

CIS 325: Programming for Business Analytics

Assignment 4 (Individual work)

Submission Instructions

- Submit your Python script according to the guidelines in this document as a .py file with the following naming convention: "A4-[ASURITE ID].py", where you will replace the text "[ASURITE ID]" with your ASURITE ID (= your ASU login ID).

For example, if your ASURITE ID is "abcd12", then your submission file name would be "A4-abcd12.py".

- On the top (header) of your submission .py file, add your name and email address as shown below:

```
""""
@author: [YOUR FULL NAME]
@email: [YOUR EMAIL ADDRESS]
""""
```

- Add Python comment statements (such as "*# Answer to Question 1*") in your submission .py file to separate your answers between questions.
- Not following the above submission instructions will result in a **0.2 point reduction** from your grade.

Question 1 (1.5 points)

Write a *generator* function that generates random integers that are between 1 and 40. For this, we will use the randint() function from the random package in order to generate a random integer.

- Search for the documentation of the randint() function from the random package. Then write a brief description of the randint() function as Python comments. Your description must contain (1) what randint() does, (2) syntax of randint(), and (3) what are the arguments (indicate required or optional) and what they do.

Starter code:

```
import random

### (WRITE YOUR DESCRIPTION OF THE randint() FUNCTION FROM THE random PACKAGE)

### Place your code below this line ###


### Place your code above this line ###

### (DIFFERENCE BETWEEN yield AND return)
```

Based on the starter code above, you need to do the following:

- Create a function named `lottery` that has one keyword argument named `n` with a default value of 6. The value of this keyword argument `n` will be the total number of random numbers to generate.
- Inside your new function, use a `for` loop statement that loops `n` times (you can use the `range()` function) and does the following at each iteration:
 - Use the `randint()` function to generate a random number between 1 and 40 (including both 1 and 40) and return it using a `yield` statement (instead of a `return` statement) so that your function becomes a generator function.
- In your own words, briefly describe (2-3 lines) how a generator function that uses a `yield` statement is different compared to a regular function that uses a `return` statement. Write your answer as Python comments.

Correctness Checks: If you were to run the following function calls, your output should be as follows:

(!!! **NOTE:** Actual numbers printed out to the console will be *different* each time you run your function since the numbers are randomly generated.)

```
>>> g = lottery(2)
>>> print(type(g))
<class 'generator'>

>>> x = next(g)
>>> print(x)
12

>>> x = next(g)
>>> print(x)
33

>>> a = list(lottery()) # generates 6 random numbers between 1 and 40
>>> print(a)
[27, 32, 11, 28, 7, 40]

>>> b = tuple(lottery(n=10)) # generates 10 random numbers between 1 and 40
>>> print(b)
(4, 12, 30, 1, 22, 20, 13, 26, 38, 39)

>>> for i in lottery(4):
...     print('Random number:', i)
Random number: 19
Random number: 29
Random number: 7
Random number: 31
```

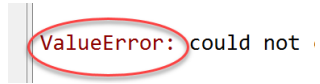
Question 2 (1 points)

Recall that a Python `ValueError` is raised when a built-in operation or function receives an argument that has the right type but an inappropriate value as seen here

<https://docs.python.org/3/library/exceptions.html#ValueError>.

Create a Python script that generates five different and unique examples of ValueError in Python.

Your script needs to generate five different and unique Python ValueError as seen in our lecture materials in the Python console with the following error categorization:


A screenshot of a Python console window. The text "ValueError: could not" is visible, with "ValueError:" circled in red.

Question 3 (1 points)

Recall that a Python TypeError is raised when an operation or function is applied to an object of an inappropriate type as seen here <https://docs.python.org/3/library/exceptions.html#TypeError>.

Create a Python script that generates five different and unique examples of TypeError in Python.

Your script needs to generate five different and unique Python TypeError as seen in our lecture materials in the Python console with the following error categorization:

A screenshot of a Python console window. The text "TypeError:" is visible, with "TypeError:" circled in red.

Question 4 (1.5 points)

You have been asked to improve the following function for your company that breaks tasks into modules. You are given the following script with a function and some quality assurance function calls for you to consider:

```
def break_modules(x,y):  
    return x/y  
  
print(break_modules(5,2))  
print(break_modules(7,0))  
print(break_modules(7,'ten'))
```

The code you have been given generates the following conditions:

1. `print(break_modules(5,2))` # when the first statement runs the function performs with no errors in Python;
2. `print(break_modules(7,0))` # when the second statement runs, the function generates a division by zero error in Python;
3. `print(break_modules(7,'ten'))` # When the third statement runs, the function generates a TypeError in Python.

You have been asked to use a Python try:except block to modify the function given to you to handle one specific error in a graceful manner. When a division by zero error occurs, your new function should return a

message that says, “All numbers must be greater than zero’. For all other errors, your new function should return a message that states, “Unknown error, please check your numbers”.

Thus, your output in the Python console should be similar to the following:

```
In [40]: print(break_modules(5,2))
2.5

In [41]: print(break_modules(7,0))
All numbers must be greater than zero

In [42]: print(break_modules(7,'ten'))
Unknown error, please check your numbers
```

Question 5 (1.5 points)

Create a Python class named `myfirstclass` with two methods named `methoda` and `methodb`. Your class, when instantiated needs to receive a number greater than one and this number will be used in your `methoda` and `methodb`. Your `methoda` simply needs to return the number passed during the initialization of your class raised to the power of 2. Your `methodb` simply needs to return the number passed during the initialization of your class raised to the power of 3.

You will know your class has been correctly implemented by using the following quality assurance steps using the number 4 and the name `myobj` for the instantiation of your class named `myfirstclass` as follows:

```
myobj = myfirstclass(4)
print(myobj.methoda()) # will output 16 to the console
print(myobj.methodb()) # will output 64 to the console
```

Question 6 (1.5 points)

Create a base class named `parentclass` with a method named `methoda`. When this base class is instantiated, the base class will need to receive a number greater than one that will be used by your `methoda`. Your base class `methoda` simply needs to return the number passed during the initialization of your base class raised to the power of 2.

Now create a sub class named `childclass` that references your parent class with two methods named `methodb` and `methodc`. Your `methodb` simply needs to return the number passed during the initialization of your class raised to the power of 3. Your `methodc` simply needs to return the number passed during the initialization of your class raised to the power of 4.

Please use the following starter code for your solution, placing your script solution between the markers named: ‘`### place your code below this line ###`’ and ‘`### place your code above this line ###`’ in the Starter Code Template below.

Starter Code Template:

```
class parentclass:
    ### place your code below this line ###

    ### place your code above this line ###

class childclass(parentclass):
    ### place your code below this line ###

    ### place your code above this line ###

myobj1 = childclass(10)
myobj1.methoda() # will output 100 to the console
myobj1.methodb() # will output 1000 to the console
myobj1.methodc() # will output 10000 to the console
```

You will know both of your classes have been correctly implemented by using the following quality assurance steps using the number 10 and the name myobj1 for the instantiation of your childclass as follows:

```
myobj1 = childclass(10)
myobj1.methoda() # will output 100 to the console
myobj1.methodb() # will output 1000 to the console
myobj1.methodc() # will output 10000 to the console
```