In [1]:
```
#python version what we are using
import sys
()
```

Out[1]:
```
()
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```
#python problem and solution
#python print the calender year and months
import calendar
y=int(input('print the year'))
m=int(input('print the month'))
print(calendar.month(y,m))
```

In [ ]:
```
#write a python program print the multiline code
print("""please make me clear it what happen "after" this 'presentatio
n'""")
```

In [ ]:
```
#python date and time we are used
import datetime
now=datetime.datetime.now()
print("current date and time")
print(now.strftime("%Y-%m-%d %H:%M:%S"))
```

In [ ]:
```
#radious of a circle
from math import pi
r=int(input("here the number of the number"))
```

```
x=r*2*pi
print(x)
```

In [ ]:
```
#python reverse method
a=str(input("enter the first name"))
b=str(input("enter the second name"))
x=b+a
print(x)
```

In [ ]:
```
#list of python
list_color=['red','yellow','green','blue']
print(list_color[2])
```

In [ ]:
```
n=int(input('enter the input number here'))
n1=int("%s"%n)
n2=int("%s%s"%(n,n))
n3=int("%s%s%s"%(n,n,n))
print(n1+n2+n3)
```

In [ ]:
```
#python print the calender
import calendar
y=int(input('enter the year'))
m=int(input('enter the month'))
print(calendar.month(y,m))
```

In [ ]:
```
#python compare the two date
f_date=date(2020,2,1)
l_date=date(1996,2,1)
compare=f_date-l_date
print(compare.days)
```

In [ ]:
```
#write a python program to get the volume of r
r=6
pi=3.14
v=4/3*pi*r**3
print(v)
```

In [ ]:
```
#python if number is greater than 17
def number(n):
    if n<17:
        return 17-n;
    else:
        return (n-17)*2
print(number(14))
print(number(25))
```

In [ ]:
```
#given a new string from a new string where is has been added to the fr
ont
```

In [ ]:
```
#write a python program to write a program in n times
def my_number(str,n):
    result=""
    for i in range(n):
        result=result+str;
    return result
print(my_number('noman siddique jahed',2))
print(my_number('ok i am here please cheq my problem',3))
```

In [ ]:
```
#write a python program to define the whetheter number is odd or even i
n user access point
x=int(input("please give the number here"))
if x%2==0:
    print('the number is even')
else:
    print('the number is odd')
```

In [ ]:
```
def substirng(str,n):
    slen=2
    if slen>len(str):
        slen=len(str)
        substr=str[:slen]
    for i in range(n):
        result=""
        result=result+str;
```

```
    return result
print(substring('abcdef',3))
print(substring('a',2))
```

In [ ]:
```
#write a python program to define character is vowel or not
def is_vowel(n):
    vowel="aeiou"
    return n in vowel
print(is_vowel('n'))
print(is_vowel('o'))
```

In [ ]:
```
#write a python program whether value is existing in list or not
def my_function(data,n):
    for value in data:
        if value==n:
            return True
    return False
print(my_function([2,3,6,6,7],7))
print(my_function([1,3,6,6,8],9))
```

In [ ]:
```
#create a histogram of a given list of integer
def histogram()
```

Concatenate all elements in a list into a string and return it

In [ ]:
```
def  my_function(n):
    result="";
    for x in n:
        result+=str(x);
    return result;
print(my_function([5,2,1,4,10]))
```

write a python program to swap two varibale

In [ ]:
```
a=int(input('here the our first varibale'))
b=int(input('here the our second varibale'))
#before swaping
print(a,b)
a,b=b,a
#after swaping
print(a,b)
```

sorted three number

In [ ]:
```
#first three are user input
n1=int(input('this is first number'))
n2=int(input('this is second number'))
n3=int(input('this is third number'))
x1=min(n1,n2,n3)
x3=max(n1,n2,n3)
x2=(n1+n2+n3)-x1-x3;
print("number is sorted order is :",x1,x2,x3)
```

In [ ]:
```
n=22
for i in n:
    if i==1:
        print(i)
        break
    elif i%2==0:
        print(i)
    else:
        print(3*i+1)
```

In [ ]:
```
#print name age and details in different line
def my_details():
    name="noman siddique"
```

```
    age=25
    details="dhaka bangladesh"
```

In [ ]:
```
write down a python program to containing all flower name in a list whi
ch is not present in other list
colors=["red","yellow",""]
```

In [ ]:
```
#22count the number occured in the given list
def num_list(n):
    count=0
    for i in n:
        if i==5:
            count=count+1
    return count
print(num_list([2,5,6,9,3,5]))
print(num_list([1,5,4,4,6,9,5]))
```

In [ ]:
```
#23write a python program n copies of first two characater of a given s
tring
def main_string(str,n):
    flen=3
    if flen>len(str):
        flen=len(str)
    substring=str[:flen]#main term for our next topic
    result=""
    for i in range(n):
        result=result+substring#our result value would be abcabcabc
    return result #abcabcabc
print(main_string("abcd",3))
print(main_string("k",5))
```

In [ ]:
```
#python code to demonstrate the coding with array(),insert(),append()
import array
arr=array.array('i',[1,2,3])
#printing the original array
print('the new original array is: ',end="")
```

In [ ]:
```
import numpy as np

a=np.array([1,2,3,4])
print(a)
```

In [ ]:
```
import numpy as np
import time
a = np.random.rand(10000000)
b = np.random.rand(10000000)
tic = time.time()
c = np.dot(a,b)
toc = time.time()
print("vector version:"+str(100*(toc-tic))+'ms')

c=0
```

In [ ]:
```
import time
import numpy as np
a = np.random.rand(10000000)
b = np.random.rand(10000000)
c=0
tic=time.time()
for i in range(10000000):
    c+=a[i]*b[i]
toc=time.time()
print(c)
print(tic)
print(toc)
print("for loop:"+str(10000*(toc-tic))+'ms')
```

In [ ]:
```
import numpy as np
A=np.array([[56.0,0.0,4.4,68.0],
            [1.2,104.0,52.0,8.0],
            [1.8,135.0,99.0,0.9],
    ])
x=A.sum(axis=0)
print(x)
```

```
p=100*A/x.reshape(1,4)
print(p)
```

In [ ]:
```
#python new code for the deep learning
import numpy as np
a = np.random.randn(5)
print(a)
```

In [ ]:
```
print(a.shape)
```

In [ ]:
```
#suppose print the a transpose this is looking the same because this is
 one dimentional column vector
print(a.T)
```

In [ ]:
```
#if we print the a dot aT vector matrix this is print the multipication
 of the problem
print(np.dot(a,a.T))
```

In [ ]:
```
a=np.random.randn(5,1)
print(a)
```

In [ ]:
```
print(a.T)
```

In [ ]:
```
print(np.dot(a,a.T))
```

In [ ]:
```
a=np.random.randn(3,3)
b=np.random.randn(3,1)
c=a*b
print(c.shape)
```

In [ ]:
```
a = np.random.randn(12288, 150) # a.shape = (12288, 150)
b = np.random.randn(150, 45) # b.shape = (150, 45)
c = np.dot(a,b)
print(c.shape)
```

In [ ]:
```
import numpy as np
def sigmoid_gradient(x):

    s=(1/(1+np.exp(-x)))
    ds=s*(1-s)
    return ds
x=np.array([1,2,3])
sigmoid_gradient(x)
print ("sigmoid_derivative(x) = " + str(sigmoid_gradient(x)))
```

In [ ]:
```
color_list=["blue","red","yellow"]
print(color_list[0],color_list[-1])
```

In [ ]:
```
#display the exam schedule time
exam_st_time=(2,5,2020)
print("the  exam will start from:%i %i %i"%exam_st_time)
```

In [ ]:
```
#row and colum calculate in python
import numpy as np
df=np.array([[1,2,3,4],
             [2,5,4,8],
             [1,2,3,4]])
x=len(df)
print(x)
y=len(df[2])
print(y)
```

In [ ]:
```
import numpy as np
x=np.array([1,2,3,4,5,6,7,8,9,10,11,12])
y=x.reshape(3,4)
print(y)
```

In [ ]:
```
import time
x1 = [9,2,2,3,1,2,5]
x2 = [5,6,2,6,1,9,10]
tic=time.process_time()
print(len(x1))
```

```
print(tic)
dot=0
for i in range(len(x1)):
    dot+=x1[i]*x2[i]
toc=time.process_time()
print(toc)
print ("dot = " + str(dot) + "\n ----- Computation time = " + str(1000*
(toc - tic)) + "ms")

#when we are using vectorization this is time consuming for us
```

In [ ]:
```
import numpy as np
a=np.ones(4)
print(a)
b=np.linspace(-1, 2, 4)
print(b)
c=np.outer(a,b)
print(c)
```

In [ ]:
```
import numpy as np
a=np.linspace(-1,2,4)
print(a)
```

In [ ]:
```
import numpy as np
a=(3,4)
b=np.zeros(a)
print(b)
print(len(a))
```

In [ ]:
```
import time

x1 = [9, 2, 5, 0, 0, 7, 5, 0, 0, 0, 9, 2, 5, 0, 0]
x2 = [9, 2, 2, 9, 0, 9, 2, 5, 0, 0, 9, 2, 5, 0, 0]

### CLASSIC DOT PRODUCT OF VECTORS IMPLEMENTATION ###
tic = time.process_time()
dot = 0
for i in range(len(x1)):
```

```
    dot+= x1[i]*x2[i]
toc = time.process_time()
print ("dot = " + str(dot) + "\n ----- Computation time = " + str(1000*
(toc - tic)) + "ms")

### CLASSIC OUTER PRODUCT IMPLEMENTATION ###
tic = time.process_time()
outer = np.zeros((len(x1),len(x2))) # we create a len(x1)*len(x2) matri
x with only zeros
for i in range(len(x1)):
    for j in range(len(x2)):
        outer[i,j] = x1[i]*x2[j]
toc = time.process_time()
print ("outer = " + str(outer) + "\n ----- Computation time = " + str(1
000*(toc - tic)) + "ms")

### CLASSIC ELEMENTWISE IMPLEMENTATION ###
tic = time.process_time()
mul = np.zeros(len(x1))
for i in range(len(x1)):
    mul[i] = x1[i]*x2[i]
toc = time.process_time()
print ("elementwise multiplication = " + str(mul) + "\n ----- Computati
on time = " + str(1000*(toc - tic)) + "ms")

### CLASSIC GENERAL DOT PRODUCT IMPLEMENTATION ###
W = np.random.rand(3,len(x1)) # Random 3*len(x1) numpy array
tic = time.process_time()
gdot = np.zeros(W.shape[0])
for i in range(W.shape[0]):
    for j in range(len(x1)):
        gdot[i] += W[i,j]*x1[j]
toc = time.process_time()
print ("gdot = " + str(gdot) + "\n ----- Computation time = " + str(100
0*(toc - tic)) + "ms")
```

In [ ]:

```
x1 = [9, 2, 5, 0, 0, 7, 5, 0, 0, 0, 9, 2, 5, 0, 0]
x2 = [9, 2, 2, 9, 0, 9, 2, 5, 0, 0, 9, 2, 5, 0, 0]
```

```
### VECTORIZED DOT PRODUCT OF VECTORS ###
tic = time.process_time()
dot = np.dot(x1,x2)
toc = time.process_time()
print ("dot = " + str(dot) + "\n ----- Computation time = " + str(1000*
(toc - tic)) + "ms")

### VECTORIZED OUTER PRODUCT ###
tic = time.process_time()
outer = np.outer(x1,x2)
toc = time.process_time()
print ("outer = " + str(outer) + "\n ----- Computation time = " + str(1
000*(toc - tic)) + "ms")

### VECTORIZED ELEMENTWISE MULTIPLICATION ###
tic = time.process_time()
mul = np.multiply(x1,x2)
toc = time.process_time()
print ("elementwise multiplication = " + str(mul) + "\n ----- Computati
on time = " + str(1000*(toc - tic)) + "ms")

### VECTORIZED GENERAL DOT PRODUCT ###
tic = time.process_time()
dot = np.dot(W,x1)
toc = time.process_time()
print ("gdot = " + str(dot) + "\n ----- Computation time = " + str(1000
*(toc - tic)) + "ms")
```

In [1]:
```
#vectorised dot products
import numpy as np
x1=[1,2,3,4,6,6]
x2=[2,6,9,10,5,2]
tic=time.process_time()
print(tic)
dot=np.dot(x1,x2)
toc=time.process_time()
print(toc)
#differece between before and the after
print("dot "+ str(dot) +"\n-----computation the time= "+ str(1000*(toc-
```

```
tic))+'ms')
v=x1.shape[1]
print(v)
```

```
---------------------------------------------------------------------
----
NameError                                 Traceback (most recent call l
ast)
<ipython-input-1-bd748a392880> in <module>
      3 x1=[1,2,3,4,6,6]
      4 x2=[2,6,9,10,5,2]
----> 5 tic=time.process_time()
      6 print(tic)
      7 dot=np.dot(x1,x2)

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

In [ ]:
```
tic=time.process_time()
print(tic)
outer=np.outer(x1,x2)
toc=time.process_time()
print(toc)
print("outer "+ str (outer)+ "\n--total computation time is= "+ str(100
0*(toc-tic))+"ms")
```

In [ ]:
```
#vector element multiflication between two varibale
tic=time.process_time()
mul=np.multiply(x1,x2)
toc=time.process_time()
print("vector multiplication would be "+str(mul)+"\n--- total computati
on "+str(1000*(toc-tic))+"ms")
```

In [37]:
```
import numpy as np
x1=np.array([[[1,1,1,1],[2,2,2],[3,3,3]],
[[2,3,4],[1,2,3],[1,2,3]]])
print(x1.shape)
b=x1.reshape(x1.shape[0],-1).T
print(b)
```

```
(2, 3)
[[list([1, 1, 1, 1]) list([2, 3, 4])]
 [list([2, 2, 2]) list([1, 2, 3])]
 [list([3, 3, 3]) list([1, 2, 3])]]
```

In [35]:
```
import numpy as np
x1=np.array[1,2,3,4]
print(x1.shape)
```

```
---------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-35-a4696441bfd1> in <module>
      1 import numpy as np
----> 2 x1=np.array[1,2,3,4]
      3 print(x1.shape)

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

In [ ]:
```

```