Lab Instructions - session 0

Introduction to Python

Open an interactive Python environment (python shell, ipython shell, or Jupyter notebook), run the following commands, and observe the output:

Integer, float, string, and boolean types

```
>>> i = 1
>>> f = 1.1
>>> s = "salam!"
>>> t = True
>>> type(i)
>>> type(f)
>>> type(s)
>>> type(t)
```

Conversion between types

```
>>> str(i)
>>> str(f)
>>> int(f)
>>> int('123')
>>> float(i)
>>> float('1.1')
>>> int(t)
>>> int(False)
```

Basic operations

```
>>> a = 3
>>> b = 10
>>> a*b
>>> a+b
>>> b**a
>>>
>>> c,d = 1,a
>>> a,b = b,a
>>> a,b = a-b,a+b
>>> 5 / 2
>>> 5 / 2.0
>>> 5 // 2
```

```
>>> 5 // 2.0
>>>
>>> 5 % 2
>>>
>>> k = a**2+b**2
>>> print(a)
>>> a *= 10
>>> print(a)
>>>
>>> a -= 2
>>> print(a)
>>>
>>> a = 20
>>> b = 30
>>> a == b
>>> a < b
>>> a >= b - 5
>>>
>>> a < -5 or a > 5
>>> a < b and a == 20
>>> not (a < b and a == 20)
```

String operations

```
>>> s = 'salam'
>>> s2 = "salam"
>>> s3 = 'it's a nice day'
>>> s3 = "it's a nice day"
>>> s == s2
>>> len(s)
>>> r = '123'
>>> s+r
>>> s + ' ' + r
>>> s*8
>>>
>>> s = 'jenabkhan'
>>> s[0]
>>> s[1]
>>> s[2]
>>> s[-1]
>>> s[-3]
>>>
>>> s[1:4]
>>> s[2:]
>>> s[:4]
>>> s[:-1]
```

```
>>> s[1:7]
>>> s[1:7:2]
>>> i = 1
>>> d = 1.1
>>> s = "Salam"
>>>
>>> "num = %d"%i
>>> s1 = "%f %i %s"%(f,i,s)
>>> print(s1)
>>>
>>> "float=%f, integer=%i, and string=%s."%(d,i,s)
>>>
>>> "float={}, integer={}, and string={}.".format(d,i,s)
>>>
>>> f"float={d}, integer={i}, and string={s}."
>>>
>>> f"float={d*2}, integer={20-4*i}, and string={s+s}."
```

Tuples

```
>>> t = (1,2,4)
>>> p = (2, 'abc', 13.2)
>>> print(p)
>>>
>>> p[0]
>>> p[1]
>>> p[-1]
>>>
>>> len(p)
>>> p + (10,20,30)
>>> t+p
>>> p*3
>>> p[1] = 100 # error
>>> a,b,c = p
>>> print(a,b,c)
```

Lists

```
>>> l = [1,2,3,4]
>>> print(l)
>>>
>>> l = [i,f,s,2]
>>> l
>>>
>>> l = l + [True]
```

```
>>> l
>>>
>>> l = l + [1,3,45]
>>> l
>>>
>>> len(l)
>>>
>>> l.append('hi')
>>> l
>>>
>>> l.insert(0,100)
>>> l
>>> l.insert(2,111)
>>> l
>>>
>>> 111 in l
>>> 112 in l
>>> 112 not in l
>>>
>>> l.extend(['a', 'b', 'c'])
>>> l
>>>
>>> l.pop()
>>> l
>>>
>>> k = l.pop(2)
>>> l
>>> k
>>>
>>> l=[100,101,102,103,104,105,106,107,108,109,110,111,112]
>>>
>>> l[0]
>>> l[1]
>>> l[5]
>>> l[-1]
>>> l[-2]
>>> l[1:5]
>>> l[10:1]
>>> l[1:10]
>>> l[1:10:2]
>>> l[::3]
>>> l[10:1]
>>> l[10:1:-1]
>>> l[::-1]
>>>
>>> 104 in l
>>> 130 in l
>>>
```

```
>>> l = [4,1,7,2,0]
>>>
>>> l[4] = 100
>>>
>>> print(l)
>>> l[1] = 100
>>> print(l)
>>>
>>>
>>> l.reverse()
>>> print(l)
>>>
>>>
>>> l.sort()
>>> print(l)
>>>
>>>
>>> l = [4,1,7,2,0]
>>> t = l
>>>
>>> l[1] = 100
>>> print(l)
>>> print(t)
>>>
>>> l = [4,1,7,2,0]
>>> t = l[:]
>>>
>>> l[1] = 100
>>> print(l)
>>> print(t)
>>>
>>> e = enumerate(l)
>>> e
>>> list(e)
>>>
>>> z = zip(l,t)
>>> z
>>> list(z)
```

The range function

```
>>> r1 = range(10)
>>> r2 = range(2,10)
>>> r3 = range(2,20,3)
>>> r4 = range(20,2,-1)
>>>
>>> list(r1)
```

```
>>> list(r2)
>>> list(r3)
>>> list(r4)
>>> tuple(r1)
```

Dictionaries

```
>>> d = {1: 'Salam', 8: 1.4}
>>> d[1]
>>> d[8]
>>> d[2]
>>>
>>> d[2] = 444
>>> d
>>> d[2]
>>>
>>> d['K'] = 99
>>> d['Ali'] = 'passwd'
>>> d['Ali']
>>> print(d)
>>>
>>> d.keys()
>>> d.values()
>>> d.items()
>>>
>>>
>>>
>>> d
>>> del d['K']
>>> d
>>>
```

Open a text editor, or a Python IDE (IDLE editor, emacs, vs code, spyder, pyCharm, pyDev, etc.). Enter and run the following pieces of code.

Decision Making

```
i = 12
b = 1
if i == 12:
   b = 2
print(b)
```

```
i = 12

i = 12

if i == 10:
   print('YES')
   print('i equals 10')
else:
   print('NO')
   print('i is not equal to 10')
```

```
i = 11
b = 'salam'

if i == 12:
   print('Twelve')
elif i == 11 and b == 'hi':
   print('Eleven-hi')
elif i > 10 and b == 'salam':
   print('SALAAAMM!!')
else:
   print('None')
```

the While loop

```
i = 10
while i > 0:
    print(i*i)
    i -= 2
```

For loop

```
l = [10,20,30,40.2, 'salam']
for k in l:
      print(k)
```

```
for k in range(2,20):
      print(k, k*k)
```

```
p = [1,2,3]
q = ['One','Two','Three']
for i in range(len(p)):
      print(p[i],q[i])
```

```
l = [10,20,1.2, 'salam']
for k in l:
      print(k)

for i, k in enumerate(l):
      print(i,k)

l1 = [10,20,30, 'salam']
l2 = ['a','b','c', 'aleik']

for i,j in zip(l1,l2):
     print(i,j)

for k in range(2,20):
      print(k, k*k)
```

Functions

```
def add(a,b):
    return a+b
print(add(2,3))
print(add(2,3.1))
print(add('abc', '123'))
```

```
def sum(l):
    s = 0
    for k in l:
        s += k
    return s

l = [1,2,4,8,16]

print(sum(l))
print(sum(range(10))
print(sum(2)) # error
```

Default argument values

```
def add(a,b=1):
    return a+b
print(add(20,4))
print(add(20))
```

Call by name

```
def sub(n1,n2):
    return n1-n2

print(sub(20,4))
print(sub(n1=20,n2=4)
print(sub(n2=20,n1=4)
```

Using python modules

```
import math
print(math.cos(0), math.exp(0))
print(math.pi)
print(math.cos(math.pi))
```

```
import math as m
print(m.tan(0))
```

```
from math import sin, cos, exp, pi
print(exp(1), sin(pi/2))
```

```
from math import *
print(pi)
print(e)
print(log(e))
print(cos(log(1)))
print(tan(pi/4))
```

List Comprehensions

```
l = [10,20,30]
t = [2*k for k in l]
print(t)

t = [k*k for k in range(10)]
print(t)

t = [k*k for k in range(10) if k % 2 == 0]
print(t)
```

Lambda Functions

```
f = lambda x : 2*x+1

print(f(1))
print(f(2))
```

Task 1

One way to represent a matrix in Python is to use a list of the rows of a matrix (a list of lists). For example the matrix

$$\begin{bmatrix} 0 & 2 & -1 \\ -2 & 0 & -4 \\ 1 & 4 & 0 \end{bmatrix}$$

can be represented as:

```
A = [ [0 2 -1], [-2 0 -4], [1 4 0] ]
```

Your task is to write a function named `mul` that receives two matrices as arguments and returns their product as a matrix:

```
def mul(A,B):
    """
    returns the product of the matrix A by the matrix B
    returns the empty list [] if the matrix dimensions
    are not consistent for multiplication
    """
    # function body


# test the function
A = [[1, 0, 0],
     [0, 0, 3],
     [0, 2, 0]];

B = [[1, 1],
     [0, .5],
     [2, 1/3.0]];

C = [[ 1,  0,  0 ],
     [ 0,  0,  0.5],
     [ 0,  1/3.0, 0]]

print(mul(A,B))
print(mul(B,A))
print(mul(A,C))
```

- The arguments A and B must be of the above format (nested list). **Do not use numpy arrays or matrices.**
- Assume each matrix is consistent (rows are of the same size). But the dimensions might be inconsistent for multiplication in which case you need to return an empty list [ ].

The output of the code above is

```
[[1, 1.0], [6, 1.0], [0, 1.0]]
[]
[[1, 0.0, 0.0], [0, 1.0, 0.0], [0, 0.0, 1.0]]
```