OPEN SOURCE SOFTWARE LAB (15B17CI575)

Lab Assignment 4 (Practice Lab)

Odd 2021

Week 4: 21 Sept-27 Sept

Topic Coverage: Python-Numpy package

Patil Amit Gurusidhappa

19104004

B11

Practice Questions

Q1

import numpy as np

Q2

Z = np.arange(50)

# print(Z)

"""

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

 48 49]

"""

Q3

Z = Z[::-1]

# print(Z)

"""

[49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26

 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2

1  0]

"""

Q4

Z = np.arange(9).reshape(3, 3)

print(Z)

"""

[[0 1 2]

 [3 4 5]

 [6 7 8]]

"""

Q5

nz = np.nonzero([1, 2, 0, 0, 4, 0])

print(nz)

"""

(array([0, 1, 4], dtype=int64),)

"""

Q6

#Q6

Z = np.eye(3)

"""

[[1. 0. 0.]

 [0. 1. 0.]

 [0. 0. 1.]]

"""

Q7

# Q7

Z = np.random.random((3, 3, 3))

"""

  [0.35099052 0.8482969  0.05222678]

  [0.91164536 0.15826356 0.16321862]]

 [[0.57601763 0.40578076 0.25892916]

  [0.97008192 0.83436763 0.42780897]

  [0.70166076 0.53645433 0.97831133]]

 [[0.692219   0.31182984 0.9122695 ]

  [0.88395433 0.68614103 0.47344483]

  [0.8201882  0.91706861 0.3739778 ]]]

"""

Q8

# Q8

Z = np.random.random((10, 10))

Zmin, Zmax = Z.min(), Z.max()

print(Zmin, Zmax)

"""

0.007110739636577268 0.9972743242569733

"""

Q9

Z = np.random.random(30)

m = Z.mean()

# print(m)

"""

0.4357109767685749

"""

Q10

Z = np.ones((10, 10))

Z[1:-1, 1:-1] = 0

"""

[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]

 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]

"""

Q11

Z = np.ones((5,5))

Z = np.pad(Z, pad\_width=1, mode='constant', constant\_values=0)

# print(Z)

"""

[[0. 0. 0. 0. 0. 0. 0.]

 [0. 1. 1. 1. 1. 1. 0.]

 [0. 1. 1. 1. 1. 1. 0.]

 [0. 1. 1. 1. 1. 1. 0.]

 [0. 1. 1. 1. 1. 1. 0.]

 [0. 1. 1. 1. 1. 1. 0.]

 [0. 0. 0. 0. 0. 0. 0.]]

"""

Q12

Z = np.dot(np.ones((5, 3)), np.ones((3, 2)))

print(Z)

"""

[[3. 3.]

 [3. 3.]

 [3. 3.]

 [3. 3.]

 [3. 3.]]

"""

Q13

Z = np.arange(11)

Z[(3 < Z) & (Z <= 8)] \*= -1

# print(Z)

"""

[ 0  1  2  3 -4 -5 -6 -7 -8  9 10]

 """

Q14

Z1 = np.random.randint(0, 10, 10)

Z2 = np.random.randint(0, 10, 10)

print(np.intersect1d(Z1, Z2))

"""

[2 3 6 7 9]

"""

Q15

yesterday = np.datetime64('today', 'D') - np.timedelta64(1, 'D')

today = np.datetime64('today', 'D')

tomorrow = np.datetime64('today', 'D') + np.timedelta64(1, 'D')

print(yesterday)

print(today)

print(tomorrow)

"""

2021-09-23

2021-09-24

2021-09-25

"""

Q16

np.arange('2016-07', '2016-08', dtype='datetime64[D]')

print(Z)

"""

[ 0  1  2  3 -4 -5 -6 -7 -8  9 10]

"""

Q17

A = np.random.randint(0,2,5)

B = np.random.randint(0,2,5)

# Assuming identical shape of the arrays and a tolerance forthe

# comparison of values

equal = np.allclose(A,B)

print(equal)

# Checking both the shape and the element values, no tolerance

# (values have to be exactly equal)

equal = np.array\_equal(A,B)

print(equal)

"""

False

False

"""

Q18

Z = np.random.random(10)

Z[Z.argmax()] = 0

print(Z)

"""

[0.14803475 0.46704285 0.54750164 0.0.48769874 0.89002492

 0.48537243 0.20740634 0.43145922 0.90670777]

"""

**Exercise Questions:**

Q1.

import numpy as np

print(np.\_\_version\_\_)

"""

1.21.2

"""

import numpy as np

def characterfrequency(str):

    n = len(str)

    frequency = np.zeros(26, dtype=np.int64)

    for i in range(0, n):

        frequency[ord(str[i]) - ord('a')] += 1

    for i in range(0, n):

        if (frequency[ord(str[i]) - ord('a')] != 0):

            print(str[i], frequency[ord(str[i]) - ord('a')],

                  end=" ")

        frequency[ord(str[i]) - ord('a')] = 0

strr = "amit"

characterfrequency(strr)

"""

a 1 m 1 i 1 t 1

"""

Q2

def segregateZeroAndOne(arr, size):

    type0 = 0

    type1 = size - 1

    while(type0 < type1):

        if(arr[type0] == 0):

            (arr[type0],

             arr[type1]) = (arr[type1],

                            arr[type0])

            type1 -= 1

        else:

            type0 += 1

arr = [1, 0, 1, 0, 0, 1, 1]

arr\_size = len(arr)

segregateZeroAndOne(arr, arr\_size)

print("After segregating",

      end=" ")

for i in range(0, arr\_size):

    print(arr[i], end="")

"""

After segregating 1111000

"""

Q3

def removeNthChar(string, index):

  str1 = string[:index]

  str2 = string[index+1:]

  str3 = str1+str2

  return str3

sentence = input("Enter any sentence: ")

index = int(input("Enter the index value: "))

print(removeNthChar(sentence, index))

"""

Enter any sentence: amit working on flutter

Enter the index value: 8

amit woring on flutter

"""

Q4

import numpy as np

Z = np.ones((3, 3))

Z = np.pad(Z, pad\_width=1, mode='constant', constant\_values=0)

print(Z)

"""

[[0. 0. 0. 0. 0.]

 [0. 1. 1. 1. 0.]

 [0. 1. 1. 1. 0.]

 [0. 1. 1. 1. 0.]

 [0. 0. 0. 0. 0.]]

"""

Q5

import numpy as np

a1 = np.array([0, 10, 20, 40, 60])

print("Arr 1: ", a1)

a2 = [0, 40]

print("Arr 2: ", a2)

print("Comparing arr 1 with arr 2")

print(np.in1d(a1, a2))

"""

[ True False False True False]

"""

Q6

import numpy as np

array1 = np.array([0, 10, 20, 40, 60, 80])

print(array1)

array2 = [10, 30, 40, 50, 70]

print(array2)

print("unique Sorted araays")

print(np.setxor1d(array1, array2))

"""

[ 0 10 20 40 60 80]

[10, 30, 40, 50, 70]

unique Sorted araays

[ 0 20 30 50 60 70 80]

"""

Q7

import numpy as np

array = np.array((1, 2, 3))

print(array)

out\_arr = np.column\_stack((array))

print("Stacked Array ", out\_arr)

"""

[1 2 3]

Stacked Array  [[1 2 3]]

"""

Q8

import numpy as np

n\_array = np.array([[50, 30], [30, 40]])

print("Matrix")

print(n\_array)

det = np.linalg.det(n\_array)

rank = np.linalg.matrix\_rank(n\_array)

trace = np.matrix.trace(n\_array)

print("Rank")

print(int(rank))

print("Trace")

print(int(trace))

print("Determinant")

print(int(det))

"""

Matrix

[[50 30]

 [30 40]]

Rank

2

Trace

90

Determinant

1100

"""