session 1 notebook

The notebook has been written during the first session
please watch the video on "Course Materials" section of iLearn for the full description

April 5, 2020

1 Fundamentals of python programming

1.0.1 1) variables and types

[3]:
```
#integer
a=1
type(a)
```

[3]:
```
int
```

[4]:
```
#float
b=3.5
type(b)
```

[4]:
```
float
```

[5]:
```
#boolean
c=True
type(c)
```

[5]:
```
bool
```

[6]:
```
#complex
d=5-3j
type(d)
```

[6]:
```
complex
```

[7]:
```
#take the real part
d.real
```

[7]:
```
5.0
```

[8]:
```
# imaginary part
d.imag
```

[8]:
```
-3.0
```

[9]:
```
#string
e="welcome to PHYS247!"
type(e)
```

[9]:
```
str
```

1.0.2 2) Operators

[10]:
```
# kind of calculator
3+4,5-6,7*8,1/2,4**2
```

[10]:
```
(7, -1, 56, 0.5, 16)
```

[11]:
```
#integer division
7//3
```

[11]:
```
2
```

[12]:
```
#remainder
7%3
```

[12]:
```
1
```

[13]:
```
#boolean operators
True and False
```

[13]:
```
False
```

[14]:
```
True or False
```

[14]:
```
True
```

[15]:
```
#comparison operators
5>4,15==16,14!=14
```

[15]:
```
(True, False, False)
```

1.0.3 3) List

[16]:
```
#create a list
f=[1,4,7,8]
type(f)
```

[16]:
```
list
```

[17]:
```
#indexing starts from 0
f[0]
```

[17]: 1

[18]:
```
f[0:2]
```

[18]: [1, 4]

[20]:
```
f[3]
```

[20]: 8

[21]:
```
# list elements can have different types
g=['hello',5,6,8]
```

[22]:
```
print (g[0])
```

```
hello
```

[23]:
```
#nested list
h=[[1,2,3],[[5,6,7],'find me']]
```

[25]:
```
print (h[1][1])
```

```
find me
```

[26]:
```
#range function
i=list(range(1,10,1))
print (i)
```

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

[27]:
```
#append
i.append(10)
i
```

[27]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[28]:
```
#insert function
i.insert(1,'hello')
i
```

[28]: [1, 'hello', 2, 3, 4, 5, 6, 7, 8, 9, 10]

[29]:
```
i.insert(3,'hello')
i
```

[29]:
```
[1, 'hello', 2, 'hello', 3, 4, 5, 6, 7, 8, 9, 10]
```

[30]:
```
#remove function
i.remove('hello')
i
```

[30]:
```
[1, 2, 'hello', 3, 4, 5, 6, 7, 8, 9, 10]
```

[31]:
```
#del function
del i[1]
i
```

[31]:
```
[1, 'hello', 3, 4, 5, 6, 7, 8, 9, 10]
```

1.0.4 4) Tuples

[32]:
```
# like list but immutable (cannot be changed)
j=(5,8)
type(j)
```

[32]:
```
tuple
```

[33]:
```
#cannot be changed
j[1]=8
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-33-ca4b7af800fc> in <module>
      1 #cannot be changed
----> 2 j[1]=8


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

1.0.5 5) Sets

[34]:
```
#similar to sets in math
A={1,2,3}
B={3,4,5,2}
type(A)
```

[34]:
```
set
```

[35]:
```
#union and intersection
A|B,A&B
```

[35]:
```
({1, 2, 3, 4, 5}, {2, 3})
```

1.0.6 6) Dictionaries

[36]:
```
#each element has key and value
k={'parameter1':1,'parameter2':2}
type(k)
```

[36]:
```
dict
```

[37]:
```
print(k['parameter1'])
```

```
1
```

[38]:
```
k.items()
```

[38]:
```
dict_items([('parameter1', 1), ('parameter2', 2)])
```

1.0.7 7) Control flow

[41]:
```
#if statment
x=4
y=5

if x<y:
    print ('x is less than y')
elif y==x:
    print ('x is equal to y')
else:
    print('x is greater than y')
```

```
x is less than y
```

[42]:
```
# for loop
for i in [1,2,3,4]:
    print(i)
```

```
1
2
3
4
```

[43]:
```
for i in range(1,5,1):
    print(i)
```

```
1
2
3
4
```

[44]:
```
for key,value in k.items():
    print(key,"=",value)
```

```
parameter1 = 1
parameter2 = 2
```

[45]:
```
# use for loop to create a list
l=[x**2 for x in range(0,10)]
```

[46]:
```
print(l)
```

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

[47]:
```
#while loop
m=5
while m<10:
    print(m)
    m=m+1
```

```
5
6
7
8
9
```

1.0.8 8) Functions

[48]:
```
def func(input):
    output=2*input
    return(output)
```

[49]:
```
func(2)
```

[49]: 4

[50]:
```
#factorial 5!=5*4*3*2*1 , 0!=1
def factorial(number):
    if number<0:
        factorial="factorial doesn't exist"
    elif number==0:
        factorial=1
    else:
        factorial=1
```

```
        for n in range (1,number+1):
            factorial=factorial*n
    return (factorial)
```

[54]:
```
factorial(7)
```

[54]:
```
5040
```

[55]:
```
#time function
import time
```

[56]:
```
time.time()
```

[56]:
```
1585847391.509418
```

[57]:
```
def factorial(number):
    t1=time.time()
    if number<0:
        factorial="factorial doesn't exist"
    elif number==0:
        factorial=1
    else:
        factorial=1
        for n in range (1,number+1):
            factorial=factorial*n
    t2=time.time()
    #print('It took '+str(t2-t1)+' seconds')
    print ('It took {s} seconds'.format(s=t2-t1))
    return (factorial)
```

[58]:
```
factorial(5)
```

```
It took 2.86102294921875e-06 seconds
```

[58]:
```
120
```

[60]:
```
#lambda function
func=lambda input:input**2
```

[61]:
```
func(2)
```

[61]:
```
4
```

1.0.9 9) Classes

[62]:
```
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def translate(self,dx,dy):
        return (self.x+dx,self.y+dy)
```

[64]:
```
P1=Point(1,2)
```

[65]:
```
P1.translate(0.5,0.5)
```

[65]:
```
(1.5, 2.5)
```

[66]:
```
#write description for functions and classes

def factorial(number):
    """
    This function computes factorial of an integer

    """
    t1=time.time()
    if number<0:
        factorial="factorial doesn't exist"
    elif number==0:
        factorial=1
    else:
        factorial=1
        for n in range (1,number+1):
            factorial=factorial*n
    t2=time.time()
    #print('It took '+str(t2-t1)+' seconds')
    print ('It took {s} seconds'.format(s=t2-t1))
    return (factorial)
```

[67]:
```
#help function
help(factorial)
```

```
Help on function factorial in module __main__:

factorial(number)
    This function computes factorial of an integer
```

1.0.10 10)Numpy

[68]:
```
import numpy as np
import matplotlib.pylab as plt
```

[70]:
```
#create numpy array
o=np.array([5,6,7,8,9])
o
```

[70]:
```
array([5, 6, 7, 8, 9])
```

[71]:
```
p=np.array([[5,6],[7,8]])
p
```

[71]:
```
array([[5, 6],
       [7, 8]])
```

[72]:
```
type(o), type(p)
```

[72]:
```
(numpy.ndarray, numpy.ndarray)
```

[73]:
```
o.shape
```

[73]:
```
(5,)
```

[74]:
```
p.shape
```

[74]:
```
(2, 2)
```

[76]:
```
o.dtype
```

[76]:
```
dtype('int64')
```

[77]:
```
# you can define data type in numpy array
q=np.array([[5,6],[7,8]],dtype=complex)
```

[78]:
```
q.dtype
```

[78]:
```
dtype('complex128')
```

[79]:
```
print(q)
```

```
[[5.+0.j 6.+0.j]
 [7.+0.j 8.+0.j]]
```

[80]:
```
#arange function
r=np.arange(0,10,0.5)
r
```

[80]:
```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. , 5.5, 6. ,
       6.5, 7. , 7.5, 8. , 8.5, 9. , 9.5])
```

[83]:
```
#linspace and logspace functions
s=np.linspace(1,5,100)
s
```

[83]:
```
array([1.        , 1.04040404, 1.08080808, 1.12121212, 1.16161616,
       1.2020202 , 1.24242424, 1.28282828, 1.32323232, 1.36363636,
       1.4040404 , 1.44444444, 1.48484848, 1.52525253, 1.56565657,
       1.60606061, 1.64646465, 1.68686869, 1.72727273, 1.76767677,
       1.80808081, 1.84848485, 1.88888889, 1.92929293, 1.96969697,
       2.01010101, 2.05050505, 2.09090909, 2.13131313, 2.17171717,
       2.21212121, 2.25252525, 2.29292929, 2.33333333, 2.37373737,
       2.41414141, 2.45454545, 2.49494949, 2.53535354, 2.57575758,
       2.61616162, 2.65656566, 2.6969697 , 2.73737374, 2.77777778,
       2.81818182, 2.85858586, 2.8989899 , 2.93939394, 2.97979798,
       3.02020202, 3.06060606, 3.1010101 , 3.14141414, 3.18181818,
       3.22222222, 3.26262626, 3.3030303 , 3.34343434, 3.38383838,
       3.42424242, 3.46464646, 3.50505051, 3.54545455, 3.58585859,
       3.62626263, 3.66666667, 3.70707071, 3.74747475, 3.78787879,
       3.82828283, 3.86868687, 3.90909091, 3.94949495, 3.98989899,
       4.03030303, 4.07070707, 4.11111111, 4.15151515, 4.19191919,
       4.23232323, 4.27272727, 4.31313131, 4.35353535, 4.39393939,
       4.43434343, 4.47474747, 4.51515152, 4.55555556, 4.5959596 ,
       4.63636364, 4.67676768, 4.71717172, 4.75757576, 4.7979798 ,
       4.83838384, 4.87878788, 4.91919192, 4.95959596, 5.        ])
```

[85]:
```
t=np.logspace(1,4,10,base=10)
t
```

[85]:
```
array([   10.        ,    21.5443469 ,    46.41588834,   100.        ,
         215.443469  ,   464.15888336,  1000.        ,  2154.43469003,
        4641.58883361, 10000.        ])
```

[86]:
```
#create a random number
np.random.rand(5,5)
```

[86]:
```
array([[0.97584749, 0.37380866, 0.6293344 , 0.69708806, 0.1051006 ],
       [0.51163618, 0.71537599, 0.74723132, 0.21562285, 0.50293207],
       [0.18081238, 0.47681452, 0.41925117, 0.36282926, 0.32264727],
       [0.53563263, 0.0352868 , 0.49422084, 0.18154162, 0.21086871],
       [0.41581825, 0.7763615 , 0.86336103, 0.48429084, 0.91177883]])
```

[87]:
```
#diagonal matrix
np.diag([1,2,3,4])
```

[87]:
```
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
```

[91]:
```
#create a meshgrid
x1,y1=np.mgrid[0:5,0:5]
x1
```

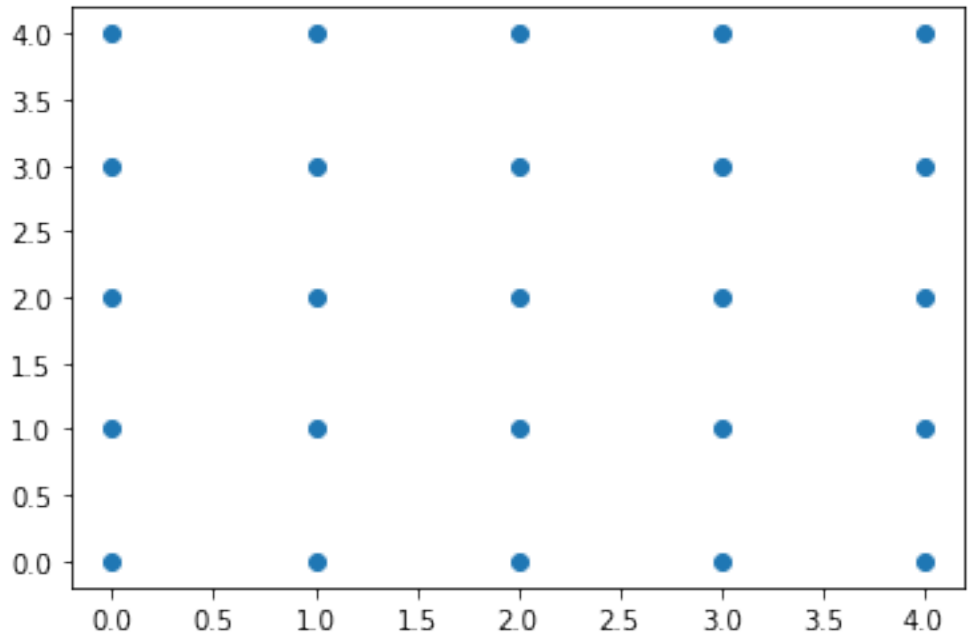
[91]:
```
array([[0, 0, 0, 0, 0],
       [1, 1, 1, 1, 1],
       [2, 2, 2, 2, 2],
       [3, 3, 3, 3, 3],
       [4, 4, 4, 4, 4]])
```

[93]:
```
y1
```

[93]:
```
array([[0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4]])
```

[94]:
```
plt.scatter(x1,y1)
```

[94]:
```
<matplotlib.collections.PathCollection at 0x11331aa60>
```



[95]:
```
#indexing in numpy
r
```

[95]:
```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. , 5.5, 6. ,
       6.5, 7. , 7.5, 8. , 8.5, 9. , 9.5])
```

[96]:
```
r[1]
```

[96]:
```
0.5
```

[97]:
```
r[3:6]
```

[97]:
```
array([1.5, 2. , 2.5])
```

[98]:
```
r[-1]
```

[98]:
```
9.5
```

[99]:
```
s=np.array([[5,6],[7,8]])
s
```

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

[101]:
```
s[1][1]
```

[101]:
```
8
```

[102]:
```
u=np.array([7,4,6,9,3,5])
```

[104]:
```
mask=(u>3) * (u<7)
mask
```

[104]:
```
array([False,  True,  True, False, False,  True])
```

[105]:
```
u[mask]
```

[105]:
```
array([4, 6, 5])
```

[106]:
```
#where function
indices=np.where(mask)
indices
```

[106]:
```
(array([1, 2, 5]),)
```

[107]:
```
u[indices]
```

[107]:
```
array([4, 6, 5])
```

[108]:
```
#take function
u.take([2,3,4])
```

[108]:
```
array([6, 9, 3])
```

[ ]:
```

```