J. Irvine Secure Software Development

Seminar 3 Preparation (and e-portfolio entry)

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?

The code would not run as there were errors. This was due to indentation. The code should return a factorial value of a number.

Can you modify this code for a more favourable outcome?

This code can be modified by adding an input that allows you to define n. To get the factorial of a number, simply multiply the number with the factorial of the previous number.
To get the value of 6!, it would be 6 × 5 × 4 × 3 × 2 × 1 = 720 or if you know the previous factorial i.e. 5, which is 120, multiply this by 6, and get 720.

What amendments have you made to the code?

The original code would not run as there were errors. This was due to indentation.

```
# ORIGINAL CODE SOURCE: SOFTWARE ARCHITECTURE WITH PYTHON

def factorial(n):
""" Return factorial of n """
if n == 0:
return 1
else:
return n*factorial(n-1)
```

Amended Code

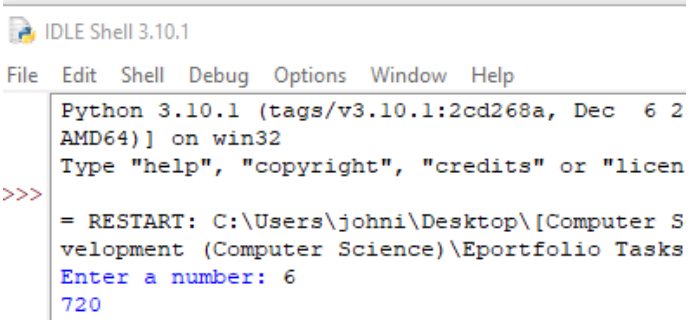
I removed the """ Return factorial of n """ line.
I added indentation and allowed a number to be inputted. The output demonstrates that it works using a factorial of 6, which is 720.

```
# CODE SOURCE: SOFTWARE ARCHITECTURE WITH PYTHON

n=int(input("Enter a number: "))

def factorial(n):
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)

print(factorial(n))
```



Question 2

`pip install pylint`
Run
`pylint`

on pylintTest.py

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

On the code sources site, it should have shown different errors. On my copy it had the following errors that had to be corrected:

The lines below required correction:

Original: choice = raw_input("would you like to encode or decode?")
Correction: choice = input("would you like to encode or decode?")

Original: word = (input("Please enter text"))
Correction: word = (input("Please enter text"))

Original: print encoded
Correction: print (encoded)

Before correcting the code errors, save the pylintTest.py file with a new name (it will be needed again in the next question).

Question 3

`pip install flake8`
Run
`flake8`
on pylintTest.py

Error 1

```
Microsoft Windows [Version 10.0.19044.3086]
(c) Microsoft Corporation. All rights reserved.

C:\Users\johni>d:

D:\>cd buffer overflow

D:\Buffer Overflow>flake8 pyLintTest.py
pyLintTest.py:26:2: E999 SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
```

After Error Correction

```
D:\Buffer Overflow>flake8 pyLintTest2.py
pyLintTest2.py:5:1: W293 blank line contains whitespace
pyLintTest2.py:7:10: F821 undefined name 'raw_input'
pyLintTest2.py:8:9: F821 undefined name 'raw_input'
pyLintTest2.py:12:3: E111 indentation is not a multiple of 4
pyLintTest2.py:14:7: E111 indentation is not a multiple of 4
pyLintTest2.py:16:7: E111 indentation is not a multiple of 4
pyLintTest2.py:17:7: E111 indentation is not a multiple of 4
pyLintTest2.py:17:14: E225 missing whitespace around operator
pyLintTest2.py:19:7: E111 indentation is not a multiple of 4
pyLintTest2.py:23:11: E111 indentation is not a multiple of 4
pyLintTest2.py:24:11: E111 indentation is not a multiple of 4
pyLintTest2.py:26:6: E211 whitespace before '('
```

Corrected Code:

```
# 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 letter in word:
        if letter == ' ':
              encoded = encoded + ' '
        else:
            x = letters.index(letter) - shift
            encoded = encoded + letters[x]

print (encoded)
```

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

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

Again this code was poor as there was a lack of indentation. After fixing I ran flake8 and had the following errors:

```
D:\Buffer Overflow>flake8 metricTest2.py
metricTest2.py:24:2: E999 IndentationError: expected an indented block after function definition on line 19

D:\Buffer Overflow>flake8 metricTest.py
metricTest.py:12:1: E302 expected 2 blank lines, found 1
metricTest.py:16:1: E302 expected 2 blank lines, found 1
metricTest.py:16:44: E231 missing whitespace after ','
metricTest.py:16:80: E501 line too long (91 > 79 characters)
metricTest.py:26:80: E501 line too long (84 > 79 characters)
metricTest.py:32:80: E501 line too long (90 > 79 characters)
metricTest.py:38:1: E305 expected 2 blank lines after class or function definition, found 1
metricTest.py:40:1: E302 expected 2 blank lines, found 1
metricTest.py:56:1: E305 expected 2 blank lines after class or function definition, found 1
metricTest.py:57:1: E302 expected 2 blank lines, found 0
metricTest.py:76:1: W391 blank line at end of file
```

Question 4

`pip install mccabe`
Run
`mccabe`
on sums.py. What is the result?

The instruction was wrong I had to run the following command:

python -m mccabe sums.py

This is the output:

```
D:\Buffer Overflow>python -m mccabe sums.py
4:0: 'test_sum' 1
If 7 2
```

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

I ran the following command: python -m mccabe sums2.py

This is the output:

```
D:\Buffer Overflow>python -m mccabe sums2.py
3:0: 'test_sum' 1
6:0: 'test_sum_tuple' 1
If 9 2
```

J. Irvine Secure Software Development

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

Tiwari and Kumar defined cyclomatic complexity as the measurement of the intricacy of a program by counting the number of decision points in the programmes control flow. Decision points include if statements, while loops, for loops, and switch statements.

In the code below for *sums.py* there is only one decision point, with the 'if' statement, giving this a cyclomatic complexity of 1.

```
def test_sum():
    assert sum([1, 2, 3]) == 6, "Should be 6"

if __name__ == "__main__":
    test_sum()
    print("Everything passed")
```

In the code below for *sums2.py* there is only two decision points, giving this a cyclomatic complexity of 2.

```
def test_sum():
    assert sum([1, 2, 3]) == 6, "Should be 6"

def test_sum_tuple():
    assert sum((1, 2, 2)) == 6, "Should be 6"

if __name__ == "__main__":
    test_sum()
    test_sum_tuple()
    print("Everything passed")
```

Tiwari, U. and Kumar, S., 2014. Cyclomatic complexity metric for component based software. ACM SIGSOFT Software Engineering Notes, 39(1), pp.1-6.