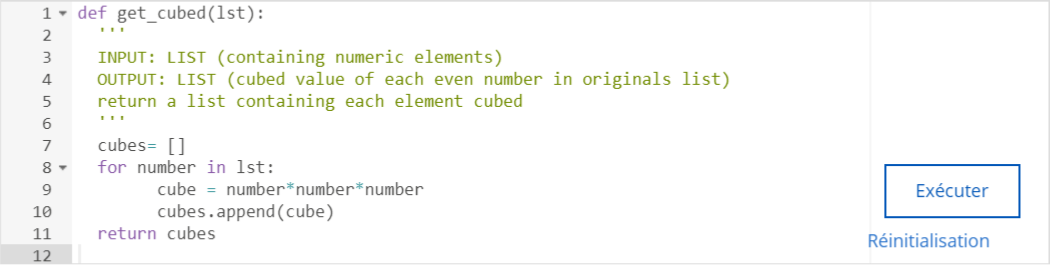
ibm-ai-workflow-business-priorities-data-ingestion: Readiness Quiz

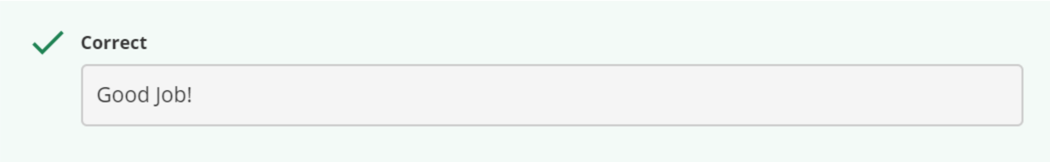
1. **For a given input list: 1, 2, 3, 4**

   1. Cube (element to power 3) each element

   2. Return the results as a list



```python
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
  '''
  cubes= []
  for number in lst:
        cube = number*number*number
        cubes.append(cube)
  return cubes

```



Correct

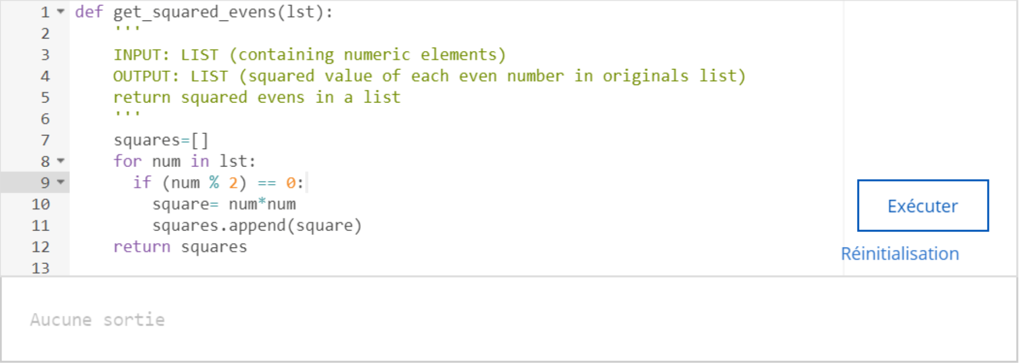
Good Job!

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

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

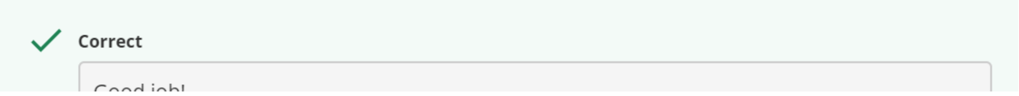
   2. Next square the even values

   3. Finally return the results as a list



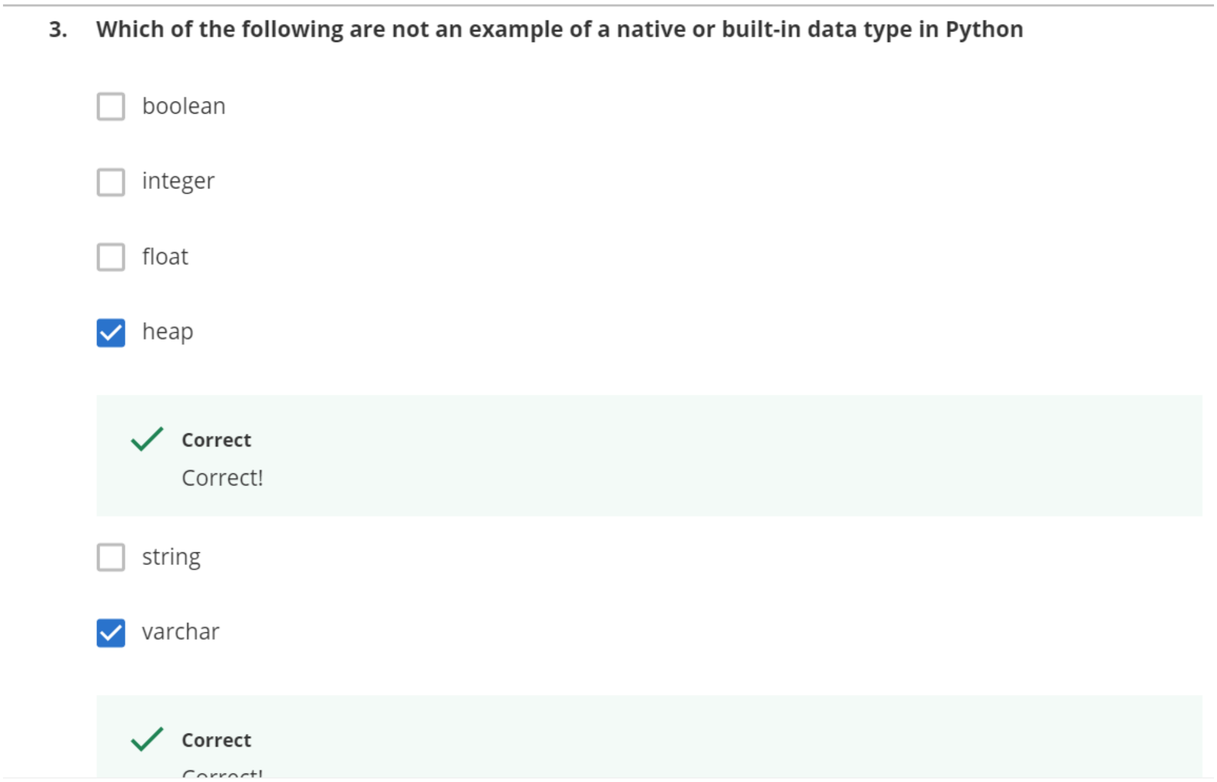
```python
def get_squared_evens(lst):
    '''
    INPUT: LIST (containing numeric elements)
    OUTPUT: LIST (squared value of each even number in originals list)
    return squared evens in a list
    '''
    squares=[]
    for num in lst:
      if (num % 2) == 0:
        square= num*num
        squares.append(square)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

```

```
Aucune sortie
```



Correct

Good job!



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

□ boolean

□ integer

□ float

☑ heap

✓ **Correct**

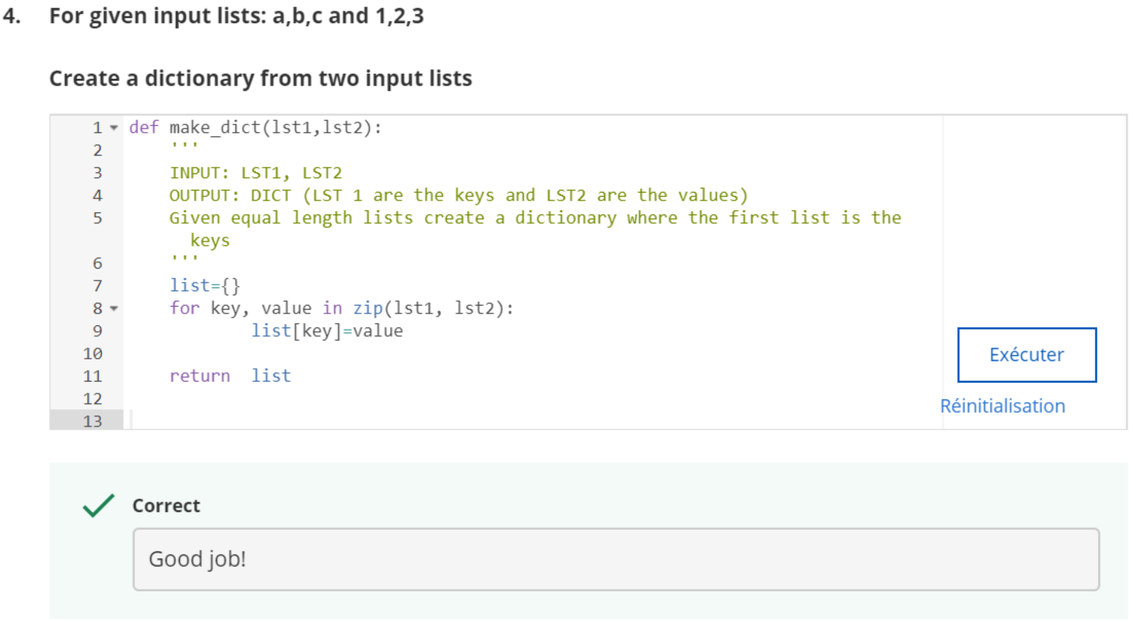
Correct!

□ string

☑ varchar

✓ **Correct**

Correct!



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

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
    '''
    list={}
    for key, value in zip(lst1, lst2):
            list[key]=value

    return  list

```

✓ **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?**

☐ bool

☐ int

☐ float

☑ set

✓ **Correct**

Correct!

☑ list

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

Python makes it easy to:

☐ carry out unit testing

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

✓ **Correct**

Correct!

☐ interactively work to test and understand algorithms

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

☐ to prototype and explore data using Ipython and Jupyter notebooks

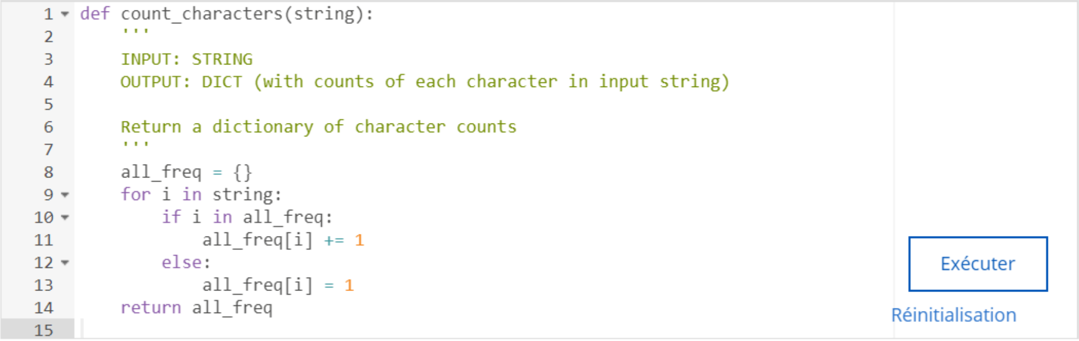
☐ include comments/pseudocode to better organize code

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

7. **For a given input list: abbcccddddeeeeeffffffggggggghhhhhhhh**

Return a dictionary of character counts

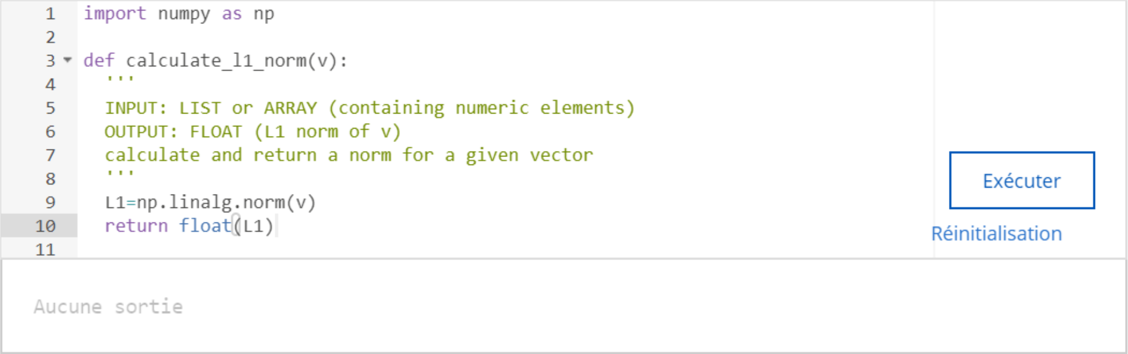
1. **Count the of the number of times each character appears in the string**
2. **Characters with a count of 0 should not be included in the output dictionary**



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

    Return a dictionary of character counts
    '''
    all_freq = {}
    for i in string:
        if i in all_freq:
            all_freq[i] += 1
        else:
            all_freq[i] = 1
    return all_freq

```

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

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
  '''
  L1=np.linalg.norm(v)
  return float(L1)

```

```
Aucune sortie
```

! **Incorrect**

Try again!

9. **NumPy array practice**

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

```

Exécuter

Réinitialisation

✓ **Correct**

Good job!

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

☐ 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**

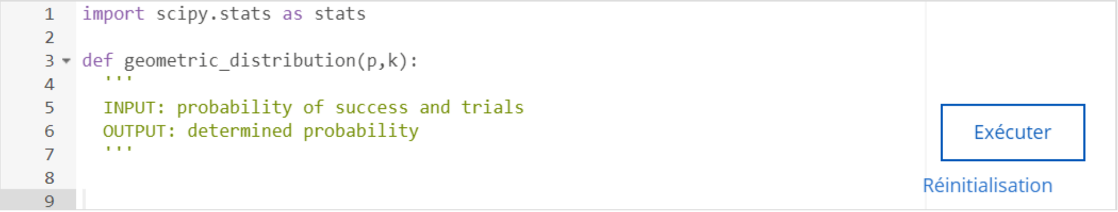
Correct!

☐ None of the above

11. Geometric distribution

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?**



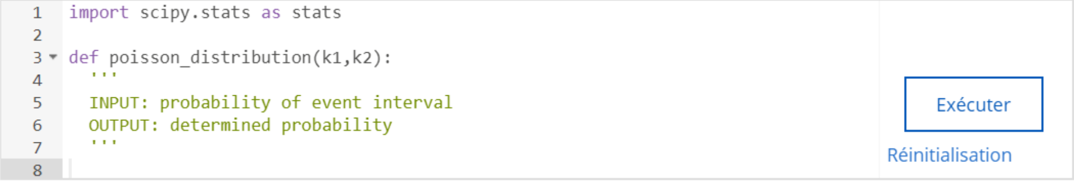
```
import scipy.stats as stats

def geometric_distribution(p,k):
    '''
    INPUT: probability of success and trials
    OUTPUT: determined probability
    '''

```

! **Incorrect**

12. Poisson distribution

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
    '''

```

! **Incorrect**

```
InfraError
```

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
    '''

```

Exécuter

Réinitialisation

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

□ 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

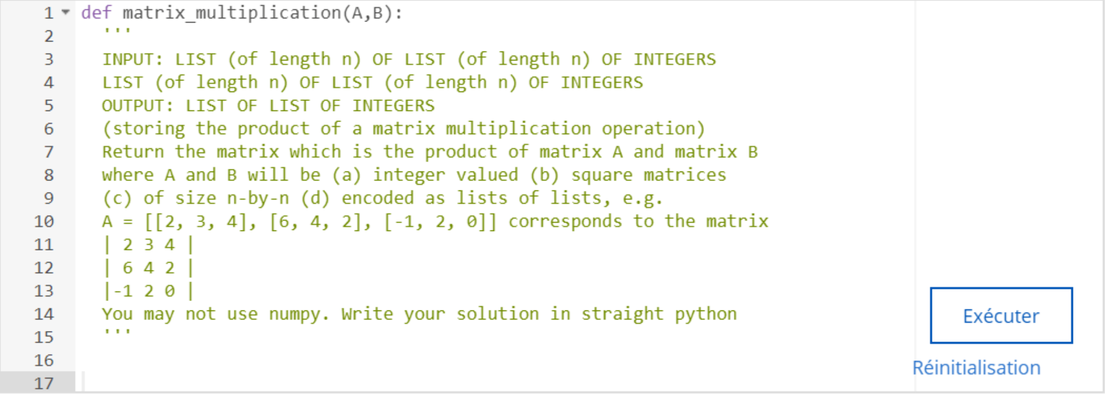
✓ **Correct**

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



```
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
  '''


```