786

Name: Muhammad Sherjeel Akhtar

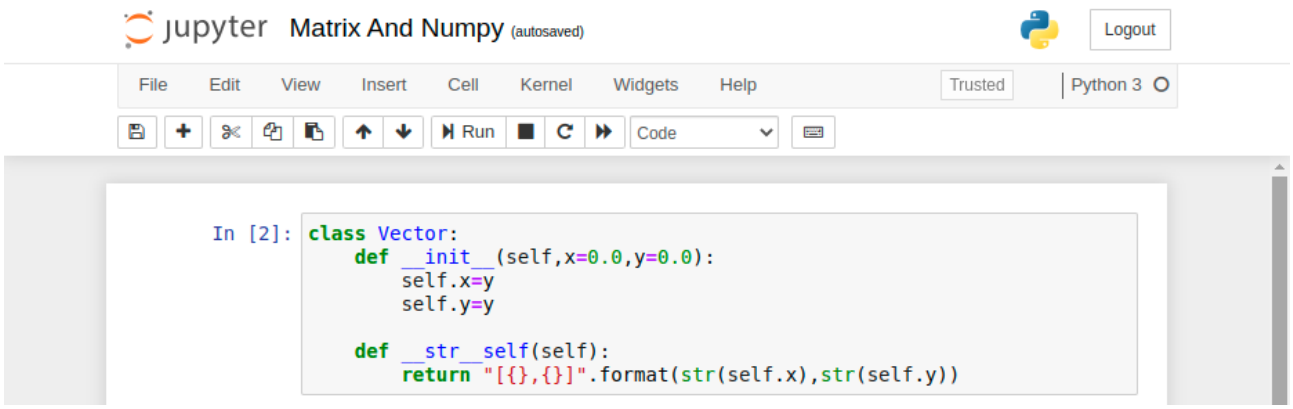
Roll No: 20p-0101

Subject: Numerical Computing

Assignment No: 02

Submitted **T**o **R**espect **S**ir: **Muhammad Nauman**

Section: 5A



In [36]:
```
a=Vector(2,4)
```

In [37]:
```
print(a)
```

```
<__main__.Vector object at 0x7ff784028820>
```

In [38]:
```
b=Vector(5,2)
print(b)
```

```
<__main__.Vector object at 0x7ff784028b80>
```

In [39]:
```
def add(self,b):
    c=Vector()
    c.x=self.x+b.x
    c.y=self.y+b.y
    return c

Vector.add=add
```

In [40]:
```
c=a.add(b)
print(c)
```

```
<__main__.Vector object at 0x7ff784028dc0>
```

In [41]:
```
def mul(self,s):
    return Vector(s*self.x,s*self.y)

Vector.mul=mul
```

In [42]:
```
d=a.mul(2)
print(d)
```

```
<__main__.Vector object at 0x7ff7840a4850>
```

In [43]:
```
def sub(self,b):
    #we use definition of vector subtraction
    # instead of defining something new

    return self.add(b.mul(-1))

Vector.sub=sub
```

In [44]:
```
d_min_b=d.sub(b)
print(d_min_b)
```

```
<__main__.Vector object at 0x7ff776ff9520>
```

.

In [68]:
```
class Matrix:
    def __init__(self,dims,fill):
        self.rows=dims[0]
        self.cols=dims[1]
        self.A=[
            [fill] * self.cols
            for i in range(self.rows)
        ]
```

In [69]:
```
m=Matrix((3,4),2.0)
```

In [70]:
```
print(m)
```

```
<__main__.Matrix object at 0x7ff776ff9610>
```

In [71]:
```
def __str__(self):
    rows=len(self.A) #Get the first dimension
    ret = ''

    for i in range(rows):
        cols=len(self.A[i])
        for j in range(cols):
            ret+=str(self.A[i][j])+"\t"
        ret+="\n"

    return ret

Matrix.__str__=__str__
```

.

In [49]:
```
print(m)
```

```
2.0     2.0     2.0     2.0
2.0     2.0     2.0     2.0
2.0     2.0     2.0     2.0
```

In [50]:
```
%time n = Matrix((100,100),0.0)
```

```
CPU times: user 70 µs, sys: 1 µs, total: 71 µs
Wall time: 74.4 µs
```

Memory Usage

In [51]:
```
from sys import getsizeof
print(getsizeof(m))
print(getsizeof(n))
```

```
48
48
```

.

In [ ]:
```
!pip install pympler
```

.

I was getting the error for PIP. This can be solved by simply typing the following command in the terminal:

“**sudo apt install python3-pip**”

```
spoofy@spoofy-Precision-M4600:~$ pip

Command 'pip' not found, but can be installed with:

sudo apt install python3-pip

spoofy@spoofy-Precision-M4600:~$ sudo apt install python3-pip
[sudo] password for spoofy:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libexpat1-dev libpython3-dev libpython3.8-dev python-pip-whl python3-dev
  python3-wheel python3.8-dev zlib1g-dev
The following NEW packages will be installed:
  libexpat1-dev libpython3-dev libpython3.8-dev python-pip-whl python3-dev
  python3-pip python3-wheel python3.8-dev zlib1g-dev
0 upgraded, 9 newly installed, 0 to remove and 126 not upgraded.
Need to get 6,805 kB of archives.
After this operation, 25.6 MB of additional disk space will be used.
```

.

After installing pip, run the “!pip install pympler” in the notebook. It will work fine.

```
In [52]: from pympler.asizeof import asizeof

In [53]: asizeof(m),asizeof(n)
Out[53]: (760, 86880)

In [54]: dim=5000

In [55]: %time m=Matrix((dim,dim),0.0)
CPU times: user 343 ms, sys: 48.2 ms, total: 391 ms
Wall time: 388 ms
```

.

In [ ]:
```
size=asizeof(m)/(1024*1024)
print("{:.2f} MBs.".format(size))
```

In [59]:
```
def get(self,i,j):
    if i<0 or i>self.rows:
        raise ValueError("Row index out of range.")
    if j<0 or j>self.cols:
        raise ValueError("Column index out of range.")


    return self.A[i][j]

Matrix.get=get
```

In [60]:
```
m.get(1,2)
```

Out[60]:
```
0.0
```

In [61]:
```
m.get(15,0)
```

Out[61]:
```
0.0
```

In [62]:
```
m.get(1,10)
```

Out[62]:
```
0.0
```

.

In [72]:
```
def get(self,i,j):
    if i<0 or i>self.rows:
        raise ValueError("Row index out of range.")
    if j<0 or j>self.cols:
        raise ValueError("Column index out of range.")

    if(i,j) in self.vals:
        return self.vals[(i,j)]

    return 0.0

Matrix.get=get
```

.

Sparse Method:

Sparse Method

In [78]:
```
class Matrix:
    def __init__(self,dims):
        self.rows=dims[0]
        self.cols=dims[1]
        self.vals={}
```

In [79]:
```
def set(self,i,j,val):
    self.vals[(i,j)]=val

Matrix.set=set
```

In [80]:
```
def get(self,i,j):
    if i<0 or i>self.rows:
        raise ValueError("Row index out of range.")
    if j<0 or j>self.cols:
        raise ValueError("Column index out of range.")

    if(i,j) not in self.vals:
        return 0

    return self.vals[(i,j)]
Matrix.get=get
```

- In [81]:
```
m=Matrix((5,5))
```

In [83]:
```
print(m.vals)
```

```
{}
```

- In [84]:
```
m.get(1,1)
```

Out[84]: 0

- In [85]:
```
m.get(10,2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-85-2353838e11b2> in <module>
----> 1 m.get(10,2)

<ipython-input-80-798f75e94ed8> in get(self, i, j)
      1 def get(self,i,j):
      2     if i<0 or i>self.rows:
----> 3         raise ValueError("Row index out of range.")
      4     if j<0 or j>self.cols:
      5         raise ValueError("Column index out of range.")

ValueError: Row index out of range.
```

In [ ]:

-

```
In [88]: m.set(1,2,15.0)

In [89]: m.get(1,2)
Out[89]: 15.0
```

-

```
In [90]: def get(self,i,j):
             if i<0 or i>self.rows:
                 raise ValueError("Row index out of range.")
             if j<0 or j>self.cols:
                 raise ValueError("Column index out of range.")

             if(i,j) not in self.vals:
                 return self.vals[(i,j)]

             return 0.0
         Matrix.get=get
```

-

```
In [100]: m=Matrix((5,5))

In [101]: print(m.vals)
          {}

In [102]: m.get(1,1)
Out[102]: 0.0
```

-

```
In [103]: m.get(10,2)
          ---------------------------------------------------------------------------
          -----
          ValueError                                Traceback (most recent call
          last)
          <ipython-input-103-2353838e11b2> in <module>
          ----> 1 m.get(10,2)

          <ipython-input-99-1fd39f03f6cd> in get(self, i, j)
                1 def get(self,i,j):
                2     if i<0 or i>self.rows:
          ----> 3         raise ValueError("Row index out of range.")
                4     if j<0 or j>self.cols:
                5         raise ValueError("Column index out of range.")

          ValueError: Row index out of range.
```

-

```
In [128]: m.set(1,2,15.0)

In [129]: m.get(1,2)
Out[129]: 15.0

In [130]: m.vals
Out[130]: {(1, 2): 15.0}

In [131]: m.set(1,4,29.9)

In [133]: m.get(1,4)
Out[133]: 29.9
```

.

```
In [134]: dim=500
          m=Matrix((dim,dim))

In [135]: asizeof(m)
Out[135]: 416

In [ ]:
```

.

```
In [3]: %matplotlib inline
```

.

This command gave some error to me, therefore I installed the “matplotlib” through the terminal.

Use the following command for this purpose:

“**pip install matplotlib**”

```
spoofy@spoofy-Precision-M4600:~$ pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.6.0-cp38-cp38-manylinux_2_12_x86_64.manylinux2010_x86
_64.whl (9.4 MB)
     |                                | 9.4 MB 107 kB/s
Requirement already satisfied: pillow>=6.2.0 in /usr/lib/python3/dist-packages (
from matplotlib) (7.0.0)
Collecting contourpy>=1.0.1
  Downloading contourpy-1.0.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_
64.whl (295 kB)
     |                                | 295 kB 131 kB/s
Collecting numpy>=1.19
  Downloading numpy-1.23.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.
whl (17.1 MB)
     |                                | 17.1 MB 338 kB/s
Collecting fonttools>=4.22.0
  Downloading fonttools-4.37.2-py3-none-any.whl (959 kB)
     |                                | 959 kB 363 kB/s
```

.

Now these both executed successfully.

```
In [ ]: asizeof(m)

In [ ]: %matplotlib inline
        %run mplimp.py

In [ ]:

In [ ]:
```

.

```
In [ ]: %matplotlib inline
        %run mplimp.py

In [30]: import numpy as np

In [31]: np.random.seed(1337)
```

.

Basics Of Matrices:

BASICS OF MATRICES:

```
In [32]: x=np.array([1,4,3])
         x
Out[32]: array([1, 4, 3])
```

.

```
In [33]: np.random.seed(1)
```

.

BASICS OF MATRICES:

```
In [46]: x=np.array([1,4,3])
         x
Out[46]: array([1, 4, 3])

In [48]: y=np.array([[1,4,3],
                     [9,2,7]])
         y
Out[48]: array([[1, 4, 3],
                [9, 2, 7]])

In [49]: x.shape
Out[49]: (3,)
```

.

```
In [49]: x.shape
Out[49]: (3,)

In [50]: y.shape
Out[50]: (2, 3)

In [51]: z=np.array([[1,4,3]])

In [52]: z.shape
Out[52]: (1, 3)
```

.

```
In [106]: z=np.arrange(1,2000,1)
          z[:10]
          ---------------------------------------------------------------------------
          AttributeError                            Traceback (most recent call last)
          <ipython-input-106-fd9a108ce422> in <module>
          ----> 1 z=np.arrange(1,2000,1)
                2 z[:10]

          ~/.local/lib/python3.8/site-packages/numpy/__init__.py in __getattr__(attr)
              309             return Tester
              310
          --> 311         raise AttributeError("module {!r} has no attribute "
              312                              "{!r}".format(__name__, attr))
              313

          AttributeError: module 'numpy' has no attribute 'arrange'
```

.

There is also this thing that this keyword is ‘np.arange’ not ‘np.arrange’.

```
In [107]: z=np.arange(1,2000,1)
          z[:10]
Out[107]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

.

```
In [105]: z.shape
Out[105]: (1, 3)
```

.

```
In [109]: z.shape
Out[109]: (1999,)
```

.

```
In [110]: z=np.arange(1,2000,1)
          z[:-10]

Out[110]: array([   1,    2,    3, ..., 1987, 1988, 1989])
```

```
In [109]: z.shape

Out[109]: (1999,)
```

.

```
In [112]: np.arange(0.5,3,0.5)

Out[112]: array([0.5, 1. , 1.5, 2. , 2.5])

In [113]: np.arange(0.5,10,1).shape

Out[113]: (10,)
```

.

```
In [114]: np.arange(0.5,10,1).reshape(5,2).shape

Out[114]: (5, 2)

In [115]: np.arange(0.5,10,1).reshape(5,3).shape
          ---------------------------------------------------------------------------
          -----
          ValueError                                Traceback (most recent call
          last)
          <ipython-input-115-84a01f6ea824> in <module>
          ----> 1 np.arange(0.5,10,1).reshape(5,3).shape

          ValueError: cannot reshape array of size 10 into shape (5,3)
```

.

```
In [116]: np.arange(0.5,20,1).reshape(5,3).shape
          ---------------------------------------------------------------------------
          -----
          ValueError                                Traceback (most recent call
          last)
          <ipython-input-116-e0337ce54c1e> in <module>
          ----> 1 np.arange(0.5,20,1).reshape(5,3).shape

          ValueError: cannot reshape array of size 20 into shape (5,3)
```

.

```
In [119]: np.linspace(3,9,10)
Out[119]: array([3.        , 3.66666667, 4.33333333, 5.        , 5.66666667,
                 6.33333333, 7.        , 7.66666667, 8.33333333, 9.        ])
```

```
In [120]: print(x)
          print(x[1])
          print(x[1:])

          [1 4 3]
          4
          [4 3]
```

```
In [121]: print(y)
          y[0,1]

          [[1 4 3]
           [9 2 7]]
Out[121]: 4
```

```
In [122]: print(y)
          y[0][1]

          [[1 4 3]
           [9 2 7]]
Out[122]: 4
```

```
In [123]: y[:,1]
Out[123]: array([4, 2])

In [124]: y[:,[1,2]]
Out[124]: array([[4, 3],
                 [2, 7]])

In [ ]:
```

```
In [125]: y[:,[1,2]].shape
Out[125]: (2, 2)
```

END.