Core python programming regards to the datascience

```
#http://localhost:8888/notebooks/NAREH%20IT/AMXWAM_C2_DataStructures.ipynb#Core-python-programming-regards-to-the-datascience# =============TYPES OF VARIABLES/data types ( INT , FLOAT, STRING, BOOL)================

# int -
x=2
x
```

```
2
```

```
x
```

```
2
```

```
type(x)
```

```
int
```

```
# float -
y = 4.6
```

```
y
```

```
4.6
```

```
type(y)
```

```
float
```

```
# string -
z = 'hello'
```

```
z
```

```
'hello'
```

```
z1 = "hi"


z1
print(z1)
type(z1)
```

```
hi
```

```
str
```

```
# bool // logical -
g = True
g
```

```
True
```

```
m1 = False + True
m1
```

```
1
```

```
g
```

```
True
```

```
a = True + True
a
```

```
2
```

```
i = "False"
```

```
i
```

```
'False'
```

```
a2=45
```

```
b2=56
```

```
a2
#b2
```

```
45
```

```
print(a2)
b2
```

```
45
```

```
56
```

```
# ================== USING VARIABLES ======================
```

```
# Arithmetic -
A = 10
B = 5

C = A + B
C
```

```
15
```

```
D = B/A
D
```

```
0.5
```

```
E = A*B
E
```

```
50
```

```
F = B-A
F
```

```
-5
```

```
C
```

```
15
```

```
D
```

```
0.5
```

```
E
```

```
50
```

```
F
```

```
-5
```

```
# why is it better to use print() // (ANS) - all o/p will display one
time
print(C)
print(D)
print(E)
print(F)
```

```
15
0.5
50
-5
```

```
# How to run the square root operation -
import math # math is the package which you are importing

math.sqrt(25) # 16 is the argument of the sqrt function
```

```
5.0
```

```
A = 10

math.sqrt(A) #A=10 which we declare at begining
```

```
3.1622776601683795
```

```
round(math.sqrt(A))
```

```
3
```

```
# lets working with string
greeting = 'hello'
name = " aspired data scientitst"

message = greeting +  name
message
```

```
'hello aspired data scientitst'
```

```
name = " nareshIT"
message = greeting + name
message
```

```
'hello nareshIT'
```

```
# ================ BOOL VARIABLES AND OPERATORS ====================
4<5
```

```
True
```

```
10>100
```

```
False
```

```
# <
# >
# <=
# >=
# and
# or
# not

result = 4<5

result
```

```
True
```

```
type(result)
```

```
bool
```

```
result2 = 5>1

result2
```

```
True
```

```
result3 = not(5>1)

result3
```

```
False
```

```
result
```

```
True
```

```
result2
```

```
True
```

```
result3
```

```
False
```

```
result or result2 or result3
```

```
True
```

```
result or result3 or result2
```

```
True
```

```
result2 and result3
```

```
False
```

```
result2 or result3
```

```
True
```

```
result and result2 or result3
```

```
True
```

```
# ================== WHILE LOOP ========================

# what is while loop why used for - lets discuss that---

#while(condition){   # checking the condition if the conditiong is
true then execute the loop and it keep continue thats why it called as
loop
#     executable code1                        # if the condition is
false then print to code4
#     executable code2
#     executable code3
# }
# executable code4

# === python is indentation based program & no need to required
# very important you have to check the indentataion & need to maintain
properly
# python there is no curly braces
# ------------------------------------------------------------

#while condition:
```

```
#    executable code1
#    executable code2
#    executable code3
#executable code4

num = 12
while num<11:
    print(num)
    num = num+1
print('END OF THE PROGRAM')
```

```
END OF THE PROGRAM
```

```
count = 13
while count<12:
    print(count)
    count = count + 1
print("End the loop")
```

```
End the loop
```

```
#count = 22  #how to print infinite loop
#while count>12: # 22>12
#    print(count)
#    count = count + 1
#print("End the loop")

# ==================== FOR LOOP ==============================

# i define as iteration
for i in range(5):  # while loop we check the  condition
    print('Hello python')
```

```
Hello python
Hello python
Hello python
Hello python
Hello python
```

```
for j in range(5):
    print("Hello python:", j)
    print('welcome to naresh')
```

```
Hello python: 0
welcome to naresh
Hello python: 1
welcome to naresh
Hello python: 2
welcome to naresh
Hello python: 3
welcome to naresh
```

```
Hello python: 4
welcome to naresh
```

```
for i in range(7):
    print("Hello python:",i)
    #print("Hello python:",i)
```

```
Hello python: 0
Hello python: 1
Hello python: 2
Hello python: 3
Hello python: 4
Hello python: 5
Hello python: 6
```

```
#Another way
mylist = [10,100,1000,2000]
type(mylist)
```

```
list
```

```
mylist[:3]
```

```
[10, 100, 1000]
```

```
mylist
```

```
[10, 100, 1000, 2000]
```

```
mylist[::-1]
```

```
[2000, 1000, 100, 10]
```

```
mylist
```

```
[10, 100, 1000, 2000]
```

```
mylist[:-1]
```

```
[10, 100, 1000]
```

```
mylist
```

```
[10, 100, 1000, 2000]
```

```
mylist[:-2]
```

```
[10, 100]
```

```
mylist
```

```
[10, 100, 1000, 2000]
```

```
mylist[-2:]
```

```
[1000, 2000]
```

```
mylist
```

```
mylist[:-2]
```

```
mylist
```

```
mylist[4]
```

```
mylist
```

```
for j in mylist:
    {
        print('jj is equal to :', j)
    }
```

```
jj is equal to : 10
jj is equal to : 100
jj is equal to : 1000
jj is equal to : 2000
```

```
for j in mylist:
    print('jj is equal to :', j)
```

```
jj is equal to : 10
jj is equal to : 100
jj is equal to : 1000
jj is equal to : 2000
```

```
range(5)  #very quick way to creat a set of 5 numbers
```

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

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

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

```
list(range(10)) #what actually range 5 does
```

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

```
list(range(1,10))
```

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

```
list(range(10,40,5))
```

```
[10, 15, 20, 25, 30, 35]
```

```
list(range(1,10,3))
```

```
[1, 4, 7]
```

```
# ================= IF STATEMENT ======================
# -----(-2),(-1),(0),(1),(2)------------

import numpy as np # numpy is package is a scientific computation,
matrix , array
from numpy.random import randn

randn()
```

```
-0.7369240633868063
```

```
# Python Program for simple OTP genertaor
import random as r
# function for otp generation
def otpgen():
    otp=""
    for i in range(2):
        otp+=str(r.randint(1, 5))
    print ("Your One Time Password is ")
    print (otp)
otpgen()
```

```
Your One Time Password is
14
```

```
# ========== IF-ELSE STATEMENT ====================
# -----(-2),(-1),(0),(1),(2)------------

#answer = None
x = randn()
if x>1:   # if is the keyword then use condition & if the condition is
true then code got executed and keep going down
    answer = "Greater then 1"
else:
    answer = "Less then 1"
print(x)
print(answer)
```

```
-0.7760316793046214
Less then 1
```

what we learned so far -

1.Type of variable 2.Using variable 3.Logical variable and operator 4.while loop 5.For loop 6.if statement, if _ else statement, elif statement 7.random variables: randn() 8.introduce to numpy & pandas

Fundamentals of python which are used in datascience -

```
# List - ordered sequeence of elements & always start with 0
# you can decleare the mixed datatype variables
```

```
# List closed in [] bracket

MyFirstList = [5, 50, 67, 2020]

MyFirstList

[5, 50, 67, 2020]

type(MyFirstList)

MysecondList = [2, 5.0,'AMXWAM',True,40+67j] #multiple datatypes you can declared in the list

MysecondList

type(MysecondList)

amx = ['how are you?',55, (3,4,5)] #list inside the list

amx

fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana', 'apple']

len(fruits)

8

fruits.count('kiwi')

1

fruits.count('apple')

3

range(15)

range(0, 15)

list(range(15))

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

x = [1,2,3,4,5,6,7]
x

y = list(range(8))

y

[0, 1, 2, 3, 4, 5, 6, 7]

list(range(1,8))
```

```
z = list(range(1,8))

z
```

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

```
a1 = list(range(5,8))
a1

a2 = list(range(100,120))
a2

a3 = list(range(10,20)) # 10 form right 20-1
a3

a4 = list(range(10,20,5))  # adding 3 steps from 10 to 20

a4
```

```
[10, 15]
```

```
xx = list(range(1,5,2))

xx

a5 = list(range(20,40,5))
a5

# Now we can check how to read information from the list
w = ['a','e','i','o','u']

w
```

```
['a', 'e', 'i', 'o', 'u']
```

```
w[0]
```

```
'a'
```

```
w[1]
```

```
'e'
```

```
w[2]
```

```
'i'
```

```
w

#if you have more no of list so how to know how many are there

len(w)
```

```
5
```

```
w
```

```
['a', 'e', 'i', 'o', 'u']
```

```
# a,e,i,o,u
# 0,1,2,3,4 --> left to rigth
# -5,-4,-3,-2,-1 --> right to left
```

```
w[-4]
```

```
'e'
```

```
w
```

```
['a', 'e', 'i', 'o', 'u']
```

```
w[:]
```

```
['a', 'e', 'i', 'o', 'u']
```

```
w[:-3]
```

```
['a', 'e']
```

```
w
```

```
['a', 'e', 'i', 'o', 'u']
```

```
w[:4]
```

```
['a', 'e', 'i', 'o']
```

```
w
```

```
['a', 'e', 'i', 'o', 'u']
```

```
w[3:]
```

```
['o', 'u']
```

```
w[0]
```

```
'a'
```

```
# overwrite values
w
```

```
['a', 'e', 'i', 'o', 'u']
```

```
w[2] = 10
w[2]
```

```
10
```

```
w
```

```
['a', 'e', 10, 'o', 'u']
```

```
# slicing : operator
x1 = ['a', 'b', 'c', 'd', 'e' ,'f', 'g', 'h']
x1
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```

```
x1[4:7] #left side part always begins with 0 but right side always
have to (no.-1)// 7-1
```

```
['e', 'f', 'g']
```

```
x1
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```

```
x1[-5:-3]
```

```
['d', 'e']
```

```
x1[-5:-2]
```

```
['d', 'e', 'f']
```

```
x1
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```

```
x1[2:7]
```

```
x1
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```

```
x1[-6:-1]
```

```
['c', 'd', 'e', 'f', 'g']
```

```
#advanced slicing - [_:_:_] starting,ending,step
#if you specified : one more time then you get 3 spot last step is
nothing but step
x1
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```

```
x1[0:7:2]
```

```
['a', 'c', 'e', 'g']
```

```
x1[3:7:3]
```

```
['d', 'g']
```

```
x1
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
```

```
x1[3:8:2]
```

```
['d', 'f', 'h']
```

```
hallo = ['A','B','C','D','E','F','G','H','I','J']
hallo
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

```
hallo[1:3]
```

```
hallo[:]
```

```
hallo[1]
```

```
hallo
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

```
hallo[2:]
```

```
['C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

```
hallo
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

```
hallo[:4]
```

```
['A', 'B', 'C', 'D']
```

```
hallo
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

```
hallo[2:10:2]
```

```
['C', 'E', 'G', 'I']
```

```
hallo
```

```
# --- advanced slicing
hallo[2:9:2]
```

```
hallo
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

```
hallo[::-1] # : - PRINT 1ST - LAST, :-1 -> REVERS DIRECT
```

```
['J', 'I', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']
```

```
hallo
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
hallo[::-3]
['J', 'G', 'D', 'A']
hallo[::-2]
['J', 'H', 'F', 'D', 'B']
hallo
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
hallo[-2:]
['I', 'J']
hallo
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
hallo[::2]   #
['A', 'C', 'E', 'G', 'I']
hallo
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
hallo[::-1]  #reversing order print
hallo
hallo[::-2]
['J', 'H', 'F', 'D', 'B']
hallo
hallo[::-4]
hallo
hallo[:-3]
hallo
hallo[:3]
hallo
hallo[6:0:-2]
```

```
hallo
```

```
hallo[6:2:-2]
```

```
hallo
```

```
hallo[7:1:-3]
```

```
#Tuples -- Immutable list or immutable sequence of values
```

```
t = (10,20,90)
```

```
t
```

```
(10, 20, 90)
```

```
t[1]
```

```
20
```

```
t[1]=30  #you are unable to change after declaration
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-77-eac27e118d6d> in <module>
----> 1 t[1]=30  #you are unable to change after declaration

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

```
# Functions in python
```

```
range(20, 31)
```

```
range(20, 31)
```

```
list(range(20, 31))
```

```
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
```

```
mylist1 = list(range(20, 31))
```

```
mylist1
```

```
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
```

```
len(mylist1)
```

```
11
```

```
type(mylist1)
```

```
list
```

```
mylist1
```

```
[20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
```

```
max(mylist1)
```

```
30
```

```
min(mylist1)
```

```
20
```

```
max(1,450,234)
```

```
450
```

```
min(1,450,234)
```

```
1
```

```
# PACKAGES IN PYTHON
# PACKAGE - BOX, BLENDERS -FUNCTIONS, FRUITS - CLASS,DATA etc...
# where & how to find the packages - github,googles Q & A query
# Install packages in anaconda prompt
# conda install package name // pip install package name
# import the packages
# from package import module
# give an example of tensorflow how to install in pip

# NUMPY & ARRAY IN PYTHON
# numpy is the package you have to import
# list is similar like to array
# numpy is also inbuild array in python

l = [23245, 27, 546, 215, -1234]
type(l)
```

```
list
```

```
l
```

```
[23245, 27, 546, 215, -1234]
```

```
l2 = [23245.5, 27.8, 546.9, 215.2, -1234.0, 'amx', 7.8]
l2
```

```
[23245.5, 27.8, 546.9, 215.2, -1234.0, 'amx', 7.8]
```

```
import numpy as np

a = np.array(l)
type(a)
#type(l)
```

```
numpy.ndarray
```

```
a
```

```
array([23245,    27,   546,   215, -1234])
```

```
a1 = np.array(l2)
a1
```

```
array(['23245.5', '27.8', '546.9', '215.2', '-1234.0', 'amx', '7.8'],
      dtype='<U32')
```

```
l1 = [3.0, 4.6]
l1
```

```
import numpy as np
```

```
a = np.array(l) # list is convert into arrays
```

```
a
```

```
b = np.array(l1)
b
```

```
# what is the main difference between list & array
# you cannot create an array of different data types
```

```
b = np.array(12,True)
b
```

```
print(b)  #everything is converted into string
```

```
l  #when you select. and then tab you get list of function
```

```
[23245, 27, 546, 215, -1234]
```

```
l.pop()
```

```
-1234
```

```
l
```

```
[23245, 27, 546, 215]
```

```
l.pop()
```

```
215
```

```
l
```

```
[23245, 27, 546]
```

```
a
```

```
array([23245,    27,   546,   215, -1234])
```

```
a.mean()
```

```
4559.8
```

```
a.mode()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-107-dd961ae6378b> in <module>
----> 1 a.mode()

AttributeError: 'numpy.ndarray' object has no attribute 'mode'
```

```
a.median()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-108-4f2ec5b41859> in <module>
----> 1 a.median()

AttributeError: 'numpy.ndarray' object has no attribute 'median'
```

```
a
```

```
array([23245,    27,   546,   215, -1234])
```

```
# slicing array
```

```
l
```

```
[23245, 27, 546]
```

```
l[1:]
```

```
[27, 546]
```

```
l[0:2]
```

```
[23245, 27]
```

```
l
```

```
l[:]
```

```
l[::2]
```

```
a
```

```
a[2:]
```

```
a
```

```
a[2:4]
```

```
a
```

```
array([23245,    27,   546,   215, -1234])
```

```
c = a.copy() # when you create a copy of numpy array, modification can not be done both object array but in list it
```

```
c
```

```
array([23245,    27,   546,   215, -1234])
```

```
a
```

```
array([23245,    27,   546,   215, -1234])
```

```
c[0]
```

```
23245
```

```
c[0] = 20
```

```
c
```

```
array([   20,    27,   546,   215, -1234])
```

```
a
```

```
a[0]
```

```
a
```

```
c
```

```
m = [1,2,3]
m[0]
```

```
m[0]= 81
```

```
m
```

```
n = list.copy(m)  # when you create a copy of list, modification will done both object
```

```
n
```

```
n=m.copy()
```

```
n
```

```
m
```

what we learned so far --

''' 1> list 2> create some list[],range() 3> using [] bracket 4> slicing list 5> Tuples 6> functions 7> packages 8> Numpys & Arrays 9> slicing arrays'''

PYTHOH END -

```

```