Introduction to Python

Data Structures in Python

1. Integer (int)
2. Float
3. String (str)
4. Boolean (bool)
5. List
6. Tuple
7. Sets
8. Array
9. Dictionary
10. Files

Integer , Float , String and Boolean

In [1]:

```
a = 13440
print(a)
print(type(a))
```

```
13440
<class 'int'>
```

In [2]:

```
a = 1.6
print(a)
print(type(a))
```

```
1.6
<class 'float'>
```

In [3]:

```
b = "This is python environment."
print(b)
print(type(b))
```

```
This is python environment.
<class 'str'>
```

In [4]:

```
c = True
print(c)
print(type(c))
```

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

In [5]:

```
a1 = 3
a2 = 4
```

In [6]:

```
a3 = a1*a2
print(a3)
```

```
12
```

In [7]:

```
a3 = a1**a2
print(a3)
```

```
81
```

In [8]:

```
a4 = "hey"
```

In [9]:

```
a5 = a4*a2
print(a5)
```

```
heyheyheyhey
```

In [10]:

```
a6 = "a1"*a2
print(a6)
```

```
a1a1a1a1
```

In [11]:

```
a7 = str(a1)*a2
print(a7)
```

```
3333
```

In [12]:

```
print('Data' + ' Science')
```

```
Data Science
```

In [13]:

```
Income1 = 50000
Income2 = 60000

print(type(Income1))
print(type(Income2))
```

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

In [14]:

```
print("Sumit Started with the income of Rs. " + Income1 + "It was gradually increased to Rs
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-f3da0be93aba> in <module>
----> 1 print("Sumit Started with the income of Rs. " + Income1 + "It was gr
adually increased to Rs. " + Income2 )

TypeError: can only concatenate str (not "int") to str
```

In [15]:

```
# Correct Method

print("Sumit Started with the income of Rs." + str(Income1) + ". It was gradually increased
```

```
Sumit Started with the income of Rs.50000. It was gradually increased to Rs.
60000
```

In [16]:

```
str.capitalize('python')
```

Out[16]:

```
'Python'
```

In [17]:

```
a8 = 'this is python class'
a8
```

Out[17]:

```
'this is python class'
```

In [18]:

```
a8.capitalize()
```

Out[18]:

```
'This is python class'
```

In [22]:

```
a9 = 'elepant'
a10 = '1234'
```

In [23]:

```
a9.isdigit()
```

Out[23]:

```
False
```

In [24]:

```
a10.isdigit()
```

Out[24]:

```
True
```

In [25]:

```
# Boolean
x = 4
y = 2
z = (x==y) # Comparison expression (Evaluates to false)
if z: # Conditional on truth/false value of 'z'
    print("We want to be a Data Scientists")
else: print("We DO not want to be a Data Scientists")
```

```
We DO not want to be a Data Scientists
```

In [26]:

```
# Implicit Conversions
# A float
x = 6.0

# An integer
y = 3

# Divide `x` by `y`
z = x/y

# Check the type of `z`
type(z)
```

Out[26]:

```
float
```

In [27]:

```
# Explicit Conversion
x = 8
y = "Game of Thrones: Season "
fav_season = y + x
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-773ae15bc933> in <module>
      2 x = 8
      3 y = "Game of Thrones: Season "
----> 4 fav_season = y + x

TypeError: can only concatenate str (not "int") to str
```

In [28]:

```
x = 8
y = "Game of Thrones: Season "
fav_season = (y) + str(x)
print(fav_season)
```

```
Game of Thrones: Season 8
```

Lists

Can be created using square " [ ] " bracket.

In [29]:

```
Income3 = [ 1000 , 2000 , 3000 , 4000 , 5000 , 6000 ]
```

In [30]:

```
# Different data types can be defined inside a list

Income4 = [ 'Rajesh' , 1000 , 'Mahesh' , 2000 , 'Suresh' , 3000 ]
```

List of Lists

In [31]:

```
Income5 = [['Rajesh',1000] , ['Mahesh',2000] , ['Suresh',3000] ]
print(Income5)
```

```
[['Rajesh', 1000], ['Mahesh', 2000], ['Suresh', 3000]]
```

In [32]:

```
print(type(Income3))
print(type(Income4))
print(type(Income5))
```

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

Accessing information from the list

Zero based indexing

In [33]:

```
print(Income3[0])
print(Income3[1])
```

```
1000
2000
```

In [34]:

```
print(Income4[0])
print(Income4[1])
```

```
Rajesh
1000
```

In [35]:

```
print(Income5[0])
print(Income5[0][0])
print(Income5[0][1])
```

```
['Rajesh', 1000]
Rajesh
1000
```

In [36]:

```
print(Income3[-1]) # Last element of the list
```

```
6000
```

In [37]:

```
# Accessing multiple elements from the list
# [Start:end] Start is inclusive while the End is exclusive

print(Income3[0:4])
```

```
[1000, 2000, 3000, 4000]
```

In [38]:

```
# to access all the elements from the list

print(Income3[:])
```

```
[1000, 2000, 3000, 4000, 5000, 6000]
```

List Manipulations

Replacing Elements

In [39]:

```
print(Income4)
```

```
['Rajesh', 1000, 'Mahesh', 2000, 'Suresh', 3000]
```

In [40]:

```
Income4[1] = 4000
print(Income4)
```

```
['Rajesh', 4000, 'Mahesh', 2000, 'Suresh', 3000]
```

In [41]:

```
Income4[2:4] = ['Venkatesh',5000]

print(Income4)
```

```
['Rajesh', 4000, 'Venkatesh', 5000, 'Suresh', 3000]
```

In [42]:

```
print(Income5)
```

```
[['Rajesh', 1000], ['Mahesh', 2000], ['Suresh', 3000]]
```

In [43]:

```
Income5[1] = ['Venkatesh',5000]
print(Income5)
```

```
[['Rajesh', 1000], ['Venkatesh', 5000], ['Suresh', 3000]]
```

In [44]:

```
Income5[0][1] = 6000
print(Income5)
```

```
[['Rajesh', 6000], ['Venkatesh', 5000], ['Suresh', 3000]]
```

Adding elements to the list

In [45]:

```
print(Income3)
```

```
[1000, 2000, 3000, 4000, 5000, 6000]
```

In [46]:

```
Income3 = Income3 + 7000
print(Income3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-46-ae24edc26979> in <module>
----> 1 Income3 = Income3 + 7000
      2 print(Income3)

TypeError: can only concatenate list (not "int") to list
```

In [47]:

```
Income3 = Income3 + [7000]
print(Income3)
```

```
[1000, 2000, 3000, 4000, 5000, 6000, 7000]
```

In [48]:

```
# by default new element will be added at the last position
# if is is to be added at at other position then we can use 'insert' function

Income3.insert(0,500)
```

In [49]:

```
Income3
```

Out[49]:

```
[500, 1000, 2000, 3000, 4000, 5000, 6000, 7000]
```

In [51]:

```
Income3.insert(4,3500)
Income3
```

Out[51]:

```
[500, 1000, 2000, 3000, 3500, 3500, 4000, 5000, 6000, 7000]
```

In [52]:

```
print(Income4)
```

```
['Rajesh', 4000, 'Venkatesh', 5000, 'Suresh', 3000]
```

In [53]:

```
Income4 = Income4 + ['Mahesh' , 6000]
print(Income4)
```

```
['Rajesh', 4000, 'Venkatesh', 5000, 'Suresh', 3000, 'Mahesh', 6000]
```

In [54]:

```
print(Income5)
```

```
[['Rajesh', 6000], ['Venkatesh', 5000], ['Suresh', 3000]]
```

In [55]:

```
Income5 = Income5 + [['Mahesh' , 4000]]
print(Income5)
```

```
[['Rajesh', 6000], ['Venkatesh', 5000], ['Suresh', 3000], ['Mahesh', 4000]]
```

Deleting element from the list

In [56]:

```
del(Income5[0:2])
print(Income5)
```

```
[['Suresh', 3000], ['Mahesh', 4000]]
```

In [57]:

```
print(Income3)
```

```
[500, 1000, 2000, 3000, 3500, 3500, 4000, 5000, 6000, 7000]
```

In [58]:

```
del(Income3[0:4])
print(Income3)
```

```
[3500, 3500, 4000, 5000, 6000, 7000]
```

In [59]:

```
print(Income4)
```

```
['Rajesh', 4000, 'Venkatesh', 5000, 'Suresh', 3000, 'Mahesh', 6000]
```

In [60]:

```
del(Income4[-2:])
```

In [61]:

```
print(Income4)
```

```
['Rajesh', 4000, 'Venkatesh', 5000, 'Suresh', 3000]
```

Making a copy of a list

In [62]:

```
Age = [22 , 24 , 35 , 46 , 67]
Age_New = Age
print(Age)
print(Age_New)
```

```
[22, 24, 35, 46, 67]
[22, 24, 35, 46, 67]
```

In [63]:

```
Age_New[1] = 89

print(Age)
print(Age_New)
```

```
[22, 89, 35, 46, 67]
[22, 89, 35, 46, 67]
```

In [64]:

```
# While replacing element in Age_New same element of Age_New is also getting replaced

# Solution

Age_New1 = Age.copy()

print(Age)
print(Age_New1)
```

```
[22, 89, 35, 46, 67]
[22, 89, 35, 46, 67]
```

In [65]:

```
Age_New1[1] = 27

print(Age)
print(Age_New1)
```

```
[22, 89, 35, 46, 67]
[22, 27, 35, 46, 67]
```

In [66]:

```
# We can know about the methods and function available with the help of dir command

print(dir(Income4))

# with _ : are funtions
# without_ : are methods
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__di
r__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__ge
titem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_
subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmu
l__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook
__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop',
'remove', 'reverse', 'sort']
```

In [67]:

```
len(Age) # Funtion : belongs to the object ( with _ )
```

Out[67]:

```
5
```

In [68]:

```
Age.sort()
print(Age)

# Method : belongs to the class of the object ( without _ )
```

```
[22, 35, 46, 67, 89]
```

Iterable and Iterator

In [69]:

```
# Iterable : an object that can be iterate over
# Iterator : an object that is used to iterate over
```

In [70]:

```
String1 ="Two Shōkaku-class aircraft carriers, Shōkaku and Zuikaku, were commissioned by th

print(String1)
```

```
Two Shōkaku-class aircraft carriers, Shōkaku and Zuikaku, were commissioned
by the Imperial Japanese Navy during World War II. They participated in the
attack on Pearl Harbor, the Indian Ocean Raid, and the battles of the Coral
Sea, the Eastern Solomons, and the Santa Cruz Islands. Their air groups sank
two of the four fleet carriers lost by the United States Navy during the war
in addition to one elderly British light carrier. Returning to Japan after t
he Battle of the Coral Sea to repair damage and replace lost aircraft, they
missed the Battle of Midway in June 1942. After the catastrophic loss of fou
r carriers during that battle, they formed the bulk of Japan's carrier force
for the rest of the war. Shōkaku was sunk by an American submarine during th
e Battle of the Philippine Sea in June 1944 as the Americans invaded the Mar
iana Islands, and Zuikaku was sacrificed as part of a decoy force four month
s later in the Battle of Leyte Gulf, both with heavy loss of life. Historian
Mark Peattie called them 'arguably the best aircraft carriers' of the early
1940s.
```

for loop

In [71]:

```
for c in String1[0:5] :
    print(c)

# Here String1 is an Iterable and c is an Iterator
```

```
T
w
o

S
```

Case Study

Use the assign_txt

1. How many times has the word RSS occured?
2. How many times has the word Modi occured?
3. Count number of total no of characters
4. Count number of capital letters
5. Count number of small letters

In [72]:

```
assign_txt="Narendra Damodardas Modi, born 17 September 1950) is the 15th and current Prime

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

```
Narendra Damodardas Modi, born 17 September 1950) is the 15th and current
Prime Minister of India, in office since 26 May 2014.odi, a leader of the
Bharatiya Janata Party was the Chief Minister of Gujarat from 2001 to 2014
and is the Member of Parliament from Varanasi. He led the BJP in the 2014
general election, which gave the party a majority in the Lok Sabha, the fi
rst for any political party in India since 1984.As the Chief Minister of G
ujarat, Modi's economic policies were praised, while his administration wa
s also criticised for failing to significantly improve the human developme
nt in the state, and for failing to prevent the 2002 Gujarat riots. A Hind
u nationalist and member of the Rashtriya Swayamsevak Sangh, Modi,remains
a controversial figure domestically and internationally. Modi was born on
17 September 1950, to a family of grocers in Vadnagar, Mehsana district, B
ombay State (present-day Gujarat).Modi's family belonged to the Modh-Ghanc
hi-Teli (oil-presser) community,hich is categorised as an Other Backward C
lass by the Indian government.Modi was the third of six children born to D
amodardas Mulchand (1915–1989) and Heeraben Modi (b. c. 1920).As a child,
Modi helped his father sell tea at the Vadnagar railway station, and later
ran a tea stall with his brother near a bus terminus.Modi completed his hi
gher secondary education in Vadnagar in 1967, where a teacher described hi
```

In [73]:

```
print(dir(assign_txt))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__
getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ite
r__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new_
_', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__se
tattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefo
ld', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'forma
t', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'is
digit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'i
stitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partit
ion', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstri
p', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'tran
slate', 'upper', 'zfill']
```

In [74]:

```
# 1

print(assign_txt.count('RSS'))
```

```
3
```

In [75]:

```
# 2

print(assign_txt.count('Modi'))
```

```
19
```

In [76]:

```
# 3

len(assign_txt)
```

Out[76]:

```
2822
```

In [77]:

```
# 4

count1 = 0

for l in assign_txt:
    if l.isupper():
        count1 = count1 + 1

print(count1)
```

```
120
```

In [78]:

```
# 5

count2 = 0

for l in assign_txt:
    if l.islower():
        count2 = count2 + 1

print(count2)
```

```
2119
```

Use the assign_txt

6. Find out the total number of lines in assign_txt (Use the split() method to create a list, assume "." to be EOL)
7. Extract the years like 1950 from the text and count their occurence

In [79]:

```
print(assign_txt)
```

```
Narendra Damodardas Modi, born 17 September 1950) is the 15th and current Pr
ime Minister of India, in office since 26 May 2014.odi, a leader of the Bhar
atiya Janata Party was the Chief Minister of Gujarat from 2001 to 2014 and i
s the Member of Parliament from Varanasi. He led the BJP in the 2014 general
election, which gave the party a majority in the Lok Sabha, the first for an
y political party in India since 1984.As the Chief Minister of Gujarat, Mod
i's economic policies were praised, while his administration was also critic
ised for failing to significantly improve the human development in the stat
e, and for failing to prevent the 2002 Gujarat riots. A Hindu nationalist an
d member of the Rashtriya Swayamsevak Sangh, Modi,remains a controversial fi
gure domestically and internationally. Modi was born on 17 September 1950, t
o a family of grocers in Vadnagar, Mehsana district, Bombay State (present-d
ay Gujarat).Modi's family belonged to the Modh-Ghanchi-Teli (oil-presser) co
mmunity,hich is categorised as an Other Backward Class by the Indian governm
ent.Modi was the third of six children born to Damodardas Mulchand (1915–198
9) and Heeraben Modi (b. c. 1920).As a child, Modi helped his father sell te
a at the Vadnagar railway station, and later ran a tea stall with his brothe
r near a bus terminus.Modi completed his higher secondary education in Vadna
gar in 1967, where a teacher described him as an average student and a keen
debater, with an interest in theatre.Modi had an early gift for rhetoric in
debates, and this was noted by his teachers and students.Modi preferred play
ing larger-than-life characters in theatrical productions, which has influen
ced his political image.At age eight, Modi discovered the Rashtriya Swayamse
vak Sangh (RSS), and began attending its local shakhas (training sessions).
There, Modi met Lakshmanrao Inamdar, popularly known as Vakil Saheb, who ind
ucted him as an RSS balswayamsevak (junior cadet) and became his political m
entor.While Modi was training with the RSS, he also met Vasant Gajendragadka
r and Nathalal Jaghda, Bharatiya Jana Sangh leaders who were founding member
s of the BJP's Gujarat unit in 1980.Engaged while still a child to a local g
irl, Jashodaben Narendrabhai Modi, Modi rejected the arranged marriage at th
e same time he graduated from high school.The resulting familial tensions co
ntributed to his decision to leave home in 1967.Modi spent the ensuing two y
ears travelling across Northern and North-eastern India, though few details
of where he went have emerged.In interviews, Modi has described visiting Hin
du ashrams founded by Swami Vivekananda: the Belur Math near Kolkata, follow
ed by the Advaita Ashrama in Almora and the Ramakrishna mission in Rajkot. M
odi remained only a short time at each, since he lacked the required college
education.
```

In [80]:

```
# 6

len(assign_txt.split('.'))
```

Out[80]:

```
23
```

In [81]:

```
print(dir(assign_txt))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__
getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ite
r__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new_
_', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__se
tattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefo
ld', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'forma
t', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'is
digit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'i
stitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partit
ion', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstri
p', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'tran
slate', 'upper', 'zfill']
```

In [82]:

```
# 7

SplitData = assign_txt.replace(","," ").replace("."," ").replace("("," ").replace(")"," ").
print(SplitData)
```

```
['Narendra', 'Damodardas', 'Modi', '', 'born', '17', 'September', '1950',
'', 'is', 'the', '15th', 'and', 'current', 'Prime', 'Minister', 'of', 'Indi
a', '', 'in', 'office', 'since', '26', 'May', '2014', 'odi', '', 'a', 'leade
r', 'of', 'the', 'Bharatiya', 'Janata', 'Party', 'was', 'the', 'Chief', 'Min
ister', 'of', 'Gujarat', 'from', '2001', 'to', '2014', 'and', 'is', 'the',
'Member', 'of', 'Parliament', 'from', 'Varanasi', '', 'He', 'led', 'the', 'B
JP', 'in', 'the', '2014', 'general', 'election', '', 'which', 'gave', 'the',
'party', 'a', 'majority', 'in', 'the', 'Lok', 'Sabha', '', 'the', 'first',
'for', 'any', 'political', 'party', 'in', 'India', 'since', '1984', 'As', 't
he', 'Chief', 'Minister', 'of', 'Gujarat', '', "Modi's", 'economic', 'polici
es', 'were', 'praised', '', 'while', 'his', 'administration', 'was', 'also',
'criticised', 'for', 'failing', 'to', 'significantly', 'improve', 'the', 'hu
man', 'development', 'in', 'the', 'state', '', 'and', 'for', 'failing', 't
o', 'prevent', 'the', '2002', 'Gujarat', 'riots', '', 'A', 'Hindu', 'nationa
list', 'and', 'member', 'of', 'the', 'Rashtriya', 'Swayamsevak', 'Sangh',
'', 'Modi', 'remains', 'a', 'controversial', 'figure', 'domestically', 'an
d', 'internationally', '', 'Modi', 'was', 'born', 'on', '17', 'September',
'1950', '', 'to', 'a', 'family', 'of', 'grocers', 'in', 'Vadnagar', '', 'Meh
sana', 'district', '', 'Bombay', 'State', '', 'present-day', 'Gujarat', '',
"Modi's", 'family', 'belonged', 'to', 'the', 'Modh-Ghanchi-Teli', '', 'oil-p
resser', '', 'community', 'hich', 'is', 'categorised', 'as', 'an', 'Other',
'Backward', 'Class', 'by', 'the', 'Indian', 'government', 'Modi', 'was', 'th
e', 'third', 'of', 'six', 'children', 'born', 'to', 'Damodardas', 'Mulchan
d', '', '1915', '1989', '', 'and', 'Heeraben', 'Modi', '', 'b', '', 'c', '',
'1920', '', 'As', 'a', 'child', '', 'Modi', 'helped', 'his', 'father', 'sel
l', 'tea', 'at', 'the', 'Vadnagar', 'railway', 'station', '', 'and', 'late
r', 'ran', 'a', 'tea', 'stall', 'with', 'his', 'brother', 'near', 'a', 'bu
s', 'terminus', 'Modi', 'completed', 'his', 'higher', 'secondary', 'educatio
n', 'in', 'Vadnagar', 'in', '1967', '', 'where', 'a', 'teacher', 'describe
d', 'him', 'as', 'an', 'average', 'student', 'and', 'a', 'keen', 'debater',
'', 'with', 'an', 'interest', 'in', 'theatre', 'Modi', 'had', 'an', 'early',
'gift', 'for', 'rhetoric', 'in', 'debates', '', 'and', 'this', 'was', 'note
d', 'by', 'his', 'teachers', 'and', 'students', 'Modi', 'preferred', 'playin
g', 'larger-than-life', 'characters', 'in', 'theatrical', 'productions', '',
'which', 'has', 'influenced', 'his', 'political', 'image', 'At', 'age', 'eig
ht', '', 'Modi', 'discovered', 'the', 'Rashtriya', 'Swayamsevak', 'Sangh',
'', 'RSS', '', '', 'and', 'began', 'attending', 'its', 'local', 'shakhas',
'', 'training', 'sessions', '', '', 'There', '', 'Modi', 'met', 'Lakshmanra
o', 'Inamdar', '', 'popularly', 'known', 'as', 'Vakil', 'Saheb', '', 'who',
'inducted', 'him', 'as', 'an', 'RSS', 'balswayamsevak', '', 'junior', 'cade
t', '', 'and', 'became', 'his', 'political', 'mentor', 'While', 'Modi', 'wa
s', 'training', 'with', 'the', 'RSS', '', 'he', 'also', 'met', 'Vasant', 'Ga
jendragadkar', 'and', 'Nathalal', 'Jaghda', '', 'Bharatiya', 'Jana', 'Sang
h', 'leaders', 'who', 'were', 'founding', 'members', 'of', 'the', "BJP's",
'Gujarat', 'unit', 'in', '1980', 'Engaged', 'while', 'still', 'a', 'child',
'to', 'a', 'local', 'girl', '', 'Jashodaben', 'Narendrabhai', 'Modi', '', 'M
odi', 'rejected', 'the', 'arranged', 'marriage', 'at', 'the', 'same', 'tim
e', 'he', 'graduated', 'from', 'high', 'school', 'The', 'resulting', 'famili
al', 'tensions', 'contributed', 'to', 'his', 'decision', 'to', 'leave', 'hom
e', 'in', '1967', 'Modi', 'spent', 'the', 'ensuing', 'two', 'years', 'travel
ling', 'across', 'Northern', 'and', 'North-eastern', 'India', '', 'though',
'few', 'details', 'of', 'where', 'he', 'went', 'have', 'emerged', 'In', 'int
erviews', '', 'Modi', 'has', 'described', 'visiting', 'Hindu', 'ashrams', 'f
ounded', 'by', 'Swami', 'Vivekananda:', 'the', 'Belur', 'Math', 'near', 'Kol
```

```
kata', '', 'followed', 'by', 'the', 'Advaita', 'Ashrama', 'in', 'Almora', 'a
nd', 'the', 'Ramakrishna', 'mission', 'in', 'Rajkot', '', 'Modi', 'remaine
d', 'only', 'a', 'short', 'time', 'at', 'each', '', 'since', 'he', 'lacked',
'the', 'required', 'college', 'education', '']
```

In [83]:

```
Numbers = []

for x1 in SplitData :
    if x1.isdigit():
        Numbers.append(x1)

print(Numbers)
```

```
['17', '1950', '26', '2014', '2001', '2014', '2014', '1984', '2002', '17',
'1950', '1915', '1989', '1920', '1967', '1980', '1967']
```

In [84]:

```
Years = []

for y1 in Numbers:
    if len(y1) == 4:
        Years.append(y1)

print(Years)
print(len(Years))
```

```
['1950', '2014', '2001', '2014', '2014', '1984', '2002', '1950', '1915', '19
89', '1920', '1967', '1980', '1967']
14
```

In [85]:

```
# 7

## Effective way of coding

Numbers1 = []
Years1 = []

for x2 in assign_txt.replace(","," ").replace("."," ").replace("("," ").replace(")"," ").re
    if x2.isdigit():
        Numbers1.append(x2)

for y2 in Numbers1:
    if len(y2) == 4 :
        Years1.append(y2)

print(Years1)
print(len(Years1))
```

```
['1950', '2014', '2001', '2014', '2014', '1984', '2002', '1950', '1915', '19
89', '1920', '1967', '1980', '1967']
14
```

List Comprehension

In [86]:

```
x = [ 2 , 3 , 4 , 5 ]

print(x)
```

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

In [87]:

```
# We want squares of all elements of x in new List
```

In [88]:

```
# Method 1 : Using for loop

x3 = []

for z in x:
    x3.append(z**2)

print(x3)
```

```
[4, 9, 16, 25]
```

In [89]:

```
# Method 2 : Using List Comprehension

x4 = [ z1**2 for z1 in x]

print(x4)
```

```
[4, 9, 16, 25]
```

Suppose we want to create a list from a and b such that it looks like the following:

[[1,'a'],[2,'b'],[3,'c'],[4,'d']]

In [90]:

```
a2 = [ 1, 2, 3, 4 ]
b2 = [ 'a', 'b', 'c', 'd' ]
```

In [91]:

```
[[z1,z2] for z1 in a2 for z2 in b2 ]


# this will give all the possible combinations
```

Out[91]:

```
[[1, 'a'],
 [1, 'b'],
 [1, 'c'],
 [1, 'd'],
 [2, 'a'],
 [2, 'b'],
 [2, 'c'],
 [2, 'd'],
 [3, 'a'],
 [3, 'b'],
 [3, 'c'],
 [3, 'd'],
 [4, 'a'],
 [4, 'b'],
 [4, 'c'],
 [4, 'd']]
```

In [92]:

```
[[z1,z2] for z1 in a2 for z2 in b2 if a2.index(z1) == b2.index(z2) ]
```

Out[92]:

```
[[1, 'a'], [2, 'b'], [3, 'c'], [4, 'd']]
```

In [93]:

```
# Using for loop

ab2 = []

for z1 in a2:
    for z2 in b2:
        if a2.index(z1) == b2.index(z2):
            ab2.append([z1,z2])

print(ab2)
```

```
[[1, 'a'], [2, 'b'], [3, 'c'], [4, 'd']]
```

In [94]:

```
# Alternate Method

zip(a2,b2)
```

Out[94]:

```
<zip at 0x193955c0c08>
```

In [95]:

```
print(list(zip(a2,b2)))
```

```
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
```

In [96]:

```
[list(x) for x in zip(a2,b2)]
```

Out[96]:

```
[[1, 'a'], [2, 'b'], [3, 'c'], [4, 'd']]
```

In [97]:

```
pairs = []

for z1 in range(0,2):
    for z2 in range(3,5):
        pairs.append([z1,z2])

print(pairs)
```

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

In [98]:

```
pairs1 = [[z1,z2] for z1 in range(0,2) for z2 in range(3,5)]

print(pairs1)
```

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

In [99]:

```
#List Comprehension - Conditions on output expression


a3 = [ 1, 'p' , 2 , 'q' , 3 , 'r' ]

[x for x in a3 if type(x)== str ]
```

Out[99]:

```
['p', 'q', 'r']
```

In [100]:

```
[ num2**2 for num2 in range(10,20) if num2%2 == 0]
```

Out[100]:

```
[100, 144, 196, 256, 324]
```

In [101]:

```
# Calculate BMI

mass=[45,55,65,76]
ht=[1.65,1.70,1.55,1.80]
```

In [102]:

```
# Method 1 : using for loop

BMI = []

for z1 in mass:
    for z2 in ht:
        if mass.index(z1) == ht.index(z2):
            BMI.append(z1/z2**2)

print(BMI)
```

```
[16.528925619834713, 19.031141868512112, 27.055150884495315, 23.456790123456
788]
```

In [103]:

```
# Method 2 : using list comprehension

[(z1/z2**2) for z1 in mass for z2 in ht if mass.index(z1) == ht.index(z2) ]
```

Out[103]:

```
[16.528925619834713,
 19.031141868512112,
 27.055150884495315,
 23.456790123456788]
```

Functions

- Used to write custom code and retrun a value based on your calculation

In [107]:

```
# Defining a Normal Function

def z5(z1,z2):
    return z1**z2

z5(3,4)
```

Out[107]:

```
81
```

In [110]:

```
def mycube(x):
    return x*x*x

mycube(3)
```

Out[110]:

```
27
```

In [112]:

```
mynums = [2,3,4,6,5,6,7,8,9,4,4,5,7]

def is_even(n):
    return n%2 == 0

even_nums = filter(is_even,mynums)  # Syntax : filter( function, object )
even_nums
```

Out[112]:

```
<filter at 0x19395c2f7c8>
```

In [113]:

```
list(even_nums)
```

Out[113]:

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

Lambdas

- We can use lambdas to write a function without a name
- We can also pass functions as object in Python
- Used when we need to just use it once

In [116]:

```
# Defining a lambda funtion and using in coding

z4 = ( lambda a,b : a**b )
Result1 = z4(2,5)

print(Result1)
```

```
32
```

In [117]:

```
myfunc = lambda x : x*x*x*x

myfunc(2)
```

Out[117]:

```
16
```

In [118]:

```
myfunc2 = lambda a,b : a*2+b

myfunc2(2,3)
```

Out[118]:

```
7
```

Lambda with Filter, Map, Reduce

- Filter is used to filter based on a criteria
- It uses 2 arguments; one is function and second is list

In [119]:

```
#Lets get the even numbers from this list

mynums
```

Out[119]:

```
[2, 3, 4, 6, 5, 6, 7, 8, 9, 4, 4, 5, 7]
```

In [120]:

```
#Get evens using Lambda

evens = list(filter(lambda x : x%2 ==0, mynums ))
evens
```

Out[120]:

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

In [128]:

```
#Now lets add 2 to each number; we use map for the same

myadd = list(map( lambda x : x+2, evens ))
myadd
```

Out[128]:

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

In [129]:

```
from functools import reduce

#Now add all these value to return one value

mysum = reduce(lambda a,b : a+b , list(myadd) )
mysum
```

Out[129]:

```
48
```

In [130]:

```
#Get y = m1x1+m2x2+m3x3

m=[45,55,65]
x=[2,3,4]

y = list(map(lambda a,b : a*b , m,x))
y
```

Out[130]:

```
[90, 165, 260]
```

In [131]:

```
# Method 3 : Using map and lambda funtion

BMI2 = map( lambda z1,z2 : z1/z2**2 , mass , ht )

print(BMI2)
```

```
<map object at 0x0000019395611A08>
```

In [132]:

```
BMI2_f = filter( lambda z3 : z3>20 , BMI2 )

print(list(BMI2_f))

# filter and lambda fn doesn't work on list
```

```
[27.055150884495315, 23.456790123456788]
```

In [133]:

```
Cities = [ 'Delhi' , 'Mumbai' , 'Chennai' , 'Ahmedabad' , 'Kolkata' , 'Pune']

Result2 = filter( lambda z1 : len(z1) > 6 , Cities )

print(list(Result2))
print(type(Result2))
```

```
['Chennai', 'Ahmedabad', 'Kolkata']
<class 'filter'>
```

Error Handling

"""Concatenate copies of word1 and two exclamation marks at the end of the string."""

In [134]:

```
def w3( z1 , z2 ):

    # Initialize empty strings

    w31 = ''
    s31 = ''


    # add exception handling with try-except

    try:

        # concatenate copies of the string using *

        w31 = z1*z2

        # concatenate '!!'

        s31 = w31 + '!!'


    except:

            # print error message

            print(' x argument must be a string and y argument must be a integer.')


    return(s31)


w3('Yay' , 'Great')
```

```
 x argument must be a string and y argument must be a integer.
```

Out[134]:

```
''
```

In [135]:

```
w3('Yay' , 3 )
```

Out[135]:

```
'YayYayYay!!'
```

Tuples

- Tuples as immutable iterables
- They are used instead of lists when we do not want that list to change

In [136]:

```
t = ( 0 , 1 , 2 , 3 )

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

```
(0, 1, 2, 3)
<class 'tuple'>
```

In [137]:

```
t[1]
```

Out[137]:

```
1
```

In [138]:

```
t[3] = 5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-138-bd39e8458e74> in <module>
----> 1 t[3] = 5

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

Sets

- Collection of unique elements
- can be created using the curly brackets

In [139]:

```
s1 = {12,34,65,76,98}
s1
```

Out[139]:

```
{12, 34, 65, 76, 98}
```

In [140]:

```
type(s1)
```

Out[140]:

```
set
```

In [141]:

```
print(dir(s1))
```

```
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash_
_', '__iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__i
ter__', '__ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__o
r__', '__rand__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__r
sub__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__su
bclasshook__', '__xor__', 'add', 'clear', 'copy', 'difference', 'difference_
update', 'discard', 'intersection', 'intersection_update', 'isdisjoint', 'is
subset', 'issuperset', 'pop', 'remove', 'symmetric_difference', 'symmetric_d
ifference_update', 'union', 'update']
```

In [142]:

```
a11 = s1.pop()
a11
```

Out[142]:

```
65
```

In [143]:

```
s1[1]

# can't access elements of set
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-143-0f209bc486aa> in <module>
----> 1 s1[1]
      2
      3 # can't access elements of set

TypeError: 'set' object is not subscriptable
```

In [144]:

```
s2 = { 12 , 23 , 34 , 45, 67, 76, 88 , 94 , 12 , 45 , 67 , 94 }
s2

# only unique elements are shown as output
```

Out[144]:

```
{12, 23, 34, 45, 67, 76, 88, 94}
```

In [145]:

```
s2.add(75)
s2
```

Out[145]:

```
{12, 23, 34, 45, 67, 75, 76, 88, 94}
```

In [146]:

```
s3 = {'a',1,2,4}
s3
```

Out[146]:

```
{1, 2, 4, 'a'}
```

In [147]:

```
s3.add('b')
s3
```

Out[147]:

```
{1, 2, 4, 'a', 'b'}
```

In [148]:

```
s31.add('z','h')
s3
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-148-4a2183dfa31f> in <module>
----> 1 s31.add('z','h')
      2 s3

NameError: name 's31' is not defined
```

In [149]:

```
s32 = {'z','h',6,8}
```

In [150]:

```
for z1 in s32:
    s3.add(z1)

s3
```

Out[150]:

```
{1, 2, 4, 6, 8, 'a', 'b', 'h', 'z'}
```

Arrays

In [151]:

```
import array as A
```

In [152]:

```
Array_char = A.array('u' , ('c','a','t','s'))
Array_char
```

Out[152]:

```
array('u', 'cats')
```

In [153]:

```
Array_ints1 = A.array('i' , (1,2,3,4))
Array_ints1
```

Out[153]:

```
array('i', [1, 2, 3, 4])
```

In [154]:

```
Array_ints2 = A.array('f' , (5,6,7,8))
Array_ints2
```

Out[154]:

```
array('f', [5.0, 6.0, 7.0, 8.0])
```

TYPE CODE	C TYPE	PYTHON TYPE	MINIMUM SIZE IN BYTES
'b'	signed char	int	1
'B'	unsigned char	int	1
'u'	Py_UNICODE	unicode character	2
'h'	signed short	int	2
'H'	unsigned short	int	2
'i'	signed int	int	2
'I'	unsigned int	int	2
'l'	signed long	int	4
'L'	unsigned long	int	4
'q'	signed long long	int	8
'Q'	unsigned long long	int	8
'f'	float	float	4
'd'	double	float	8

Dictionaries

- Dictionaries as hashables
- Extracting keys, values
- Sorting a dictionary
- Looping through a dictionary

In [155]:

```
Income = [ 10000 , 20000 , 30000 ]
name = [ 'Sagar' , 'Amruta' , 'Samay' ]
In_r = name.index('Amruta')
Income[In_r]
```

Out[155]:

```
20000
```

In [156]:

```
# Dictionary - key:value pairs

Inc1 = {'Sagar': 10000 , 'Amruta':20000, 'Samay':30000 }
print(Inc1)
```

```
{'Sagar': 10000, 'Amruta': 20000, 'Samay': 30000}
```

In [157]:

```
# Dictionary keys

print(Inc1['Sagar'])
```

```
10000
```

In [158]:

```
print(dir(Inc1))
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__do
c__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
'__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__',
'__len__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__
repr__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclassh
ook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popite
m', 'setdefault', 'update', 'values']
```

In [159]:

```
# print keys in Inc1

print(Inc1.keys())

# print values in Inc1

print(Inc1.values())
```

```
dict_keys(['Sagar', 'Amruta', 'Samay'])
dict_values([10000, 20000, 30000])
```

In [160]:

```
print(Inc1.items())
```

```
dict_items([('Sagar', 10000), ('Amruta', 20000), ('Samay', 30000)])
```

In [161]:

```
for x in Inc1:
    print(x)
```

```
Sagar
Amruta
Samay
```

In [162]:

```
for x in Inc1:
    print(x,Inc1[x])
```

```
Sagar 10000
Amruta 20000
Samay 30000
```

In [163]:

```
# list comprehension on dictionaries


pos_neg = { l:-l for l in range(3)}
print(pos_neg)
type(pos_neg)
```

```
{0: 0, 1: -1, 2: -2}
```

Out[163]:

```
dict
```

In [164]:

```
# Dictionary of dictionaries


Inc2 = {'Sagar':{'Gender': 'Male','Income':10000} , 'Amruta':{'Gender': 'Female','Income':2
        'Samay':{'Gender': 'Male','Income':30000} , 'Soumya':{'Gender': 'Female','Income':4

print(Inc2)
```

```
{'Sagar': {'Gender': 'Male', 'Income': 10000}, 'Amruta': {'Gender': 'Femal
e', 'Income': 20000}, 'Samay': {'Gender': 'Male', 'Income': 30000}, 'Soumy
a': {'Gender': 'Female', 'Income': 40000}}
```

In [165]:

```
# Print out the income of Samay

print(Inc2['Samay']['Income'])
```

```
30000
```

In [166]:

```
# Create sub-dictionary data

Inc21 = {'Gender': 'Male','Income':50000}
```

In [167]:

```
# Add sub dictionary data to Inc2

Inc2['Vidit'] = Inc21

print(Inc2)
```

```
{'Sagar': {'Gender': 'Male', 'Income': 10000}, 'Amruta': {'Gender': 'Femal
e', 'Income': 20000}, 'Samay': {'Gender': 'Male', 'Income': 30000}, 'Soumy
a': {'Gender': 'Female', 'Income': 40000}, 'Vidit': {'Gender': 'Male', 'Inco
me': 50000}}
```

In [168]:

```
# Replacing the elements


Inc2['Sagar']['Income'] = 60000

print(Inc2)
```

```
{'Sagar': {'Gender': 'Male', 'Income': 60000}, 'Amruta': {'Gender': 'Femal
e', 'Income': 20000}, 'Samay': {'Gender': 'Male', 'Income': 30000}, 'Soumy
a': {'Gender': 'Female', 'Income': 40000}, 'Vidit': {'Gender': 'Male', 'Inco
me': 50000}}
```

In [169]:

```
# del the elements


del(Inc2['Vidit'])

print(Inc2)
```

```
{'Sagar': {'Gender': 'Male', 'Income': 60000}, 'Amruta': {'Gender': 'Femal
e', 'Income': 20000}, 'Samay': {'Gender': 'Male', 'Income': 30000}, 'Soumy
a': {'Gender': 'Female', 'Income': 40000}}
```

Use the assign_txt¶

8. Find out the occurence of each word in text string (Create a list of each word in the string, loop over this list to create a dictionary of words and their occurences

In [170]:

```
assign_txt="Narendra Damodardas Modi, born 17 September 1950) is the 15th and current Prime

print(assign_txt)
```

```
Narendra Damodardas Modi, born 17 September 1950) is the 15th and current Pr
ime Minister of India, in office since 26 May 2014.odi, a leader of the Bhar
atiya Janata Party was the Chief Minister of Gujarat from 2001 to 2014 and i
s the Member of Parliament from Varanasi. He led the BJP in the 2014 general
election, which gave the party a majority in the Lok Sabha, the first for an
y political party in India since 1984.As the Chief Minister of Gujarat, Mod
i's economic policies were praised, while his administration was also critic
ised for failing to significantly improve the human development in the stat
e, and for failing to prevent the 2002 Gujarat riots. A Hindu nationalist an
d member of the Rashtriya Swayamsevak Sangh, Modi,remains a controversial fi
gure domestically and internationally. Modi was born on 17 September 1950, t
o a family of grocers in Vadnagar, Mehsana district, Bombay State (present-d
ay Gujarat).Modi's family belonged to the Modh-Ghanchi-Teli (oil-presser) co
mmunity,hich is categorised as an Other Backward Class by the Indian governm
ent.Modi was the third of six children born to Damodardas Mulchand (1915–198
9) and Heeraben Modi (b. c. 1920).As a child, Modi helped his father sell te
a at the Vadnagar railway station, and later ran a tea stall with his brothe
r near a bus terminus.Modi completed his higher secondary education in Vadna
gar in 1967, where a teacher described him as an average student and a keen
debater, with an interest in theatre.Modi had an early gift for rhetoric in
debates, and this was noted by his teachers and students.Modi preferred play
ing larger-than-life characters in theatrical productions, which has influen
ced his political image.At age eight, Modi discovered the Rashtriya Swayamse
vak Sangh (RSS), and began attending its local shakhas (training sessions).
There, Modi met Lakshmanrao Inamdar, popularly known as Vakil Saheb, who ind
ucted him as an RSS balswayamsevak (junior cadet) and became his political m
entor.While Modi was training with the RSS, he also met Vasant Gajendragadka
r and Nathalal Jaghda, Bharatiya Jana Sangh leaders who were founding member
s of the BJP's Gujarat unit in 1980.Engaged while still a child to a local g
irl, Jashodaben Narendrabhai Modi, Modi rejected the arranged marriage at th
e same time he graduated from high school.The resulting familial tensions co
ntributed to his decision to leave home in 1967.Modi spent the ensuing two y
ears travelling across Northern and North-eastern India, though few details
of where he went have emerged.In interviews, Modi has described visiting Hin
du ashrams founded by Swami Vivekananda: the Belur Math near Kolkata, follow
ed by the Advaita Ashrama in Almora and the Ramakrishna mission in Rajkot. M
odi remained only a short time at each, since he lacked the required college
education.
```

In [171]:

```
d1 = {}

for z1 in assign_txt.replace( "." ," " ).replace( "," ," " ).replace( "(" ," " ).replace( "
    if z1 in d1 :

        d1[z1] = d1[z1] + 1

    else :

        d1[z1] = 1

print(d1)
```

```
{'Narendra': 1, 'Damodardas': 2, 'Modi': 17, '': 54, 'born': 3, '17': 2, 'Se
ptember': 2, '1950': 2, 'is': 3, 'the': 28, '15th': 1, 'and': 15, 'current':
1, 'Prime': 1, 'Minister': 3, 'of': 10, 'India': 3, 'in': 15, 'office': 1,
'since': 3, '26': 1, 'May': 1, '2014': 3, 'odi': 1, 'a': 12, 'leader': 1, 'B
haratiya': 2, 'Janata': 1, 'Party': 1, 'was': 6, 'Chief': 2, 'Gujarat': 5,
'from': 3, '2001': 1, 'to': 9, 'Member': 1, 'Parliament': 1, 'Varanasi': 1,
'He': 1, 'led': 1, 'BJP': 1, 'general': 1, 'election': 1, 'which': 2, 'gav
e': 1, 'party': 2, 'majority': 1, 'Lok': 1, 'Sabha': 1, 'first': 1, 'for':
4, 'any': 1, 'political': 3, '1984': 1, 'As': 2, "Modi's": 2, 'economic': 1,
'policies': 1, 'were': 2, 'praised': 1, 'while': 2, 'his': 8, 'administratio
n': 1, 'also': 2, 'criticised': 1, 'failing': 2, 'significantly': 1, 'improv
e': 1, 'human': 1, 'development': 1, 'state': 1, 'prevent': 1, '2002': 1, 'r
iots': 1, 'A': 1, 'Hindu': 2, 'nationalist': 1, 'member': 1, 'Rashtriya': 2,
'Swayamsevak': 2, 'Sangh': 3, 'remains': 1, 'controversial': 1, 'figure': 1,
'domestically': 1, 'internationally': 1, 'on': 1, 'family': 2, 'grocers': 1,
'Vadnagar': 3, 'Mehsana': 1, 'district': 1, 'Bombay': 1, 'State': 1, 'presen
t-day': 1, 'belonged': 1, 'Modh-Ghanchi-Teli': 1, 'oil-presser': 1, 'communi
ty': 1, 'hich': 1, 'categorised': 1, 'as': 4, 'an': 5, 'Other': 1, 'Backwar
d': 1, 'Class': 1, 'by': 4, 'Indian': 1, 'government': 1, 'third': 1, 'six':
1, 'children': 1, 'Mulchand': 1, '1915': 1, '1989': 1, 'Heeraben': 1, 'b':
1, 'c': 1, '1920': 1, 'child': 2, 'helped': 1, 'father': 1, 'sell': 1, 'te
a': 2, 'at': 3, 'railway': 1, 'station': 1, 'later': 1, 'ran': 1, 'stall':
1, 'with': 3, 'brother': 1, 'near': 2, 'bus': 1, 'terminus': 1, 'completed':
1, 'higher': 1, 'secondary': 1, 'education': 2, '1967': 2, 'where': 2, 'teac
her': 1, 'described': 2, 'him': 2, 'average': 1, 'student': 1, 'keen': 1, 'd
ebater': 1, 'interest': 1, 'theatre': 1, 'had': 1, 'early': 1, 'gift': 1, 'r
hetoric': 1, 'debates': 1, 'this': 1, 'noted': 1, 'teachers': 1, 'students':
1, 'preferred': 1, 'playing': 1, 'larger-than-life': 1, 'characters': 1, 'th
eatrical': 1, 'productions': 1, 'has': 2, 'influenced': 1, 'image': 1, 'At':
1, 'age': 1, 'eight': 1, 'discovered': 1, 'RSS': 3, 'began': 1, 'attending':
1, 'its': 1, 'local': 2, 'shakhas': 1, 'training': 2, 'sessions': 1, 'Ther
e': 1, 'met': 2, 'Lakshmanrao': 1, 'Inamdar': 1, 'popularly': 1, 'known': 1,
'Vakil': 1, 'Saheb': 1, 'who': 2, 'inducted': 1, 'balswayamsevak': 1, 'junio
r': 1, 'cadet': 1, 'became': 1, 'mentor': 1, 'While': 1, 'he': 4, 'Vasant':
1, 'Gajendragadkar': 1, 'Nathalal': 1, 'Jaghda': 1, 'Jana': 1, 'leaders': 1,
'founding': 1, 'members': 1, "BJP's": 1, 'unit': 1, '1980': 1, 'Engaged': 1,
'still': 1, 'girl': 1, 'Jashodaben': 1, 'Narendrabhai': 1, 'rejected': 1, 'a
rranged': 1, 'marriage': 1, 'same': 1, 'time': 2, 'graduated': 1, 'high': 1,
'school': 1, 'The': 1, 'resulting': 1, 'familial': 1, 'tensions': 1, 'contri
buted': 1, 'decision': 1, 'leave': 1, 'home': 1, 'spent': 1, 'ensuing': 1,
'two': 1, 'years': 1, 'travelling': 1, 'across': 1, 'Northern': 1, 'North-ea
stern': 1, 'though': 1, 'few': 1, 'details': 1, 'went': 1, 'have': 1, 'emerg
ed': 1, 'In': 1, 'interviews': 1, 'visiting': 1, 'ashrams': 1, 'founded': 1,
'Swami': 1, 'Vivekananda:': 1, 'Belur': 1, 'Math': 1, 'Kolkata': 1, 'followe
d': 1, 'Advaita': 1, 'Ashrama': 1, 'Almora': 1, 'Ramakrishna': 1, 'mission':
```

```
1, 'Rajkot': 1, 'remained': 1, 'only': 1, 'short': 1, 'each': 1, 'lacked':
1, 'required': 1, 'college': 1}
```

Files

- Reading files
- Manipulating data in file objects
- Writing out files

In [172]:

```
import os

os.chdir("C:\\Users\\Swapnil bandekar\\Downloads\\Swapnil\\Data Analytics\\My Work\\Python\
```

In [173]:

```
file1 = open('tweets_assignment.txt','r')
file2 = file1.read()
print(type(file1))
print(type(file2))
file1.close()
```

```
<class '_io.TextIOWrapper'>
<class 'str'>
```

In [174]:

```
file3 = open("write1.txt",'w')

for dt in file2:
    file3.write(dt+'\n')
file3.close()

# created a new txt file with each word of file2 written on new line
```

In [175]:

```
file2[0:100]
```

Out[175]:

```
'"Tweet_Date"|"Tweet_Text"|"Tweet_Handle"\n"Thu Dec 26 123415 0000 2013"|"as
hdubey ratigirl FirstpostB'
```

Assignment

- Use the file tweets_assignment.txt
- Count the total number of tweets in the file
- Extract the time stamp from the file and store it in a list

In [176]:

```
# Alternate way to open a file


with open("tweets_assignment.txt" , "r") as file4:
    file5 = file4.read()

    file4.close()

file5[0:200]
```

Out[176]:

```
'"Tweet_Date"|"Tweet_Text"|"Tweet_Handle"\n"Thu Dec 26 123415 0000 2013"|"as
hdubey ratigirl FirstpostBiz First economic move by AamAadmiParty  How come
it has gone down in Guj"|"maverickenator"\n"Thu Dec'
```

In [177]:

```
# no of tweets


len(file5.split('\n'))
```

Out[177]:

```
14275
```

In [178]:

```
print(file5.split('\n'))
```

```
['"Tweet_Date"|"Tweet_Text"|"Tweet_Handle"', '"Thu Dec 26 123415 0000 201
3"|"ashdubey ratigirl FirstpostBiz First economic move by AamAadmiParty  H
ow come it has gone down in Guj"|"maverickenator"', '"Thu Dec 26 123413 00
00 2013"|"RT anandpassion Growing popularity can seen as AAP websites Rank
275 in India highest ever for a political party JoinAAP ankitlal Aam"|"aap
kask"', '"Thu Dec 26 123402 0000 2013"|"RT anandpassion Growing popularity
can seen as AAP websites Rank 275 in India highest ever for a political pa
rty JoinAAP ankitlal Aam"|"arpshalder"', '"Thu Dec 26 123354 0000 2013"|"A
rvindKejriwal AapYogendra AamAadmiParty I hope your party realizes why the
likes of alkalamba are trying to join AAP now Be careful"|"GargaC"', '"Thu
Dec 26 123351 0000 2013"|"RT ssttuuttii Press Conference AamAadmiParty App
lication form for candidature for LS2014 is being uploaded at website AAP4
India http"|"kejriwalfanclub"', '"Thu Dec 26 123346 0000 2013"|"anandpassi
on AamAadmiParty ankitlal Why Dont u ppl put Advertisement on ur website a
fter showing these numbersno NEED of Donations LOL"|"varaliprasoon"', '"Th
u Dec 26 123343 0000 2013"|"ChaloDelhi SatyaPrakashDu5 AamAadmiParty Thank
sWhat is the address of AAP Jamshedpur Unit of AAP"|"SharanVikash"', '"Thu
Dec 26 123341 0000 2013"|"Outlookindia When will Home Min probe financing
of other political parties AamAadmiParty ArvindKejriwal"|"gauravlavania"',
```

In [179]:

```
# Extart the time stamp

for line1 in file5.split('\n'):
    print(line1.split('|')[0])
```

```
"Tweet_Date"
"Thu Dec 26 123415 0000 2013"
"Thu Dec 26 123413 0000 2013"
"Thu Dec 26 123402 0000 2013"
"Thu Dec 26 123354 0000 2013"
"Thu Dec 26 123351 0000 2013"
"Thu Dec 26 123346 0000 2013"
"Thu Dec 26 123343 0000 2013"
"Thu Dec 26 123341 0000 2013"
"Thu Dec 26 123333 0000 2013"
"Thu Dec 26 123328 0000 2013"
"Thu Dec 26 123324 0000 2013"
"Thu Dec 26 123319 0000 2013"
"Thu Dec 26 123306 0000 2013"
"Thu Dec 26 123301 0000 2013"
"Thu Dec 26 123255 0000 2013"
"Thu Dec 26 123255 0000 2013"
"Thu Dec 26 123253 0000 2013"
"Thu Dec 26 123252 0000 2013"
```

NumPy

NumPy (or Numpy) is a Linear Algebra Library for Python, the reason it is so important for Data Science with Python is that almost all of the libraries in the PyData Ecosystem rely on NumPy as one of their main building blocks.

Numpy is also incredibly fast, as it has bindings to C libraries.

Numpy has many built-in functions and capabilities. We won't cover them all but instead we will focus on some of the most important aspects of Numpy: vectors,arrays,matrices, and number generation.

Numpy Arrays

Numpy arrays essentially come in two types : vectors and matrices. Vectors are strictly 1-d arrays and matrices are 2-d (but a matrix can still have only one row or one column).

Creating NumPy Arrays

From a Python List

We can create an array by directly converting a list or list of lists:

In [180]:

```
import numpy as np
```

In [181]:

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

Out[181]:

```
[1, 2, 3]
```

In [182]:

```
np.array(my_list)
```

Out[182]:

```
array([1, 2, 3])
```

In [183]:

```
my_matrix = [[1,2,3],[4,5,6],[7,8,9]]
my_matrix
```

Out[183]:

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

In [184]:

```
np.array(my_matrix)
```

Out[184]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Built-in Methods

There are lots of built-in ways to generate Arrays

arange

Return evenly spaced values within a given interval.

In [185]:

```
np.arange(0,11)
```

Out[185]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [186]:

```
np.arange(0,11,2)
```

Out[186]:

```
array([ 0,  2,  4,  6,  8, 10])
```

zeros and ones

returns arrays of zeros and ones

In [187]:

```
np.zeros(3)
```

Out[187]:

```
array([0., 0., 0.])
```

In [188]:

```
np.zeros((3,3))
```

Out[188]:

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [189]:

```
np.ones(5)
```

Out[189]:

```
array([1., 1., 1., 1., 1.])
```

In [190]:

```
np.ones((5,5))
```

Out[190]:

```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

linspace

Return evenly spaced numbers over a specified interval.

In [191]:

```
np.linspace(0,10,4)
```

Out[191]:

```
array([ 0.        ,  3.33333333,  6.66666667, 10.        ])
```

In [192]:

```
np.linspace(0,10,10)
```

Out[192]:

```
array([ 0.        ,  1.11111111,  2.22222222,  3.33333333,  4.44444444,
        5.55555556,  6.66666667,  7.77777778,  8.88888889, 10.        ])
```

In [193]:

```
np.linspace(0,10,50)
```

Out[193]:

```
array([ 0.        ,  0.20408163,  0.40816327,  0.6122449 ,  0.81632653,
        1.02040816,  1.2244898 ,  1.42857143,  1.63265306,  1.83673469,
        2.04081633,  2.24489796,  2.44897959,  2.65306122,  2.85714286,
        3.06122449,  3.26530612,  3.46938776,  3.67346939,  3.87755102,
        4.08163265,  4.28571429,  4.48979592,  4.69387755,  4.89795918,
        5.10204082,  5.30612245,  5.51020408,  5.71428571,  5.91836735,
        6.12244898,  6.32653061,  6.53061224,  6.73469388,  6.93877551,
        7.14285714,  7.34693878,  7.55102041,  7.75510204,  7.95918367,
        8.16326531,  8.36734694,  8.57142857,  8.7755102 ,  8.97959184,
        9.18367347,  9.3877551 ,  9.59183673,  9.79591837, 10.        ])
```

eye

Creates an identity matrix

In [194]:

```
np.eye(4)
```

Out[194]:

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

Random

Numpy also has lots of ways to create random number arrays:

rand

Create an array of the given shape and populate it with random samples from a uniform distribution over `[0, 1)` .

In [195]:

```
np.random.rand(2)
```

Out[195]:

```
array([0.10521124, 0.10709684])
```

In [196]:

```
np.random.rand(5,5)
```

Out[196]:

```
array([[0.54216368, 0.70644453, 0.58173857, 0.4723618 , 0.98590484],
       [0.23228482, 0.6892124 , 0.22757181, 0.31523001, 0.44591059],
       [0.76604424, 0.76274436, 0.01506603, 0.9311469 , 0.81600304],
       [0.21662181, 0.31391856, 0.629476  , 0.11636865, 0.3715718 ],
       [0.49040386, 0.42124859, 0.5717034 , 0.79648297, 0.59595728]])
```

randn

Return a sample (or samples) from the "standard normal" distribution. Unlike rand which is uniform

In [197]:

```
np.random.randn(2)
```

Out[197]:

```
array([ 0.92066078, -0.30991776])
```

In [198]:

```
np.random.randn(4,4)
```

Out[198]:

```
array([[ 1.15356798,  1.39513889,  0.3122405 , -0.92254931],
       [ 2.10634456,  0.96825995, -1.2229045 ,  1.09136067],
       [-1.0040257 , -0.45182068, -0.72745808,  0.16725752],
       [-0.91443608,  0.23247481, -0.70147332,  0.80811193]])
```

randint

Return random integers from `low` (inclusive) to `high` (exclusive).

In [199]:

```
np.random.randint(1,100)
```

Out[199]:

```
82
```

In [200]:

```
np.random.randint(1,100,10)
```

Out[200]:

```
array([88, 99, 62, 88,  4, 80, 16, 89, 73, 65])
```

Array Attributes and Methods

Reshape

Returns an array containing the same data with a new shape.

In [201]:

```
arr = np.arange(0,10)
ranarr = np.random.randint(0,50,30)
```

In [202]:

```
arr
```

Out[202]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [203]:

```
ranarr
```

Out[203]:

```
array([29, 44, 14, 34, 40, 44, 32,  3, 12, 24, 40, 16,  6,  2,  6, 28, 25,
       12,  0,  3, 30, 16,  8, 17, 35, 15, 48, 12,  6, 23])
```

In [204]:

```
ranarr.reshape(5,6)
```

Out[204]:

```
array([[29, 44, 14, 34, 40, 44],
       [32,  3, 12, 24, 40, 16],
       [ 6,  2,  6, 28, 25, 12],
       [ 0,  3, 30, 16,  8, 17],
       [35, 15, 48, 12,  6, 23]])
```

max,min,argmax,argmin

These are useful methods for finding max or min values. Or to find their index locations using argmin or argmax

In [205]:

```
ranarr.max()
```

Out[205]:

```
48
```

In [206]:

```
ranarr.argmax()
```

Out[206]:

```
26
```

In [207]:

```
ranarr.min()
```

Out[207]:

```
0
```

In [208]:

```
ranarr.argmin()
```

Out[208]:

```
18
```

Shape

Shape is an attribute that arrays have (not a method):

In [209]:

```
ranarr.shape
```

Out[209]:

```
(30,)
```

In [210]:

```
ranarr.reshape(30,1)
```

Out[210]:

```
array([[29],
       [44],
       [14],
       [34],
       [40],
       [44],
       [32],
       [ 3],
       [12],
       [24],
       [40],
       [16],
       [ 6],
       [ 2],
       [ 6],
       [28],
       [25],
       [12],
       [ 0],
       [ 3],
       [30],
       [16],
       [ 8],
       [17],
       [35],
       [15],
       [48],
       [12],
       [ 6],
       [23]])
```

In [211]:

```
ranarr.reshape(30,1).shape
```

Out[211]:

```
(30, 1)
```

In [212]:

```
ranarr.reshape(1,30)
```

Out[212]:

```
array([[29, 44, 14, 34, 40, 44, 32,  3, 12, 24, 40, 16,  6,  2,  6, 28,
        25, 12,  0,  3, 30, 16,  8, 17, 35, 15, 48, 12,  6, 23]])
```

In [213]:

```
users.groupby('occuption')ranarr.reshape(1,30).shape
```

Out[213]:

```
(1, 30)
```

In [214]:

```
arr.dtype
```

Out[214]:

```
dtype('int32')
```

In [215]:

```
ranarr.dtype
```

Out[215]:

```
dtype('int32')
```

If Elfi else

- Used to write custom code and retrun a value based on your calculation

In [220]:

```
a = np.random.randint(1,10)

if a > 3:
    print("not more than 5")

elif a == 5:
    print("Yes! it is")

else:
    print("Nope keep trying")
```

```
not more than 5
```

Bracket Indexing and Selection

- The simplest way to pick one or some elements of an array looks very similar to python lists:

In [229]:

```
arr_new = np.arange(0,30,4)
arr_new
```

Out[229]:

```
array([ 0,  4,  8, 12, 16, 20, 24, 28])
```

In [230]:

```
#Get a value at an index

arr_new[5]
```

Out[230]:

```
20
```

In [232]:

```
#Get values in a range

arr_new[2:4]
```

Out[232]:

```
array([ 8, 12])
```

Broadcasting

Numpy arrays differ from a normal Python list because of their ability to broadcast:

In [233]:

```
#Setting a value with index range (Broadcasting)

arr_new[0:4] = 100
arr_new
```

Out[233]:

```
array([100, 100, 100, 100,  16,  20,  24,  28])
```

In [234]:

```
arr_new1 = np.arange(0,10)
arr_new1
```

Out[234]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [236]:

```
# Important note on slicer

slice_of_arr_new1  = arr_new1[0:5]
slice_of_arr_new1
```

Out[236]:

```
array([0, 1, 2, 3, 4])
```

In [237]:

```
# Change values of slice of array

slice_of_arr_new1[ : ] = 99
slice_of_arr_new1
```

Out[237]:

```
array([99, 99, 99, 99, 99])
```

In [238]:

```
# lets have a look at the original array

arr_new1


# changes are done is original array as well
# Data is not copied, slice is a view of the original array! This avoids memory problems!
# to get a new copy we can use copy() method
```

Out[238]:

```
array([99, 99, 99, 99, 99,  5,  6,  7,  8,  9])
```

Indexing a 2D array (Matrices)

- The general format is array_name[row,col] or array_name[row][col]

In [241]:

```
arr_2d = np.array(([10,11,12],[13,14,15],[16,17,18]))
arr_2d
```

Out[241]:

```
array([[10, 11, 12],
       [13, 14, 15],
       [16, 17, 18]])
```

In [242]:

```
# Indexing row

arr_2d[1]
```

Out[242]:

```
array([13, 14, 15])
```

In [245]:

```
# Indexing column

arr_2d[:,1]
```

Out[245]:

```
array([11, 14, 17])
```

In [247]:

```
# 2D array slicing

#Shape (2,2) from top right corner

arr_2d[:2,1:]
```

Out[247]:

```
array([[11, 12],
       [14, 15]])
```

Fancy Indexing

- it means passing an array of indices to access multiple array elements at once.
- Fancy indexing allows you to select entire rows or columns out of order

In [261]:

```
value1 = np.random.randint(50,100,10)
value1
```

Out[261]:

```
array([68, 79, 77, 87, 80, 96, 75, 78, 77, 73])
```

In [263]:

```
# Suppose we want to access 3 different elements

[value1[2],value1[5],value1[8]]
```

Out[263]:

```
[77, 96, 77]
```

In [264]:

```
# Alternatively , we can pass a single list or array of indices to achieve the same

ind1 = [2,5,8]

value1[ind1]
```

Out[264]:

```
array([77, 96, 77])
```

In [269]:

```
# when using the fancy indexing , the shape the outlut depends on the shape of the index ar


ind2 = np.array([[2,5],
                 [1,7]])

value1[ind2]
```

Out[269]:

```
array([[77, 96],
       [79, 78]])
```

In [270]:

```
# Fancy indexing also works in multiple dimensions

value2 = np.arange(12).reshape((3, 4))
value2
```

Out[270]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [273]:

```
row = np.array([0,1,2])
col = np.array([1,2,3])

value2[row,col]
```

Out[273]:

```
array([ 1,  6, 11])
```

In [275]:

```
row[:, np.newaxis]
```

Out[275]:

```
array([[0],
       [1],
       [2]])
```

In [279]:

```
col[np.newaxis,:]
```

Out[279]:

```
array([[1, 2, 3]])
```

In [281]:

```
value2[row[:,np.newaxis],col]
```

Out[281]:

```
array([[ 1,  2,  3],
       [ 5,  6,  7],
       [ 9, 10, 11]])
```

In [284]:

```
value2[row,col[:,np.newaxis]]
```

Out[284]:

```
array([[ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [285]:

```
#Set up matrix

arr2d_1 = np.zeros((10,10))
arr2d_1
```

Out[285]:

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

In [290]:

```
#Length of array

arr_length = arr2d_1.shape[1]
arr_length
```

Out[290]:

```
10
```

In [291]:

```
# set up an array

for i in range(arr_length):
    arr2d_1[i] = i
```

In [292]:

```
arr2d_1
```

Out[292]:

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [2., 2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [3., 3., 3., 3., 3., 3., 3., 3., 3., 3.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [5., 5., 5., 5., 5., 5., 5., 5., 5., 5.],
       [6., 6., 6., 6., 6., 6., 6., 6., 6., 6.],
       [7., 7., 7., 7., 7., 7., 7., 7., 7., 7.],
       [8., 8., 8., 8., 8., 8., 8., 8., 8., 8.],
       [9., 9., 9., 9., 9., 9., 9., 9., 9., 9.]])
```

Selection

- How to use brackets for selection based off of comparison operators.

In [296]:

```
arr3 = np.arange(1,11)
arr3
```

Out[296]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [297]:

```
arr3>4
```

Out[297]:

```
array([False, False, False, False,  True,  True,  True,  True,  True,
        True])
```

In [299]:

```
bool_arr = arr3>4
bool_arr
```

Out[299]:

```
array([False, False, False, False,  True,  True,  True,  True,  True,
        True])
```

In [300]:

```
arr3[bool_arr]
```

Out[300]:

```
array([ 5,  6,  7,  8,  9, 10])
```

In [301]:

```
arr3[arr3>2]
```

Out[301]:

```
array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

In [302]:

```
x11 =2

arr3[arr3>2]
```

Out[302]:

```
array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

NumPy Operations

In [307]:

```
arr4 = np.arange(0,10)
arr4
```

Out[307]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [308]:

```
arr4 + arr4
```

Out[308]:

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

In [309]:

```
arr4 - arr4
```

Out[309]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [310]:

```
arr4*arr4
```

Out[310]:

```
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81])
```

In [311]:

```
arr4/arr4


# Warning on division by zero, but not an error!
# Just replaced with nan
```

```
C:\Users\Swapnil bandekar\Downloads\Swapnil\Data Analytics\anaconda3\lib\sit
e-packages\ipykernel_launcher.py:1: RuntimeWarning: invalid value encountere
d in true_divide
  """Entry point for launching an IPython kernel.
```

Out[311]:

```
array([nan,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
```

In [312]:

```
1/arr4

# Also warning, but not an error instead infinity
```

```
C:\Users\Swapnil bandekar\Downloads\Swapnil\Data Analytics\anaconda3\lib\sit
e-packages\ipykernel_launcher.py:1: RuntimeWarning: divide by zero encounter
ed in true_divide
  """Entry point for launching an IPython kernel.
```

Out[312]:

```
array([       inf, 1.        , 0.5       , 0.33333333, 0.25      ,
       0.2       , 0.16666667, 0.14285714, 0.125     , 0.11111111])
```

In [313]:

```
arr4*4
```

Out[313]:

```
array([ 0,  4,  8, 12, 16, 20, 24, 28, 32, 36])
```

In [314]:

```
#Taking Square Roots

np.sqrt(arr4)
```

Out[314]:

```
array([0.        , 1.        , 1.41421356, 1.73205081, 2.        ,
       2.23606798, 2.44948974, 2.64575131, 2.82842712, 3.        ])
```

In [315]:

```
#Calcualting exponential (e^)

np.exp(arr4)
```

Out[315]:

```
array([1.00000000e+00, 2.71828183e+00, 7.38905610e+00, 2.00855369e+01,
       5.45981500e+01, 1.48413159e+02, 4.03428793e+02, 1.09663316e+03,
       2.98095799e+03, 8.10308393e+03])
```

In [316]:

```
np.max(arr4)

# same as arr4.max()
```

Out[316]:

```
9
```

In [318]:

```
np.sin(arr4)
```

Out[318]:

```
array([ 0.        ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ,
       -0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849])
```

In [319]:

```
np.log(arr4)
```

```
C:\Users\Swapnil bandekar\Downloads\Swapnil\Data Analytics\anaconda3\lib\sit
e-packages\ipykernel_launcher.py:1: RuntimeWarning: divide by zero encounter
ed in log
  """Entry point for launching an IPython kernel.
```

Out[319]:

```
array([      -inf, 0.        , 0.69314718, 1.09861229, 1.38629436,
       1.60943791, 1.79175947, 1.94591015, 2.07944154, 2.19722458])
```

Numpy Excercises

1. Create an array of 10 zeros
2. Create an array of 10 ones
3. Create an array of 10 fives
4. Create an array of the integers from 10 to 50
5. Create an array of all the even integers from 10 to 50
6. Create a 3x3 matrix with values ranging from 0 to 8
7. Create a 3x3 identity matrix
8. Use NumPy to generate a random number between 0 and 1
9. Use NumPy to generate an array of 25 random numbers sampled from a standard normal distribution
10. Create the following matrix:

```
array([[ 0.01,  0.02,  0.03,  0.04,  0.05,  0.06,  0.07,  0.08,  0.09,  0.1 ],
       [ 0.11,  0.12,  0.13,  0.14,  0.15,  0.16,  0.17,  0.18,  0.19,  0.2 ],
       [ 0.21,  0.22,  0.23,  0.24,  0.25,  0.26,  0.27,  0.28,  0.29,  0.3 ],
       [ 0.31,  0.32,  0.33,  0.34,  0.35,  0.36,  0.37,  0.38,  0.39,  0.4 ],
       [ 0.41,  0.42,  0.43,  0.44,  0.45,  0.46,  0.47,  0.48,  0.49,  0.5 ],
       [ 0.51,  0.52,  0.53,  0.54,  0.55,  0.56,  0.57,  0.58,  0.59,  0.6 ],
       [ 0.61,  0.62,  0.63,  0.64,  0.65,  0.66,  0.67,  0.68,  0.69,  0.7 ],
       [ 0.71,  0.72,  0.73,  0.74,  0.75,  0.76,  0.77,  0.78,  0.79,  0.8 ],
       [ 0.81,  0.82,  0.83,  0.84,  0.85,  0.86,  0.87,  0.88,  0.89,  0.9 ],
       [ 0.91,  0.92,  0.93,  0.94,  0.95,  0.96,  0.97,  0.98,  0.99,  1.  ]])
```

11. Create an array of 20 linearly spaced points between 0 and 1

12. Create the following matrix:

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

13. Use Matrix from Q12 and replicate following output :

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

14. Use Matrix from Q12 and replicate following output :

```
20
```

15. Use Matrix from Q12 and replicate following output :

```
array([[ 2],
       [ 7],
       [12]])
```

16. Use Matrix from Q12 and replicate following output :

```
array([21, 22, 23, 24, 25])
```

17. Use Matrix from Q12 and replicate following output :

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

18. Get the sum of all the values in matrix `(325)`
19. Get the standard deviation of the values in mat `(7.2111025509279782)`
20. Get the sum of all the columns in mat `( array([55, 60, 65, 70, 75]) )`

In [320]:

```
# 1

np.zeros(10)
```

Out[320]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [321]:

```
# 2

np.ones(10)
```

Out[321]:

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In [322]:

```
# 3

np.ones(10)*5
```

Out[322]:

```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

In [323]:

```
# 4

np.arange(10,50)
```

Out[323]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
       44, 45, 46, 47, 48, 49])
```

In [324]:

```
# 5

np.arange(10,50,2)
```

Out[324]:

```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48])
```

In [330]:

```
# 5 - concise way

np.array(list(filter(lambda x : x%2 == 0 , np.arange(10,50))))
```

Out[330]:

```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48])
```

In [331]:

```
# 6.

np.arange(0,9).reshape(3,3)
```

Out[331]:

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

In [332]:

```
# 7

np.eye(3,3)
```

Out[332]:

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [333]:

```
# 8

np.random.rand(2)
```

Out[333]:

```
array([0.0148278 , 0.64338702])
```

In [334]:

```
# 9

np.random.randn(25)
```

Out[334]:

```
array([ 0.32427637,  0.33316204,  0.66875865,  1.20428061, -0.17684916,
        0.49243329, -1.74903795, -2.07829765,  1.03589818,  0.28923411,
       -0.11271368, -1.01945073, -0.81142607,  0.3167488 ,  0.6308168 ,
        1.11137936, -0.420039  , -0.03936239,  0.46584199, -0.25490644,
       -1.0802023 ,  0.02902356, -1.04662905,  0.68615462, -1.17810558])
```

In [338]:

```
# 10

np.linspace(0.01,1,100).reshape(10,10)
```

Out[338]:

```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

In [339]:

```
# 11

np.linspace(0,1,20)
```

Out[339]:

```
array([0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
       0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
       0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
       0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

In [344]:

```
# 12

mat = np.arange(1,26).reshape(5,5)
mat
```

Out[344]:

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [345]:

```
# 13

mat[2:,1:]
```

Out[345]:

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [352]:

```
# 13 : Alternate way

row = np.array([2,3,4])
col = np.array([1,2,3,4])

mat[row[:,np.newaxis],col]
```

Out[352]:

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [353]:

```
# 14

mat[3,4]
```

Out[353]:

```
20
```

In [356]:

```
# 15

mat[0:3,1].reshape(3,1)
```

Out[356]:

```
array([[ 2],
       [ 7],
       [12]])
```

In [358]:

```
# 15 : Alternate way

alt1 = mat[0:3,1]
alt1[:,np.newaxis]
```

Out[358]:

```
array([[ 2],
       [ 7],
       [12]])
```

In [359]:

```
# 16

mat[3]
```

Out[359]:

```
array([16, 17, 18, 19, 20])
```

In [360]:

```
# 17

mat[[3,4]]
```

Out[360]:

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [361]:

```
# 18

np.sum(mat)
```

Out[361]:

```
325
```

In [363]:

```
# 19

np.std(mat)
```

Out[363]:

```
7.211102550927978
```

In [365]:

```
# 20

sum(mat)
```

Out[365]:

```
array([55, 60, 65, 70, 75])
```