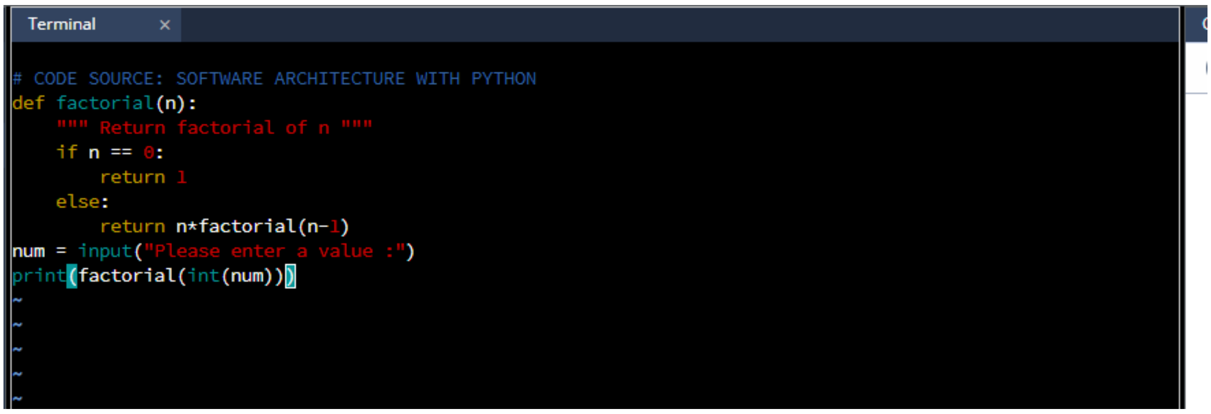
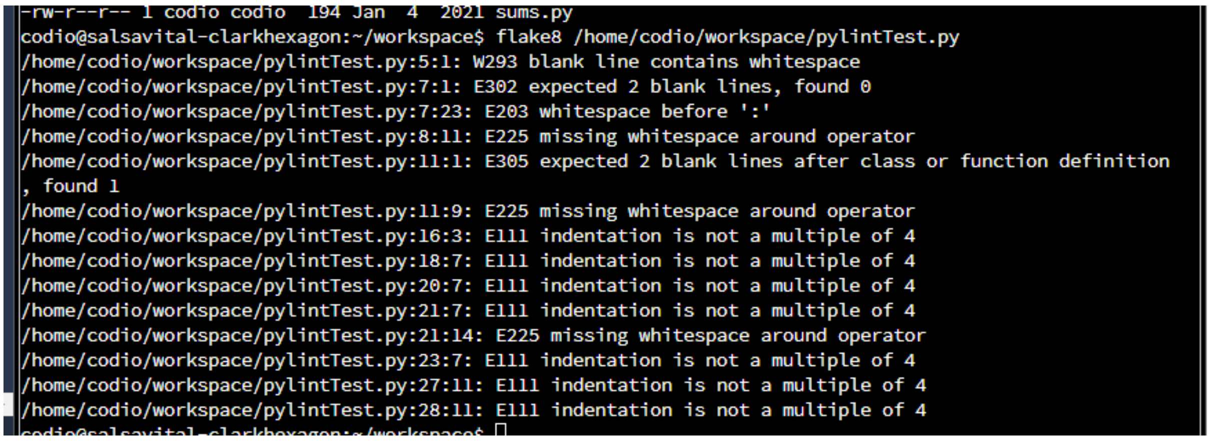
Unit 4

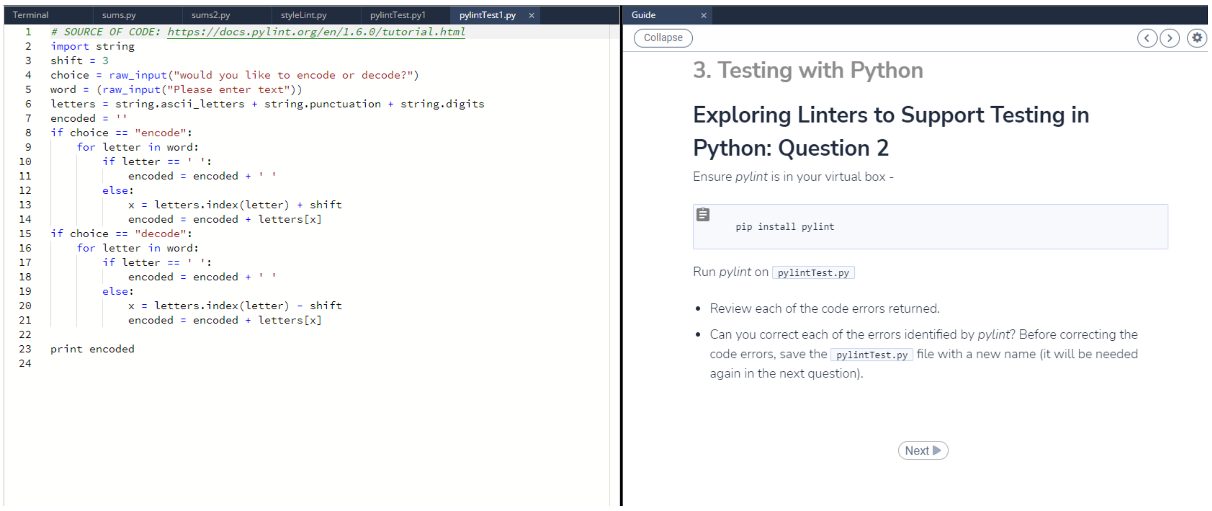
styleLint.py has been modified to take a input from the user, which is storing the answer in a variable called num. The factorial function is being called passing the num, which is a type int.



```
# CODE SOURCE: SOFTWARE ARCHITECTURE WITH PYTHON
def factorial(n):
    """ Return factorial of n """
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)
num = input("Please enter a value :")
print(factorial(int(num)))
~
~
~
~
~
```



```
-rw-r--r-- 1 codio codio  194 Jan  4  2021 sums.py
codio@salsavital-clarkhexagon:~/workspace$ flake8 /home/codio/workspace/pylintTest.py
/home/codio/workspace/pylintTest.py:5:1: W293 blank line contains whitespace
/home/codio/workspace/pylintTest.py:7:1: E302 expected 2 blank lines, found 0
/home/codio/workspace/pylintTest.py:7:23: E203 whitespace before ':'
/home/codio/workspace/pylintTest.py:8:11: E225 missing whitespace around operator
/home/codio/workspace/pylintTest.py:11:1: E305 expected 2 blank lines after class or function definition
, found 1
/home/codio/workspace/pylintTest.py:11:9: E225 missing whitespace around operator
/home/codio/workspace/pylintTest.py:16:3: E111 indentation is not a multiple of 4
/home/codio/workspace/pylintTest.py:18:7: E111 indentation is not a multiple of 4
/home/codio/workspace/pylintTest.py:20:7: E111 indentation is not a multiple of 4
/home/codio/workspace/pylintTest.py:21:7: E111 indentation is not a multiple of 4
/home/codio/workspace/pylintTest.py:21:14: E225 missing whitespace around operator
/home/codio/workspace/pylintTest.py:23:7: E111 indentation is not a multiple of 4
/home/codio/workspace/pylintTest.py:27:11: E111 indentation is not a multiple of 4
/home/codio/workspace/pylintTest.py:28:11: E111 indentation is not a multiple of 4
codio@salsavital-clarkhexagon:~/workspace$
```



```python
# SOURCE OF CODE: https://docs.pylint.org/en/1.6.0/tutorial.html
import string
shift = 3
choice = raw_input("would you like to encode or decode?")
word = (raw_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

```

3. Testing with Python

Exploring Linters to Support Testing in Python: Question 2

Ensure *pylint* is in your virtual box -

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

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

The Cyclomatic Complexity is commonly considered in modules on testing the validity of code design today. However, in your opinion, should it be? Does it remain relevant today? Specific to the focus of this module, is it relevant in our quest to develop secure software? Justify all opinions which support your argument and share your responses with your team.

My understanding of Cyclomatic Complexity is the number of counts of logical paths. This can include if statement for loops and while loops which will increase the complexity. I feel that is an invalid way of determining how complex code is. What is important is how easy the code is to understand so that when performing a code change it does not become a breaking change. A great write up by Adam Tornhill although selling a product he describes a nested for loop has the same complicity as a switch statement. However, comparing the two the switch statement is far easier to read and understand [Tornhill, n/a]. As part of Object-Oriented programming there is a concept of DRY don't repeat yourself. The principle is that code logic has one single unambiguous representation within the system. [Zobenica, 2020] This approach would be beneficial in reducing the Cyclomatic Complexity. Ensuring that code is not duplicated by using principles of polymorphism and inheritance where the child class is able to re-use methods and attributes from the parent, however the flexibility is present to able to override if there is a need to be redefined in the child class. When considering secure coding. Object-Oriented approach of abstraction and encapsulation where only essential attributes are visible and all other details are hidden.[Hartman (2020]]

Hartman. J (2022) OOPs Concepts in Java: What is, Basics with Examples Distribution Available at: https://www.guru99.com/java-oops-concept.html Accessed: 14 December 2022).

Tornhill, A The Bumpy Road Code Smell: Measuring Code Complexity by its Shape and Distribution Available at: https://codescene.com/blog/bumpy-road-code-complexity-in-context/ Accessed: 14 December 2022).

Zobenica .D (2020) Object Oriented Design Principles in Java Available at: https://stackabuse.com/object-oriented-design-principles-in-java/#dontrepeatyourselfdryprinciple/ Accessed: 14 December 2022).

Read the Cryptography with Python blog at tutorialspoint.com (link is in the reading list). Select one of the methods described/ examples given and create a python program that can take a short piece of text and encrypt it.

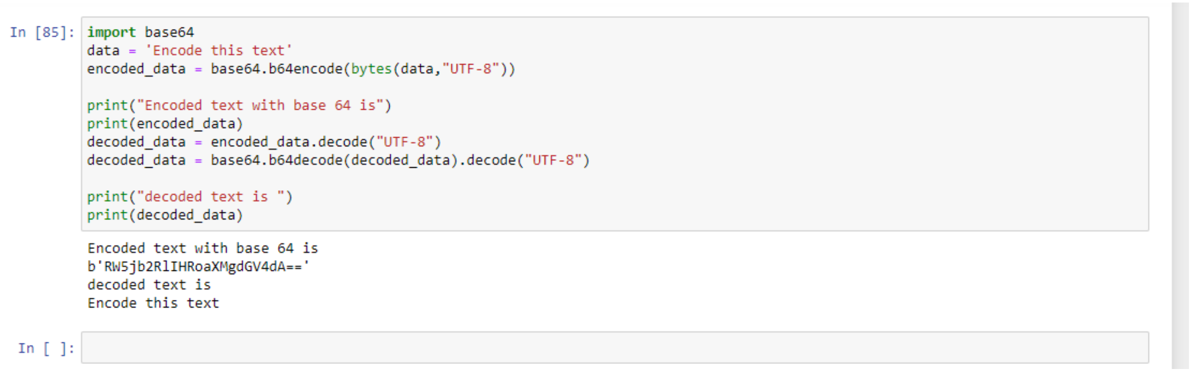
Create a python program in Codio (you can use the Jupyter Notebooks space provided in the Codio resources section) that can take a text file and output an encrypted version as a file in your folder on the Codio system. Demonstrate your program operation in this week's seminar session.

- Why did you select the algorithm you chose?

  Base64 is also called the Privacy enhanced Electronic Mail and used for email encryption.

- Would it meet the GDPR regulations? Justify your answer.

  The fact that base64 is used for email encryption I think it would meet the GDPR standards.

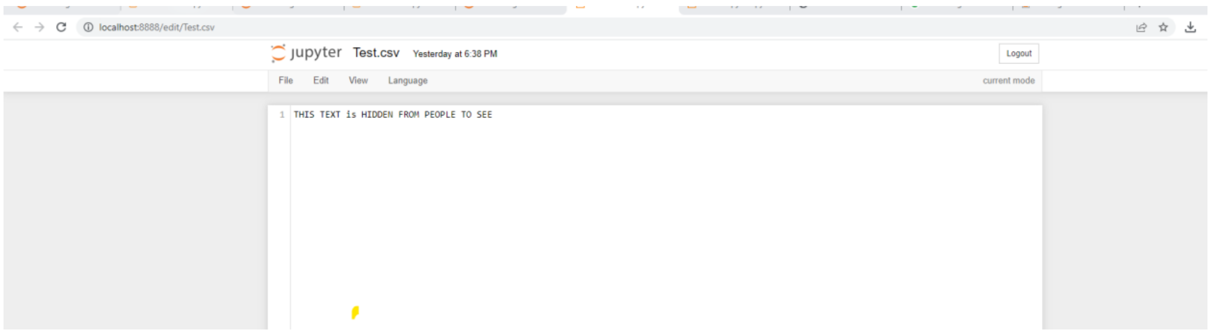


```
In [85]: import base64
         data = 'Encode this text'
         encoded_data = base64.b64encode(bytes(data,"UTF-8"))

         print("Encoded text with base 64 is")
         print(encoded_data)
         decoded_data = encoded_data.decode("UTF-8")
         decoded_data = base64.b64decode(decoded_data).decode("UTF-8")

         print("decoded text is ")
         print(decoded_data)
```

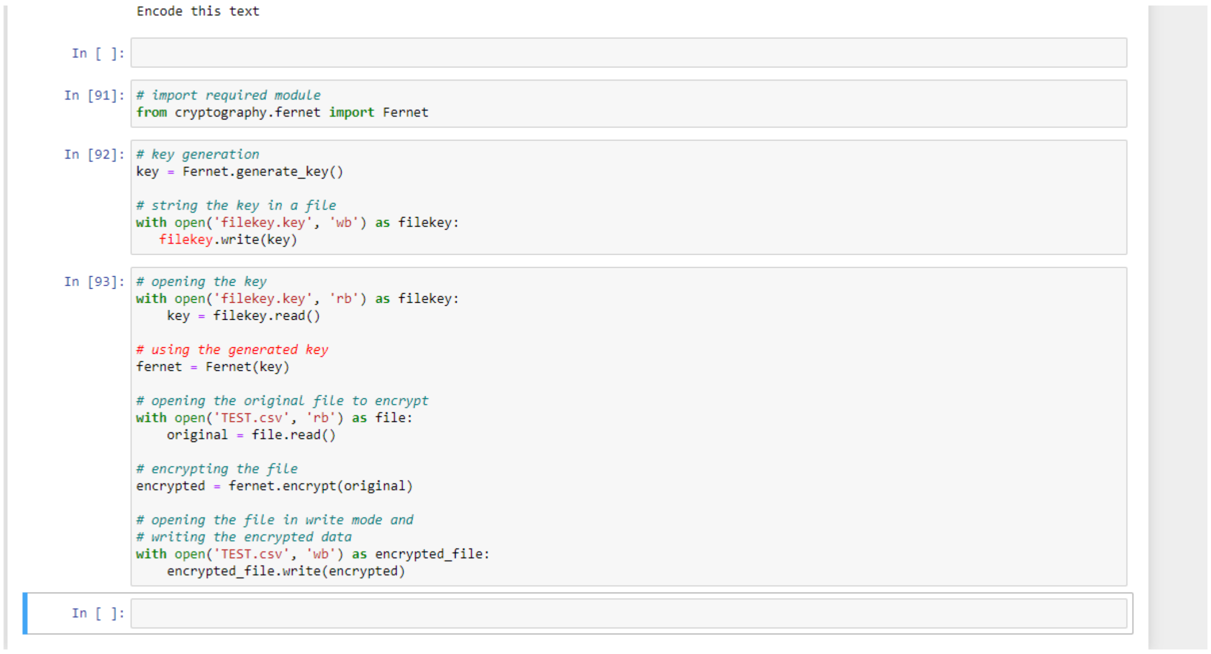
```
Encoded text with base 64 is
b'RW5jb2RlIHRoaXMgdGV4dA=='
decoded text is
Encode this text
```

```
In [ ]:
```

TEST File


localhost:8888/edit/Test.csv
jupyter Test.csv Yesterday at 6:38 PM
Logout
File Edit View Language
current mode

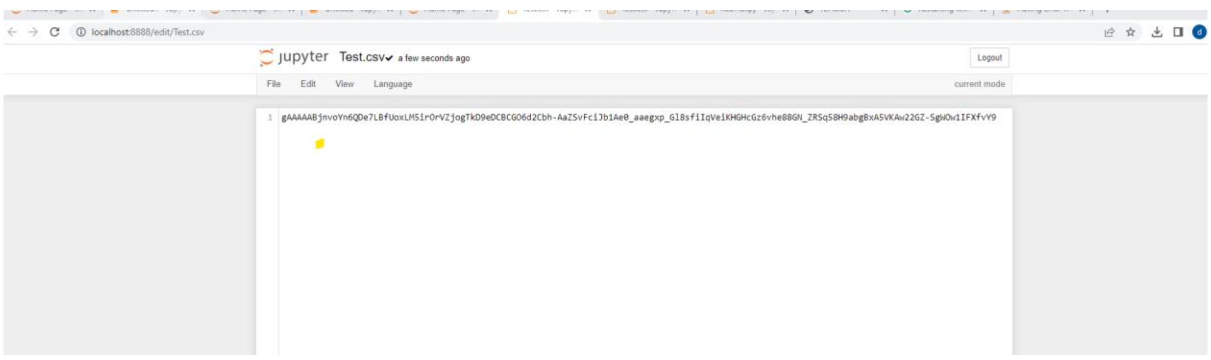
```
THIS TEXT is HIDDEN FROM PEOPLE TO SEE
```



Encode this text

```
In [ ]:

In [91]: # import required module
         from cryptography.fernet import Fernet

In [92]: # key generation
         key = Fernet.generate_key()

         # string the key in a file
         with open('filekey.key', 'wb') as filekey:
            filekey.write(key)

In [93]: # opening the key
         with open('filekey.key', 'rb') as filekey:
             key = filekey.read()

         # using the generated key
         fernet = Fernet(key)

         # opening the original file to encrypt
         with open('TEST.csv', 'rb') as file:
             original = file.read()

         # encrypting the file
         encrypted = fernet.encrypt(original)

         # opening the file in write mode and
         # writing the encrypted data
         with open('TEST.csv', 'wb') as encrypted_file:
             encrypted_file.write(encrypted)

In [ ]:
```

TEST file after


localhost:8888/edit/Test.csv
jupyter Test.csv✔ a few seconds ago
Logout
File Edit View Language
current mode

```
gAAAAABjnvoYn6QDe7LBfUoxLM5irOrVZjogTkD9eDCBCGO6d2Cbh-AaZSvFciJb1Ae0_aaegxp_Gl8sfiIqVeiKHGHcGz6vhe88GN_ZRSq58H9abgBxA5VKAw22GZ-SgWOw1IFXfvY9
```