Readiness Quiz
Graded Quiz • 45 min

Due Apr 27, 2:59 AM EDT

✓ **Congratulations! You passed!**

TO PASS 50% or higher

Keep Learning

Retake the assignment in **7h 52m**

GRADE
100%

Readiness Quiz

LATEST SUBMISSION GRADE

100%

1. For a given input list: 1, 2, 3, 4

1 / 1 point

1. Cube (element to power 3) each element

2. Return the results as a list

```
def get_cubed(lst):
  '''
  INPUT: LIST (containing numeric elements)
  OUTPUT: LIST (cubed value of each even number in originals list)
  return a list containing each element cubed
  '''
  return [_**2 for _ in lst]

```

Run

Reset

✓ **Correct**

Good Job!

2. For a given input list: 1,2,3,4,5,6,7

1 / 1 point

1. Inspect each number in the input list and determine if it is even

2. Next square the even values

3. Finally return the results as a list

```
def get_squared_evens(lst):
    '''
    INPUT: LIST (containing numeric elements)
    OUTPUT: LIST (squared value of each even number in originals list)
    return squared evens in a list
    '''
    even = [_  for _ in lst if _ % 2 == 0]
    return [_**2 for _ in even]

```

Run

Reset

✓ **Correct**

Good job!

3. Which of the following **are not** an example of a native or built-in data type in Python

1 / 1 point

□ boolean

□ integer

□ float

☑ heap

✓ **Correct**

Correct!

□ string

☑ varchar

✓ **Correct**

Correct!

4. For given input lists: a,b,c and 1,2,3

1 / 1 point

Create a dictionary from two input lists

```
def make_dict(lst1,lst2):
    '''
    INPUT: LST1, LST2
    OUTPUT: DICT (LST 1 are the keys and LST2 are the values)
    Given equal length lists create a dictionary where the first list is the keys
    '''
    return {k:val for k, val in zip(lst1, lst2)}

```

Run

Reset

✓ **Correct**

Good job!

5. Mutable data types/collections in Python can be changed in place. Immutable ones can not change in place. Which of the following are mutable?

1 / 1 point

□ bool

□ int

□ float

☑ set

✓ **Correct**

Correct!

☑ list

✓ **Correct**

Correct!

□ string

□ tuple

□ complex

6. Python is a general-purpose language, but which (1 or more) of the following ideas is not realistic with Python?

1 / 1 point

Python makes it easy to:

□ to prototype and explore data using Ipython and Jupyter notebooks

□ carry out unit testing

□ include comments/pseudocode to better organize code

☑ naturally parallelizes across cores and machines with little to no overhead

✓ **Correct**

Correct!

□ save files with an editor then subsequently execute them from the command line

□ interactively work to test and understand algorithms

□ save multiple functions in a file then import those functions from a different file

7. For a given input list: abbcccddddeeeeeffffffggggggghhhhhhhh

1 / 1 point

Return a dictionary of character counts

1. Count the of the number of times each character appears in the string
2. Characters with a count of 0 should not be included in the output dictionary

```
from collections import Counter
def count_characters(string):
    '''
    INPUT: STRING
    OUTPUT: DICT (with counts of each character in input string)

    Return a dictionary of character counts
    '''
    count = Counter(string)
    return {key:val for key, val in count.items() if val > 0}
```

Run

Reset

✓ **Correct**

Good job!

8. For the vector v = [2.0, -3.5, 5.1]:

1 / 1 point

1. Find the L1 norm of v

2. Return the result as a float

```
import numpy as np

def calculate_l1_norm(v):
  '''
  INPUT: LIST or ARRAY (containing numeric elements)
  OUTPUT: FLOAT (L1 norm of v)
  calculate and return a norm for a given vector
  '''
  return sum(list(map(abs,v)))
```

Run

Reset

✓ **Correct**

Good job!

9. NumPy array practice

1 / 1 point

1. Create a vector that starts at 1 and increases until 150

2. Turn the vector into a matrix with 10, rows and 15 columns

3. Return the sum for the 10 rows. HINT: there should be ten values for the printed sum

Use the following input vector values: vectorLower = 1; vectorUpper = 151

```
import numpy as np

def get_vector_sum(vectorLower, vectorUpper):
  '''
  INPUT: vector lower and upper bounds
  OUTPUT: calculated value for vector sum
  (1) create a vector ranging from 1:150
  (2) transform the vector into a matrix with 10 rows and 15 columns
  (3) print the sum for the 10 rows
  '''
  return np.arange(1,151).reshape((10,15))[:10,:].sum(axis = 1)
```

Run

Reset

✓ **Correct**

Good job!

10. Which of the following pairs of events are **mutually exclusive**. There can be more than one answer.

1 / 1 point

☐ Odd numbers and the number 3

☐ Even numbers and numbers greater than 10

☑ Negative numbers and positive numbers less than 25

✓ **Correct**

Correct!

☑ Numbers between 100-200 and numbers between 201-300

✓ **Correct**

Correct!

□ None of the above

11. Geometric distribution

1 / 1 point

The geometric distribution is a useful tool for modeling time to event data. A successful street vendor says that on average 1 out of every 10 people who walk by on the street stop to buy a taco.

1. Represent these data with a geometric distribution
2. What is the probability that the vendor *has to wait* until 20 people walk buy before someone buys a taco?

```
import scipy.stats as stats

def geometric_distribution(p,k):
    '''
    INPUT: probability of success and trials
    OUTPUT: determined probability
    '''
    return stats.geom.pmf(20, 1/10)


```

Run

Reset

✓ **Correct**

Good job!

12. Poisson distribution

1 / 1 point

The Poisson distribution is a useful tool for modeling count data given discrete intervals. Based on historical data the expected number of accidents at a busy intersection is 4 per month.

1. Represent these data with a Poisson distribution
2. What is the probability of more than 7 accidents at that intersection next month?

```
import scipy.stats as stats

def poisson_distribution(k1,k2):
    '''
    INPUT: probability of event interval
    OUTPUT: determined probability
    '''
    return stats.poisson.pmf(k2, k1)

```

Run

Reset

✓ **Correct**

Good job!

13. Gaussian distribution

1 / 1 point

The Gaussian or Normal distribution is use heavily throughout statistics and data science. Lets assume scores for this assessment have a mean of 50% and a standard deviation of 10.

1. Represent these data with a Normal distribution
2. What is the probability of observing a score >= 80?

Use 50.0, 20.0, and 80 for your input values

```
import scipy.stats as stats

def gaussian_distribution(loc_val, scale_val, cdf_val):
      '''
      INPUT: loc, scale, and cdf values
      OUTPUT: determined probability
      '''
      return 1-stats.norm.cdf(80,50.0,20.0)


```

Run

Reset

✓ **Correct**

Good job!

14. Which statement(s) about Pearson correlation and cosine similarity are true?

1 / 1 point

□ The dot product of two sample vectors divided by the product of their norms yields the correlation coefficient

☑ The cosine similairity of two centered vectors yields the correlation coefficient

✓ **Correct**

Correct!

□ The product of two sample vector norms that have been centered yields the correlation coefficient

☑ The dot products of centered vectors divided by the product of their norms yields the correlation coefficient

✓ **Correct**

Correct!

□ Two normalized sample vectors divided by centered dot products yield the correlation coefficient

□ The cosine similarity of the dot product of two sample vectors is the correlation coefficient

15. Perform matrix multiplication on a square matrix HINT: A 2X2 matrix times a 2x2 matrix should yield a 2x2 matrix

1 / 1 point

```
def matrix_multiplication(A,B):
  '''
  INPUT: LIST (of length n) OF LIST (of length n) OF INTEGERS
  LIST (of length n) OF LIST (of length n) OF INTEGERS
  OUTPUT: LIST OF LIST OF INTEGERS
  (storing the product of a matrix multiplication operation)
  Return the matrix which is the product of matrix A and matrix B
  where A and B will be (a) integer valued (b) square matrices
  (c) of size n-by-n (d) encoded as lists of lists, e.g.
  A = [[2, 3, 4], [6, 4, 2], [-1, 2, 0]] corresponds to the matrix
  | 2 3 4 |
  | 6 4 2 |
  |-1 2 0 |
  You may not use numpy. Write your solution in straight python
  '''
  res = [[None for _ in range(len(B[0]))] for _ in range(len(A))]

  for i in range(len(A)):
    for j in range(len(B[0])):
      vec_b = [_[j] for _ in B]
      vec_a = A[i]
      res[i][j] = sum([a*b for a,b in zip(vec_a, vec_b)])

  return res
```

Run

Reset

✓ **Correct**

Good job!