Readiness Quiz

LATEST SUBMISSION GRADE

80%

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
  '''
  cube_lst = list(map(lambda x: x ** 3, lst))
  return cube_lst
lst = [1, 2, 3, 4]
get_cubed(lst)

```

Run

Reset

```
[1, 8, 27, 64]
```

✓ **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
    '''
    evens = list(filter(lambda x: (x%2 == 0) , lst))
    squared_evens = list(map(lambda x: x ** 2, evens))
    return squared_evens
lst = [1, 2, 3, 4, 5, 6, 7]
get_squared_evens(lst)


```

Run

Reset

```
[4, 16, 36]
```

✓ **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
      keys
    '''
    res = dict(zip(lst1, lst2))
    return res

lst1 = ['a', 'b', 'c']
lst2 = [1, 2, 3]
make_dict(lst1, lst2)
```

Run

Reset

```
{'a': 1, 'b': 2, 'c': 3}
```

✓ **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:

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

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

□ to prototype and explore data using Ipython and Jupyter notebooks

□ carry out unit testing

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

✓ **Correct**

Correct!

□ include comments/pseudocode to better organize code

□ interactively work to test and understand algorithms

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

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

    Return a dictionary of character counts
    '''
    res = {}
    for keys in string:
      res[keys] = res.get(keys, 0) + 1
    return res
string = 'abbcccddddeeeeeffffffggggggghhhhhhhh'
count_characters(string)



```

Run

Reset

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8}
```

✓ **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

```python
import numpy as np
from numpy import array
from numpy.linalg import norm

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
  '''
  l1 = norm(v, 1)
  return l1

v = array([2.0, -3.5, 5.1])
calculate_l1_norm(v)

```

Run

Reset

```
10.6
```

✓ **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

```python
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
  '''
  v = np.arange(vectorLower, vectorUpper)
  v = v.reshape(10, 15)
  return np.sum(v, axis=1)

vectorLower = 1
vectorUpper = 151
get_vector_sum(vectorLower, vectorUpper)
```

Run

Reset

```
[ 120  345  570  795 1020 1245 1470 1695 1920 2145]
```

✓ **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

0 / 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
from scipy.stats import geom

def geometric_distribution(p,k):
  '''
  INPUT: probability of success and trials
  OUTPUT: determined probability
  '''
  r = geom.rvs(p, size=1000)
  f = (1-p) ** (k-1) * p
  rv = geom(p)
  prob = geom.cdf(k, p)
  return prob

p = 0.1
k = 20
geometric_distribution(p, k)
```

Run

Reset

```
0.878423345409
```

! **Incorrect**

Try again!

12. Poisson distribution

0 / 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
from scipy.stats import poisson

def poisson_distribution(k1,k2):
  '''
  INPUT: probability of event interval
  OUTPUT: determined probability
  '''
  rv = poisson(k1)
  prob = poisson.cdf(k2, k1)
  return prob
k1 = 4
k2 = 7
poisson_distribution(k1, k2)

```

Run

Reset

```
0.948866384207
```

! **Incorrect**

Try again!

Gaussian distribution

13. Gaussian distribution

0 / 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
from scipy.stats import norm

def gaussian_distribution(loc_val, scale_val, cdf_val):
    '''
    INPUT: loc, scale, and cdf values
    OUTPUT: determined probability
    '''
    #s = stats.norm(loc_val, scale_val)
    #prob = norm.cdf(cdf_val, s)
    return stats.norm.cdf(cdf_val,loc_val, scale_val)
loc_val =50
scale_val = 10 # mean and standard deviation
cdf_val = 80
gaussian_distribution(loc_val, scale_val, cdf_val)
```

Run

Reset

```
0.998650101968
```

! **Incorrect**

Try again!

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

```
import math
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 = [[0 for x in range(3)] for y in range(3)]
  for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            # resulted matrix
            res[i][j] += A[i][k] * B[k][j]
  return res
A = [[2, 3, 4], [6, 4, 2], [-1, 2, 0]]
B = [[2, 3, 4], [6, 4, 2], [-1, 2, 0]]
matrix_multiplication(A,B)
```

Run

Reset

```
[[18, 26, 14], [34, 38, 32], [10, 5, 0]]
```

✓ **Correct**

Good job!