Linters

These questions are provided in the **Codio workspace** – Testing with Python – where the activities should be completed.

Question 1

Run styleLint.py in Codio.

What happens when the code is run?

An IndentationError occurred with the following error code returned.

File "styleLint.py", line 5

""" Return factorial of n """

Can you modify this code for a more favourable outcome?

Yes, the code indentation and formatting need to be corrected

What amendments have you made to the code?

```
# CODE SOURCE: SOFTWARE ARCHITECTURE WITH PYTHON

def factorial(n):
    """ Return factorial of n """  # indentation was corrected.
    if n == 0:  # indentation was corrected to bring this line with the function and
#subsequent lines below.
        return 1
    else:
        return n * factorial(n - 1)
```

Question 2

Run pylint on pylintTest.py

Review each of the code errors returned. Can you correct each of the errors identified by pylint?

```
# SOURCE OF CODE: https://docs.pylint.org/en/1.6.0/tutorial.html

import string

shift = 3
choice = input(“would you like to encode or decode?”)
word = input("Please enter text")
letters = string.ascii_letters + string.punctuation + string.digits
encoded = ''
if choice == "encode":
    for letter in word:
        if letter == ' ':
            encoded = encoded + ' '
        else:
            x = letters.index(letter) + shift
            encoded = encoded + letters[x]
        if choice == "decode":
            for letters in word:
                if letters == ' ':
                    encoded = encoded + ' '
                else:
                    x = letters.index(letter) - shift
                    encoded = encoded + letters[x]

print(encoded)
```

Question 3

Run flake8 on pylintTest.py

Review the errors returned. In what way does this error message differ from the error message returned by pylint?

Pylint provides thoroughly detailed reports of the code compared to flake8.

Run flake8 on metricTest.py. Can you correct each of the errors returned by flake8? What amendments have you made to the code?

```
# CODE SOURCE: SOFTWARE ARCHITECTURE WITH PYTHON

"""
 Module metricTest.py

Metric example - Module which is used as a testbed for static
checkers.
 This is a mix of different functions and classes doing
different things.

 """
import random


def fn(x, y):
    """ A function which performs a sum """
    return x + y


def find_optimal_route_to_my_office_from_home(start_time,
                                              expected_time,
                                              favorite_route='SBS1K',
```

```
                                        favorite_option='bus'):
    d = expected_time.total_seconds() / 60.0

    if d <= 30:
        return ‘car’

    # If d>30 but <45, first drive then take metro
    if 30 < d < 45:
        return 'car', 'metro'

    # If d>45 there are a combination of optionsWriting Modifiable and Readable Code

    if d > 45:
        if d < 60:
            # First volvo,then connecting bus
            return 'bus:335E', 'bus:connector'
        elif d > 80:
            # Might as well go by normal bus
            return random.choice(('bus:330', 'bus:331', ':'.
                                  join((favorite_option,
                                        favorite_route))))
        elif d > 90:
            # Relax and choose favorite route
            return ':'.join((favorite_option,
                             favorite_route))

    class C(object):
        """ A class which does almost nothing """

    def __init__(self, x, y):
        self.x = x
        self.y = y

    def f(self):
```

```
        pass

    def g(self, x, y):

        if self.x > x:
            return self.x + self.y
        elif x > self.x:
            return x + self.y

    class D(C):
        """ D class """

    def __init__(self, x):
        self.x = x

    def f(self, x, y):
        if x > y:
            return x - y
        else:

            return x + y

    def g(self, y):

        if self.x > y:
            return self.x + y
        else:
            return y - self.x
```

Question 4

Run McCabe on sums.py. What is the result?

Cyclomatic Complexity :

("4:0: 'test_sum'", 1)

('If 7', 2)

'test_sum' complexity 1

'If on line 7' complexity 2

Run mccabe on sums2.py. What is the result?

("7:0: 'test_sum_tuple'", 1)

("4:0: 'test_sum'", 1)

('If 10', 2)

What are the contributors to the cyclomatic complexity in each piece of code?

Functions and if statements.

Question 5 (e-portfolio entry): Exploring the Cyclomatic Complexity's Relevance Today

(i) The Cyclomatic Complexity is commonly considered in modules on testing the validity of code design Today. However, in your opinion, should it be?

Yes. Cyclomatic complexity can assess the system complexity and estimate the number of test cases required to achieve maximum code coverage. Complexity can be calculated based on the number of conditional statements present in the source code, and such conditional statements are modularised (Suresh, Pati and Rath, 2012).

(ii) Does it remain relevant Today?

Yes. Complexity measurements provide the basis for quality assessment to software professionals to fetch the requisite information about those metric suites. Which can predict faults while developing the metric-quality software products, estimate the number of test cases required to achieve maximum code coverage, and assess the system's complexity using the Traditional and Object-Oriented approaches (Lopez and Habra, 2005).

Specific to the focus of this module, is it relevant in our quest to develop secure software?

Alenezi and Zarour (2020) state that vulnerability is an emergent property of software caused by code quality, trusted data inputs, and complexity. Thus it is essential to reduce the software complexity and be able to measure and improve application quality.

References :

Alenezi, M. and Zarour, M. (2020) ‘On the Relationship between Software Complexity and Security’, *International Journal of Software Engineering & Applications*, 11(1), pp. 51–60. doi: 10.5121/ijsea.2020.11104.

Lopez, M. and Habra, N. (2005) ‘Relevance of the cyclomatic complexity threshold for the Java programming language’, *Smef 2005*, p. 195.

Suresh, Y., Pati, J. and Rath, S. K. (2012) ‘Effectiveness of Software Metrics for Object-oriented System’, *Procedia Technology*, 6, pp. 420–427. doi: 10.1016/j.protcy.2012.10.050.