyorker.com'), ('gender', 'Male')])
```

In [142]:

```
with open('C:/Users/Syed Afzal/Downloads/MOCK_DATA.csv', 'r') as csv_file:
    csv_reader = csv.DictReader(csv_file)
    with open('C:/Users/Syed Afzal/Downloads/new_data.csv',  'w') as new_file:
        fieldname = ['id','first_name', 'last_name', 'email', 'gender']
        csv_writer = csv.DictWriter(new_file, fieldnames=fieldname, delimiter='-')
        csv_writer.writeheader()
        for line in csv_reader:
            csv_writer.writerow(line)
```

In [145]:

```
with open('C:/Users/Syed Afzal/Downloads/new_data.csv',  'r') as csv_file:
    csv_reader = csv.reader(csv_file)
    for line in csv_reader:
        print(line)
```

```
['id-first_name-last_name-email-gender']
[]
['1-Madelena-Hammelberg-mhammelberg0@nydailynews.com-Female']
[]
['2-Doralia-Brusin-dbrusin1@wired.com-Female']
[]
['3-Deane-Crowch-dcrowch2@boston.com-Female']
[]
['4-Hewe-Fruin-hfruin3@zdnet.com-Male']
[]
['5-Caryn-Bailes-cbailes4@goo.ne.jp-Female']
[]
['6-Christyna-Gabey-cgabey5@posterous.com-Female']
[]
['7-Lynnea-Dodell-ldodell6@scientificamerican.com-Female']
[]
```

```
['8-Koren-Moyler-kmoyler7@google.de-Female']
[]
['9-Ivonne-Siddons-isiddons8@pcworld.com-Female']
[]
['10-Kettie-Cleef-kcleef9@dell.com-Female']
[]
['11-Babita-Bugg-bbugga@pcworld.com-Female']
[]
['12-Idaline-Algore-ialgoreb@domainmarket.com-Female']
[]
['13-Krystyna-Linge-klingec@seesaa.net-Female']
[]
['14-Michele-Whorlow-mwhorlowd@state.tx.us-Male']
[]
['15-Phillipp-Chorley-pchorleye@jigsy.com-Male']
[]
['16-Arabella-Parzis-aparzisf@nationalgeographic.com-Female']
[]
['17-Rodolfo-Cauldwell-rcauldwellg@irs.gov-Male']
[]
['18-Sampson-Meneyer-smeneyerh@salon.com-Male']
[]
['19-Rois-Padginton-rpadgintoni@liveinternet.ru-Female']
[]
['20-Nealson-Vasler-nvaslerj@newyorker.com-Male']
[]
```

RegEx - Write and Match Regular Expression

In [150]:

```
import re   # Importing regular expression
```

In [151]:

```
# re.compile() --> allow us to seperate out patterns into a variable and also make it eaiser to re
use that variable to perform-
# - multiple searches
```

In [226]:

```
pattern = re.compile(r'M(r|s|rs)\.?\s[a-zA-Z]+\w')  # matching data in file from different
loactaion.
with open('C:/Users/Syed Afzal/Desktop/test.txt', 'r') as f:
    contents = f.read()
    matches = pattern.finditer(contents)
    for match in matches:
        print(match)
```

```
<re.Match object; span=(201, 212), match='Mr. Schafer'>
<re.Match object; span=(213, 221), match='Mr smith'>
<re.Match object; span=(222, 231), match='Mrs Davis'>
<re.Match object; span=(232, 239), match='Ms ella'>
```

Look Before You Leave Method (NoN Pythonic Way) - LBYL

Easier to Ask Forgiveness than Permission (Pythonic Way) - EAFP

In [255]:

```
# LBYL

person = {'name': 'Jess', 'age':26, 'job':'Programmer'}
persons = {'name': 'Jess', 'age':26}

if 'name' in person and 'age' in person and 'job' in person:
    print('Iam {name}, and Iam {age} years old. Iam a {job}'.format(**persons))
else:
    print('Some keys are missing')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-255-81b5791c36d1> in <module>
      5
      6 if 'name' in person and 'age' in person and 'job' in person:
----> 7     print('Iam {name}, and Iam {job} years old. Iam a {job}'.format(**persons))
      8 else:
      9     print('Some keys are missing')

KeyError: 'job'
```

In [258]:

```
# EAFP

try:
    print("Iam {name}, and Iam {age}. Iam a {job}".format(**persons))
except KeyError as e:
    print("Missing {} key".format(e))
```

```
Missing 'job' key
```

Object Oriented Programming

In [299]:

```
# Classess and Instances
class Employee:
    raise_amount = 1.04
    num_of_emps = 0
    def __init__(self,first,last,pay):
        self.first = first
        self.last = last
        self.pay = pay
        self.email = first+'.'+last+'@companymail.com'
        Employee.num_of_emps +=1

    def fullname(self):
        return '{} {}'.format(self.first, self.last)

    def apply_raise(self):
        self.pay = int(self.pay*self.raise_amount)

# Class methods and Static methods
# Regular method in classess automatically take instances as first argument
# To turn a regular method into a class method we have to add a decorator

    @classmethod
    def set_raise_amt(cls,amount): # cls = class variable
        cls.raise_amt = amount

Employee.set_raise_amt(1.05)

emp_1 = Employee('John', 'Doe', 60000)
emp_2 = Employee('Jess', 'ell', 70000)
emp_3 = Employee('Jenn', 'avx', 90000)

#print(emp_1.pay)
#emp_1.apply_raise()
#print(emp_1.pay)
#print(Employee.num_of_emps)
#print(Employee.raise_amt)
#print(emp_1.raise_amt)
#print(emp_2.raise_amt)

# Creating a static method
import datetime

def is_working_day(day):
    if day.weekday()==5 or day.weekday==6:
        return False
    else:
```

```
    else:
        return True
#my_day = datetime.date.today()

#is_working_day(my_day)

# Inheritance and Creating Subclassess

class Developer(Employee):  # Inheriting Employee class into Developer class
    def __init__(self, first, last, pay, pro_lang):
        Super().__init__(first,last,pay)
        self.pro_lang = pro_lang
class Manager(Employee):
    def __init__(self, first, last, pay, Employee=None):
        Supe.__init__(first, last, pay)
    if employees == None:
        self.employees = []
    else:
        self.employees= employees
### Will come back again
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-299-7826c5e4de84> in <module>
     56         Super().__init__(first,last,pay)
     57         self.pro_lang = pro_lang
---> 58 class Manager(Employee):
     59     def __init__(self, first, last, pay, employees=None):
     60         Supe.__init__(first, last, pay)

<ipython-input-299-7826c5e4de84> in Manager()
     59     def __init__(self, first, last, pay, employees=None):
     60         Supe.__init__(first, last, pay)
---> 61     if employees == None:
     62         self.employees = []
     63     else:

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

Special or Magic or Dunder Methods

In [313]:

```
def __add__(self):
    pass

print(int. __add__(1,2))

def __len__(self):
    pass

print('test'.__len__())
```

```
3
4
```

Property Decorator - Getter, Setter and Deleter

In [329]:

```
# Let us take an example

class Employee:
    def __init__(self, first, last):
        self.fist = first
        self.last = last
        #self.email = first+'.'+last+'@mail.com'

    @property
```

```
    def email(self):
        return '{}.{}@mail.com'.format(self.first, self.last)

    #def fullname(self):
     #   return '{} {}'.format(self.first, self.last)

    #@fullname setter
    def fullname(self,name):
        first, last = name.split(' ')     # Automatically detect first and last name no need to upda
te seperately
        self.first = first
        self.last = last

emp_1 = Employee('John', 'Doe')
emp_2 = Employee('Jess', 'mary')
emp_1.first = 'Jim'  # first name of emp_1 has been changed to jim

#print(emp_1.first) # reflect first name as jim - Updated
#print(emp_1.email) # where as in email the first name which is changed as Jim is not updated

# to update first and last name automatically in all fields first remove self.email set in class E
mployee

print(emp_1.email) # now the changes has been updated
print(emp_1.first)
```

```
Jim.Doe@mail.com
Jim
```

Working wih Json - Javascript Object Notation

In [330]:

```
from PIL import Image # importing pillow module
import os
```

In [333]:

```
img1 = Image.open('C:/Users/Syed Afzal/Desktop/nature.jpg')
#img1.show()  # open image and display on the screen
img1.save('C:/Users/Syed Afzal/Desktop/nat.png') # save the image with different file name and ext
ensions required
```