pda-401

March 19, 2025

[1]:
```
a=90
b=90.8
print(a+b)
```

```
180.8
```

[3]:
```
def addition(a,b):
    return (a+b)
```

[5]:
```
addition(20,25)
```

[5]: 45

[17]:
```
def taxcal(s,t):
    tax=((t/100)*s)
    return tax
```

[23]:
```
taxcal(50000, 10)
```

[23]: 5000.0

[37]:
```
def salary(S):
    if S>0 and S<10000:
        return 0.05*S
    elif S>=10000 and S<50000:
        return 0.1*S
    elif S>=50000 and S<200000:
         return 0.15*S
    elif S>=200000:
        return 0.20*S
    else:
        print("Invalid")
```

[39]:
```
salary(60000)
```

```
Invalid
```

[42]:
```
salary(0)
```

[42]: 0.0

loops

[49]:
```
w=[67,45,23,34]
h=[34,44,49,23]
for i,j in zip(w,h):
    print(i/(j*j))
```

```
0.05795847750865052
0.02324380165289256
0.009579341940857976
0.06427221172022685
```

[53]:
```
w=[67,45,23,34]
h=[34,44,49,23]
for i in range(len(w)):
    print(w[i] /(h[i]*h[i]))
```

```
0.05795847750865052
0.02324380165289256
0.009579341940857976
0.06427221172022685
```

1 Numpy

[60]:
```
import numpy as np
ar1 = np.array([90,56,12,34])
ar2= np.array([78,45,55,67])
print(ar1+ar2)
```

```
[168 101  67 101]
```

[64]:
```
arr1=np.ones((2,3))
print(arr1)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

[68]:
```
arr1=np.eye(3)
print(arr1)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

[70]:
```
arr1=np.zeros((2,3))
print(arr1)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

[76]:
```
arr1=np.array([[3,4,5],[9,5,0]])
print(arr1)
print(np.ndim(arr1))
```

```
[[3 4 5]
 [9 5 0]]
2
```

[78]:
```
arr1=np.array([[3,4,5],[9,5,0]])
print(arr1)
print(np.shape(arr1))
```

```
[[3 4 5]
 [9 5 0]]
(2, 3)
```

[86]:
```
arr1=np.array([2,3,4,5,6,7,8,9])
arr1=arr1.reshape(4,2)
```

[88]:
```
arr1
```

[88]:
```
array([[2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

[90]:
```
arr1.resize(4,2)
```

[92]:
```
arr1
```

[92]:
```
array([[2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

[96]:
```
arr7=np.arange(7,701,7)
print(arr7)
print(type(arr7))
```

```
[  7  14  21  28  35  42  49  56  63  70  77  84  91  98 105 112 119 126
 133 140 147 154 161 168 175 182 189 196 203 210 217 224 231 238 245 252
 259 266 273 280 287 294 301 308 315 322 329 336 343 350 357 364 371 378
 385 392 399 406 413 420 427 434 441 448 455 462 469 476 483 490 497 504
 511 518 525 532 539 546 553 560 567 574 581 588 595 602 609 616 623 630
```

```
 637 644 651 658 665 672 679 686 693 700]
<class 'numpy.ndarray'>
```

[102]:
```
arr9=np.linspace(2,8,6) #it generates evenly spaced values between 2 and 8
print(arr9)
```

```
[2.  3.2 4.4 5.6 6.8 8. ]
```

[104]:
```
arr10 = np.array([[[1,2,3],[6,7,8]],[[4,5,2],[3,6,0]]])
print(arr10)
print(np.shape(arr10))
print(np.ndim(arr10))
```

```
[[[1 2 3]
  [6 7 8]]

 [[4 5 2]
  [3 6 0]]]
(2, 2, 3)
3
```

[21]:
```
import numpy as np
mat1=np.array([9,4,6,7]).reshape(2,2)
mat2=np.array([1,2,3,4]).reshape(2,2)
print("matrix 1\n:",mat1)
print("matrix 2\n:",mat2)
```

```
matrix 1
: [[9 4]
 [6 7]]
matrix 2
: [[1 2]
 [3 4]]
```

[6]:
```
print(mat1.dot(mat2))
```

```
[[21 34]
 [27 40]]
```

[14]:
```
print(mat1@mat2)
```

```
[[21 34]
 [27 40]]
```

[16]:
```
print(np.linalg.inv(mat1))
```

```
[[ 0.17948718 -0.1025641 ]
 [-0.15384615  0.23076923]]
```

Statistics

[19]:
```
arr11=np.array([90,45,34,26,23,12])
print(np.mean(arr11))
```

```
38.333333333333336
```

[23]:
```
arr11=np.array([90,45,34,26,23,12])
print(np.median(arr11))
```

```
30.0
```

[25]:
```
print(np.std(arr11))
```

```
25.210227201585386
```

[27]:
```
print(np.var(arr11))
```

```
635.5555555555557
```

Trignomentry

[30]:
```
print(np.pi)
```

```
3.141592653589793
```

[32]:
```
red=[90,30,45]
for i in red:
    print(np.sin(i))
```

```
0.8939966636005579
-0.9880316240928618
0.8509035245341184
```

[34]:
```
red=[90,30,45]
for i in red:
    print(np.cos(i))
```

```
-0.4480736161291701
0.15425144988758405
0.5253219888177297
```

[36]:
```
red=[90,30,45]
for i in red:
    print(np.tan(i))
```

```
-1.995200412208242
-6.405331196646276
1.6197751905438615
```

[44]:
```
import numpy as np
avg=[np.pi/4,np.pi/2,np.pi/3]
for i in avg:
    print(np.sin(i))
```

```
0.7071067811865476
1.0
0.8660254037844386
```

[46]:
```
print(np.hypot(6,8))
```

```
10.0
```

Arthamatic operations

[53]:
```
a=np.array([8,9,1])
b=np.array([2,5,8])
print(np.sum((a,b)))
```

```
33
```

[57]:
```
c=np.array([[1,2,3],[6,7,3],[9,1,6]])
print(np.cumsum(c,axis=0)) #colum vise sum
```

```
[[ 1  2  3]
 [ 7  9  6]
 [16 10 12]]
```

[59]:
```
c=np.array([[1,2,3],[6,7,3],[9,1,6]])
print(np.cumsum(c,axis=1)) #row vise sum
```

```
[[ 1  3  6]
 [ 6 13 16]
 [ 9 10 16]]
```

[63]:
```
print(np.cumprod(c))
```

```
[    1     2     6    36   252   756  6804  6804 40824]
```

[65]:
```
print(np.cumprod(c,axis=0))
```

```
[[ 1  2  3]
 [ 6 14  9]
 [54 14 54]]
```

[67]:
```
print(np.cumprod(c,axis=1))
```

```
[[  1   2   6]
 [  6  42 126]
```

```
 [  9   9  54]]
```

[71]:
```
s1=np.array([90,23,40,12])
s2=np.array([10,2,11,5])
print(np.mod(s1,s2))
```

```
[0 1 7 2]
```

[73]:
```
print(np.divmod(s1,s2))
```

```
(array([ 9, 11,  3,  2]), array([0, 1, 7, 2]))
```

Universal functions

[82]:
```
A=np.array([56,78,12,32,111,109,"kill"])
print(max(A))
```

```
kill
```

Sorting

[86]:
```
B=np.array([90,12,45,1,89,98])
B.sort()
print(B)
```

```
[ 1 12 45 89 90 98]
```

[90]:
```
c=np.array([90,12,45,1,89,98])
D=sorted(c)
print(c)
print(D)
```

```
[90 12 45  1 89 98]
[1, 12, 45, 89, 90, 98]
```

Ronunding

[104]:
```
s2=np.array([9.1,-7.8])
print(np.ceil(s2))
```

```
[10. -7.]
```

[106]:
```
print(np.floor(s2))
```

```
[ 9. -8.]
```

RANDOM MODULE

[122]:
```
import numpy.random as rd
```

[124]:
```
ran1=rd.rand(2)
print(ran1)
```

```
[0.33107757 0.10963255]
```

[139]:
```
ran2=rd.randint(5)
print(ran2)
```

```
0
```

[141]:
```
ran2=rd.randint(5,size=(6))
print(ran2)
```

```
[1 3 3 0 4 3]
```

[143]:
```
ran2=rd.randint(5,size=(6,2))
print(ran2)
```

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

[149]:
```
ran2=rd.randint(5,size=(6,2,3))
print(ran2)
```

```
[[[2 4 2]
  [4 0 4]]

 [[0 2 3]
  [1 0 1]]

 [[4 1 2]
  [3 1 3]]

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

 [[3 0 3]
  [1 4 2]]

 [[2 0 0]
  [1 4 2]]]
```

Stack

[154]:
```
Ar1=np.array([[9,4,23],[3,4,5]])
Ar2=np.array([[8,1,2],[33,42,51]])
Ar3=np.hstack(Ar1)
```

[157]:
```
print(Ar3)
```

```
[ 9  4 23  3  4  5]
```

[159]:
```
Ar1=np.array([[9,4,23],[3,4,5]])
Ar2=np.array([[8,1,2],[33,42,51]])
Ar3=np.vstack(Ar1)
```

[161]:
```
print(Ar3)
```

```
[[ 9  4 23]
 [ 3  4  5]]
```

[163]:
```
Ar1=np.array([[9,4,23],[3,4,5]])
Ar2=np.array([[8,1,2],[33,42,51]])
print(Ar1)
print("\n")
print(Ar2)
```

```
[[ 9  4 23]
 [ 3  4  5]]


[[ 8  1  2]
 [33 42 51]]
```

[165]:
```
Ar4=np.vstack((Ar1,Ar2)) #one top of another
print(Ar4)
```

```
[[ 9  4 23]
 [ 3  4  5]
 [ 8  1  2]
 [33 42 51]]
```

[167]:
```
Ar5=np.hstack((Ar1,Ar2)) #side by side
print(Ar5)
```

```
[[ 9  4 23  8  1  2]
 [ 3  4  5 33 42 51]]
```

[169]:
```
Ar6=np.dstack((Ar5))
print(Ar6)
```

```
[[[ 9  3]
  [ 4  4]
  [23  5]
  [ 8 33]
  [ 1 42]
  [ 2 51]]]
```

[171]:
```
Ar7=np.arange(1,13).reshape(3,2,2)
print(Ar7)
```

```
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
```

[173]:
```
Ar6=np.dstack((Ar7))
print(Ar6)
```

```
[[[ 1  5  9]
  [ 2  6 10]]

 [[ 3  7 11]
  [ 4  8 12]]]
```

[177]:
```
num1=81
print(np.sqrt(num1))
```

```
9.0
```

[179]:
```
num1=81
num2=99
print(np.gcd(num1,num2))
```

```
9
```

[181]:
```
num1=81
num2=99
print(np.lcm(num1,num2))
```

```
891
```

[183]:
```
AA=[45,67,89]
print(np.lcm.reduce(AA))
```

```
268335
```

[185]:
```
print(np.gcd.reduce(AA))
```

```
1
```

[187]:
```
AB=np.array([0,-5,23])
print(np.absolute(AB))
```

```
[ 0  5 23]
```

LOGIRTHMS

[190]:
```
n=45
print(np.log(n))
```

```
3.8066624897703196
```

[192]:
```
print(np.log10(n))
```

```
1.6532125137753437
```

[194]:
```
print(np.log2(n))
```

```
5.491853096329675
```

SET

[197]:
```
s1=np.array([9,3,5,2,1])
s2=np.array([4,5,2,1,3])
print(np.union1d(s1,s2))
```

```
[1 2 3 4 5 9]
```

[199]:
```
print(np.intersect1d(s1,s2))
```

```
[1 2 3 5]
```

[201]:
```
print(np.setdiff1d(s1,s2))
```

```
[9]
```

[203]:
```
col1=np.array([44,33,12,67,19])
index=np.where(col1%2 ==0)
print(index)
```

```
(array([0, 2], dtype=int64),)
```

[209]:
```
import numpy as np

col2 = np.array([45, 33, 21, 50, 60, 15])
index = np.where((col2 % 5 == 0) & (col2 % 3 == 0))
```

```
(array([0, 4, 5], dtype=int64),)
```

[211]:
```
import numpy as np
array_1d = np.arange(1, 10)
matrix_3x3 = array_1d.reshape(3, 3)

print(matrix_3x3)0
```

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

[213]:
```
import numpy as np
data = np.array([10, 20, 30, 40, 50])
mean_value = np.mean(data)
median_value = np.median(data)
std_deviation = np.std(data)

print(f"Mean: {mean_value}")
print(f"Median: {median_value}")
print(f"Standard Deviation: {std_deviation}")
```

```
Mean: 30.0
Median: 30.0
Standard Deviation: 14.142135623730951
```

[215]:
```
import numpy as np
array = np.array([10, 20, 20, 30, 30, 30, 40, 50, 50, 50, 50])
unique_elements, counts = np.unique(array, return_counts=True)
for element, count in zip(unique_elements, counts):
    print(f"Element {element} occurs {count} times")
```

```
Element 10 occurs 1 times
Element 20 occurs 2 times
Element 30 occurs 3 times
Element 40 occurs 1 times
Element 50 occurs 4 times
```

[217]:
```
import numpy as np
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])
vertical_stack = np.vstack((array1, array2))

print("Vertical Stack:")
print(vertical_stack)
```

```
Vertical Stack:
[[1 2 3]
```

```
 [4 5 6]]
```

[219]:
```
import numpy as np
arr = np.array([10, 60, 30, 80, 50])
arr[arr > 50] = 0

print(arr)
```

```
[10  0 30  0 50]
```

[221]:
```
import numpy as np
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])

result = np.matmul(matrix1, matrix2)


print(result)
```

```
[[19 22]
 [43 50]]
```

[223]:
```
import numpy as np

matrix = np.array([[4, 7], [2, 6]])

determinant = np.linalg.det(matrix)

if determinant != 0:
    inverse_matrix = np.linalg.inv(matrix)
    print("Inverse Matrix:")
    print(inverse_matrix)
else:
    print("The matrix is singular and does not have an inverse.")

print(f"Determinant: {determinant}")
```

```
Inverse Matrix:
[[ 0.6 -0.7]
 [-0.2  0.4]]
Determinant: 10.000000000000002
```

[225]:
```
import numpy as np

arr = np.array([1, 2, 3, 6, 7, 8, 4, 5])

mask = arr > 5
```

```
extracted_elements = arr[mask]

print("Extracted Elements:", extracted_elements)
```

```
Extracted Elements: [6 7 8]
```

[227]:
```
import numpy as np

array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])

result = array1 * array2

print(result)
```

```
[ 4 10 18]
```

[ ]:
```

```