

Introductory Seminar Preparation:

Team Contract

Team Name: **Complete Mediation** _____

GOALS: What are our team goals for this project?

What do we want to accomplish? What skills do we want to develop or refine?

Goals:

Design and implement an application that provides a secure repository for an organisation with domain-specific requirements. The distributed/microservice version of the application should also be designed and developed. We hope the application can have high quality and low-security risks. During the development, we hope each member can get improved in technical and teamwork abilities.

Achieving:

technical: Practice our learning from the SSD module, and apply best practices in our software system, especially in the security area.

Teamwork: Establish a productive and trustworthy team environment, and each member has a chance for self-improvement and growth.

To accomplish the goals, we need to know about:

- the organizations (NASA, The Netherland Police, CERN) and their system requirements
- privacy and security regulations such as British GDPR
- frameworks such as STRIDE, OWASP and their security challenges
- distributed/microservice systems
- UML design diagrams
- other skills such as authentication, authorization, data encryption and event monitoring
- system analysis and design
- programming
- team communication
- time management
- planning arrangement
- risk management.

EXPECTATIONS: What do we expect of one another in regard to attendance at meetings, participation, frequency of communication, the quality of work, etc.?

Each member of the team is going to make sure to:

1. respectfully treat other members of the team.
2. share own ideas and let other members to do so freely.
3. equally participate in the project we all are assigned for.

4. try to participate in project meetings. However, if it is impossible to do it, a member will contact the leader or any member of the team for instructions to complete the assigned part of the job in the project.
POLICIES & PROCEDURES: What rules can we agree on to help us meet our goals and expectations, such as preparatory tasks, generating ideas, evaluating outcomes?
<p>1. We will create a space (for example a google account) to share our pieces of work in the cloud for all members to have access to them.</p> <p>2. Each member is responsible to review any work that was done by others and give feedback and any critical comment to improve our common goal.</p> <p>3. All ideas or solutions that are not our own creation have to have literature references to avoid plagiarism.</p> <p>4. All members should have copies of stored files in the cloud on their own personal computers in the case of any loss.</p> <p>5. In dividing jobs and tasks in the project, personal strengths regarding technical skills will be prioritized.</p>
ROLES: Which roles do we need in this project and how do we allocate them? Will there be a project lead? Is there a need to rotate roles?
<p>Assigning roles in a group is critical. Some roles are as follows:</p> <p><u>Leader</u>: We will choose the group leader if the chosen person agrees to the role. If not, the role of the team leader will be changing. Each member of the team will be the team Leader for an equal number of weeks throughout the module.</p> <p>Responsibilities of the team leader:</p> <ul style="list-style-type: none"> • dividing work between the rest of the members of the team. • gathering all pieces of work and updating the cloud with the latest versions of files. • Coordinate team meetings and make team meeting reports. <p><u>Editor</u>:</p> <p>Responsibilities of the editor:</p> <ul style="list-style-type: none"> • editing the completed work • compiling pieces of work from group members
CONSEQUENCES: How will we address non-performance regarding these goals, expectations, policies and procedures? How do we resolve disagreements?
<p>Sometimes, some group members don't contribute to the project. To prevent the problem from occurring, a guideline should be set up in the first meeting and roles be assigned to each member. Despite a clear guideline, uneven contribution still can happen. To tackle the problem, the team leader should first speak to the non-performing members of the group to find out whether they are consistent non-performers or stressed/unfocused performing members. Then, the team leader should listen to the member to know the causes of non-performance. At the end, the leader should share the ways that they want the member to improve. To resolve the problem completely, a regular follow-up meeting might be needed to keep track of the member's progress.</p> <p>We will evaluate each other's contribution objectively, if someone fails to contribute their share they will ultimately be held accountable through the evaluation process. To avoid having to give poor feedback we will set clear targets, meet deadlines and support one another.</p>

To resolve disagreements: the team should vote from a selection of available ideas.

By signing this contract, we confirm that:

- We have participated in formulating these goals, expectations, roles, procedures, and consequences as stated in this contract.
- we agree to abide by the contents of this contract

Maja Tagt 20 June 2022

Team member name and date

Elham Mortazavi Sarmad 19-June-2022

Team member name and date

Grzegorz Pikus 19-Jun-22

Team member name and date

Richard Allen 19-Jun-22

Team member name and date

Wang Wang 19-Jun-22

Team member name and date

Seminar 1 Preparation:

(Seminar 1 blog post)

Cyber Attacks

Cybercriminals are constantly looking for ways to attack computers. To be successful, they try to find security vulnerabilities in the computers, and they take advantage of the flaws to gain money or ruin reputation. Therefore, being aware of different types of vulnerabilities is necessary for organizations. Security vulnerabilities are divided into different types which are related to network, software, human or process (Dosal, 2020). Among them, human-related vulnerabilities can

play important roles in *attacks*. According to CompTIA, more than half of the security breaches are linked to human error. There are two aspects in the human error issue: First, the error on the part of non-IT employees and customers of the organizations. Second, the error on the part of IT professionals of the companies. Today, non-IT employees and end users are easy targets for cybercriminals because they are less aware of security. So, 'passive attacks' are carried out which means that people security behaviours are monitored by the attackers to find a way to exploit them later (CompTIA, 2017). To prevent the attacks, organizations should equip their devices and software with more enhanced security such as multifactor *authentication* (MFA). Moreover, security team should communicate security policy to the employees and equip them with security trainings.

From IT professionals' perspective, error can occur due to increase in numbers of available tools and the complexity in integrating them. The tools include network monitoring, intrusion detection and security information and event management (SIEM) software. To prevent the error, IT professionals also need regular trainings (CompTIA, 2017). Certifications are the best way of investing in IT staff and equipping them with technical skills required for cyber security.

Having said about cyber-attacks, it is time to mention insider threats which refer to risk within the organizations. It involves staffs who have access to the information and misuse this privilege (imperva, 2022). To prevent the insider threats, companies can set up multiple layers of protections including database firewalls which block SQL injection, Identify and Access Management systems which monitor activities of privileged users, data masking and data encryption.

References

- Dosal, E. (2020). Top 5 computer security vulnerabilities - compuquip.
Available from <https://www.compuquip.com/blog/computer-security-vulnerabilities> [Accessed 27 June 2022]
- N.D. (2017). The UK cybersecurity landscape 2017 and beyond: A report.
Available from: <https://connect.comptia.org/content/whitepapers/the-cybersecurity-landscape> [Accessed 27 June 2022]
- N.D. (2022). Insider threat. Available from:
<https://www.imperva.com/learn/application-security/insider-threats/> [Accessed 27 June 2022]

Seminar 2 Preparation:

(Tower of Hanoi)

Read the explanation, study the code, and then create your own version using Python (if you want to make it more interesting you can use asterisks to represent the disks). Create a version that asks for the number of disks and then executes the moves, and then finally displays the number of moves executed.

```

1  # getting input (the number of disks) from the user
2  number_of_disks = int(input("Enter the number of disks: "))
3
4  # calculating the number of moves
5  number_of_moves = (2 ** number_of_disks) - 1
6  print(f"The number of the moves: {number_of_moves}")
7
8  def tower_of_hanoi(number_of_disks, source_peg, spare_peg, target_peg):
9      if number_of_disks == 1:
10         print(f"Move disk 1 from peg {source_peg} to peg {target_peg}")
11         return
12
13         # calling the function inside itself (for the first time)
14         tower_of_hanoi(number_of_disks - 1, source_peg, target_peg, spare_peg)
15
16         print(f"Move disk {number_of_disks} from peg {source_peg} to peg {target_peg}")
17
18         # calling the function inside itself (for the second time)
19         tower_of_hanoi(number_of_disks - 1, spare_peg, source_peg, target_peg)
20
21 # calling the function
22 tower_of_hanoi(number_of_disks, "A", "B", "C")
23

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Microsoft Windows [Version 10.0.19043.1889]
(c) Microsoft Corporation. All rights reserved.

```

C:\Users\emsar>python towerOfHanoi.py
Enter the number of disks: 4
The number of the moves: 15
Move disk 1 from peg A to peg B
Move disk 2 from peg A to peg C
Move disk 1 from peg B to peg C
Move disk 3 from peg A to peg B
Move disk 1 from peg C to peg A
Move disk 2 from peg C to peg B
Move disk 1 from peg A to peg B
Move disk 4 from peg A to peg C
Move disk 1 from peg B to peg C
Move disk 2 from peg B to peg A
Move disk 1 from peg C to peg A
Move disk 3 from peg B to peg C
Move disk 1 from peg A to peg B
Move disk 2 from peg A to peg C
Move disk 1 from peg B to peg C

```

What is the (theoretical) maximum number of disks that your program can move without generating an error?

Theoretically, any number of disks can be moved.

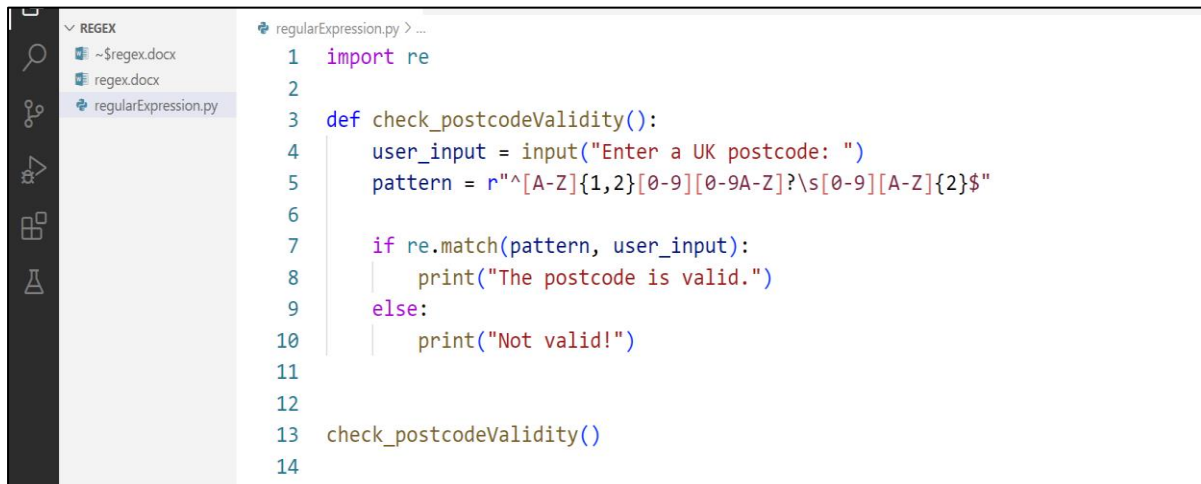
In the python code of Tower of Hanoi, recursion has been used. Recursion means that the function calls itself within its code. Practically, the number of disks should be based on the stack, otherwise the memory cannot manage, and the program will give error.

What limits the number of iterations? What is the implication for application and system security?

The if statement in the code can stop the recursion and return. But recursive programs can fail to terminate, and this causes memory overflow and crashes the software at the end.

(Regex)

Create a python program that implements a regex that complies with the rules provided above – test it against the examples provided. Examples: M1 1AA, M60 1NW, CR2 6XH, DN55 1PT, W1A 1HQ, EC1A 1BB.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder named 'REGEX' containing three files: '\$regex.docx', 'regex.docx', and 'regularExpression.py'. The code editor is open to 'regularExpression.py' and contains the following Python code:

```
1 import re
2
3 def check_postcodeValidity():
4     user_input = input("Enter a UK postcode: ")
5     pattern = r"^[A-Z]{1,2}[0-9][0-9A-Z]?[0-9][A-Z]{2}$"
6
7     if re.match(pattern, user_input):
8         print("The postcode is valid.")
9     else:
10        print("Not valid!")
11
12
13 check_postcodeValidity()
14
```

Enter a UK postcode: M1 1AA
The postcode is valid.

Enter a UK postcode: M60 1NW
The postcode is valid.

Enter a UK postcode: CR2 6XH
The postcode is valid.

Enter a UK postcode: DN55 1PT
The postcode is valid.

Enter a UK postcode: W1A 1HQ
The postcode is valid.

Enter a UK postcode: EC1A 1BB
The postcode is valid.

How do you ensure your solution is not subject to an evil regex attack?

Creating good patterns prevent Evil Regex which cause Denial of Service (DoS) attack or crash the software. To create a good pattern, it is recommended not to use repetition symbols like + or *.

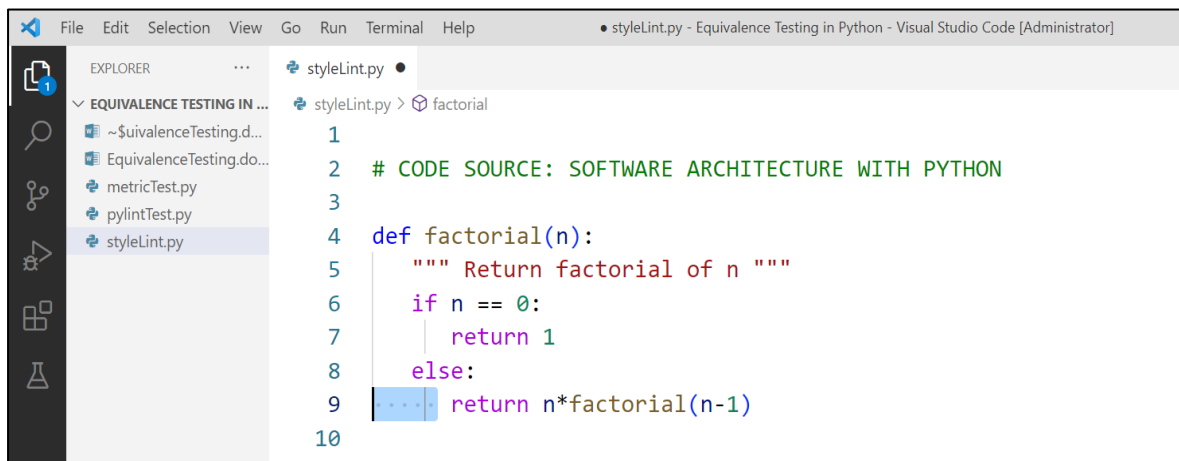
Seminar 3 Preparation:

Question 1

Run `styleLint.py`

- **What happens when the code is run?** There is an Indentation error.
- **Can you modify this code for a more favourable outcome?** Yes
- **What amendments have you made to the code?** I indented every line.

`styleLint.py`



The screenshot shows the Visual Studio Code interface with the file `styleLint.py` open. The Explorer sidebar on the left shows a project structure under 'EQUIVALENCE TESTING IN ...' with files: `~$ivalenceTesting.d...`, `EquivalenceTesting.do...`, `metricTest.py`, `pylintTest.py`, and `styleLint.py`. The main editor displays the following Python code:

```
1
2 # CODE SOURCE: SOFTWARE ARCHITECTURE WITH PYTHON
3
4 def factorial(n):
5     """ Return factorial of n """
6     if n == 0:
7         return 1
8     else:
9         return n*factorial(n-1)
10
```

Question 2

```
pip install pylint
```

Run pylint on `pylintTest.py`

- Review each of the code errors returned.

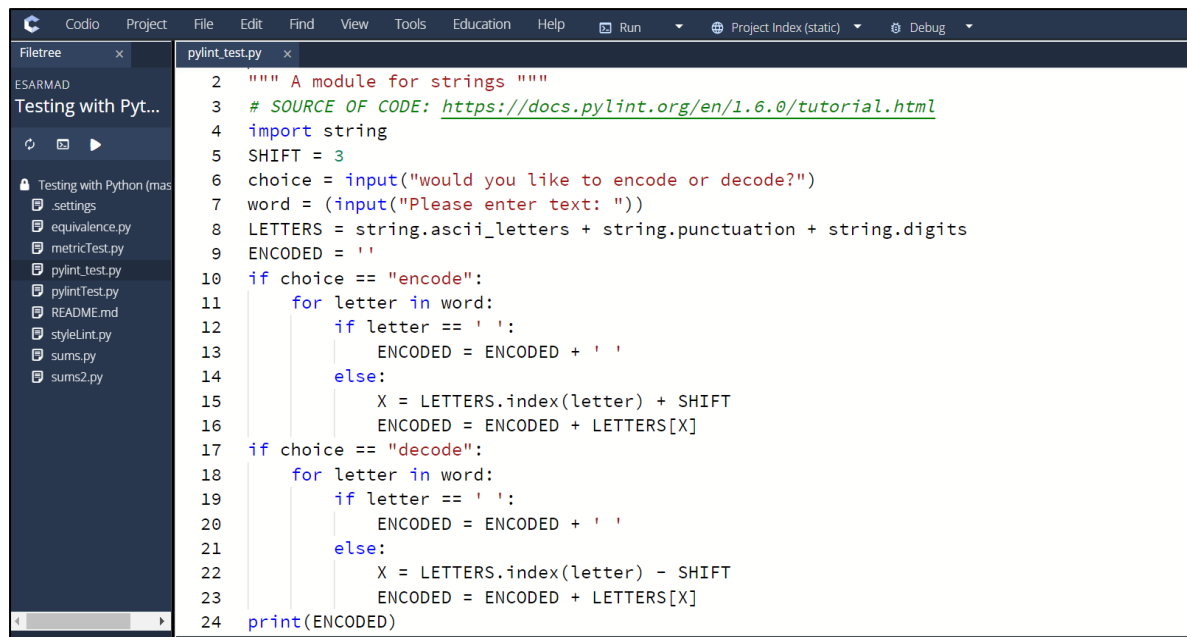
The errors are as follows:

- Module name "styleLint" doesn't conform to snake_case naming
- Missing module docstring pylint (missing-module-docstring)
- Constant name "shift" doesn't conform to UPPER_CASE naming style pylint (invalid-name)
- Bad indentation. Found 2 spaces, expected 4 pylint (bad-indentation)
- Trailing whitespace pylint (trailing-whitespace)
- `print(*values: object, sep: str | None = ..., end: str | None = ..., file: SupportsWrite[str] | None = ..., flush: Literal[False] = ...) -> None`

- Can you correct each of the errors identified by *pylint*?

- For snake-case naming convention, we should change the name of the file into 'pylint_test.py'.
- For module docstring, we should add `""" """` before import module.
- For upper-case naming style for constants, we should change the names into 'SHIFT', 'LETTERS', 'ENCODED' and 'X'.
- For bad indentation, we should change the indentation to four spaces
- For trailing whitespace, we should delete whitespaces.
- For the last statement in the code, we should change it into `print(ENCODED)`.

pylint_test.py



```
2 """ A module for strings """
3 # SOURCE OF CODE: https://docs.pylint.org/en/1.6.0/tutorial.html
4 import string
5 SHIFT = 3
6 choice = input("would you like to encode or decode?")
7 word = (input("Please enter text: "))
8 LETTERS = string.ascii_letters + string.punctuation + string.digits
9 ENCODED = ''
10 if choice == "encode":
11     for letter in word:
12         if letter == ' ':
13             ENCODED = ENCODED + ' '
14         else:
15             X = LETTERS.index(letter) + SHIFT
16             ENCODED = ENCODED + LETTERS[X]
17 if choice == "decode":
18     for letter in word:
19         if letter == ' ':
20             ENCODED = ENCODED + ' '
21         else:
22             X = LETTERS.index(letter) - SHIFT
23             ENCODED = ENCODED + LETTERS[X]
24 print(ENCODED)
```

Question 3

`pip install flake8`

Run flake8 on `pylintTest.py`

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

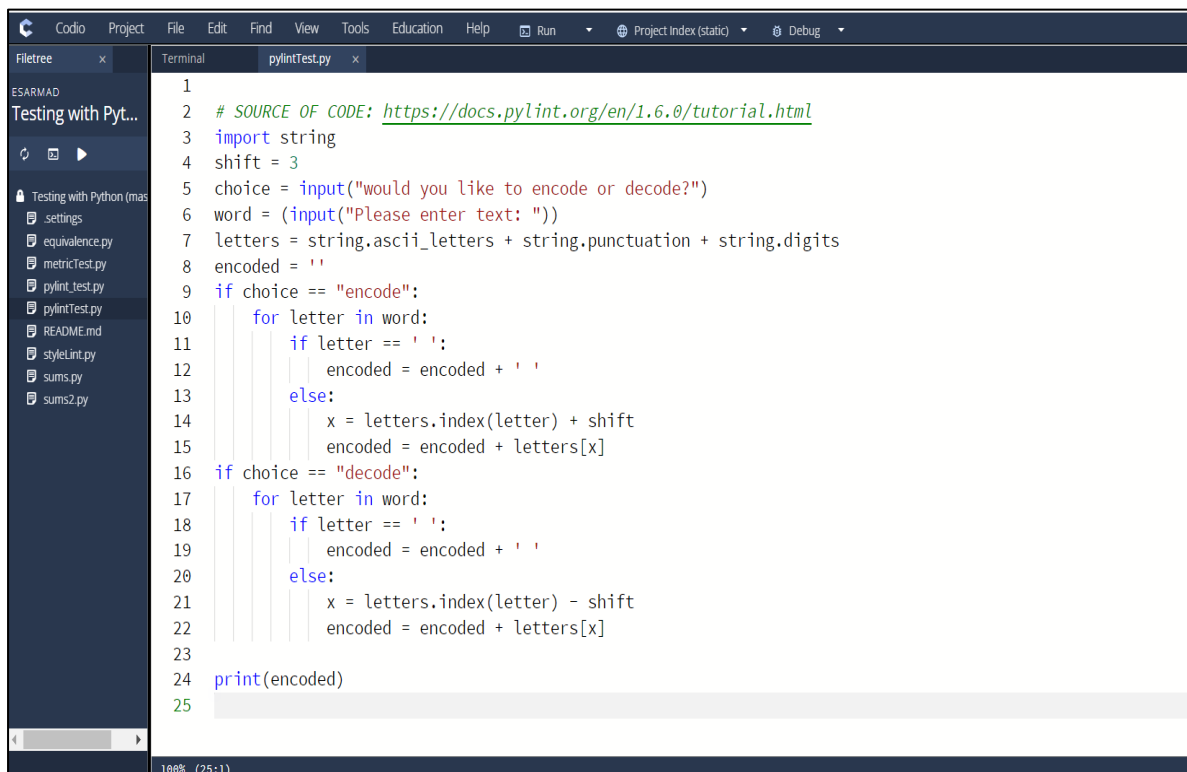
The errors are as follows:

- Whitespace error
- "raw_input" is not defined Pylance ([reportUndefinedVariable](#))
- indentation is not a multiple of 4 flake8(E111)
- missing whitespace around operator flake8(E225)
- no newline at end of file flake8(W292)

The corrections are as follows:

- For whitespace error, we should remove the whitespaces after import module
 - For raw-input(), we should change it into 'input()'.
 - For indentation error, we should change the indentation into 4 spaces.
 - For missing whitespace, we should put whitespace around = operator like this: 'encoded = encoded + letters[x]'
- For newline error, we should change the place of cursor from this line (print(encoded)) to the next line.

pylintTest.py



```
1
2 # SOURCE OF CODE: https://docs.pylint.org/en/1.6.0/tutorial.html
3 import string
4 shift = 3
5 choice = input("would you like to encode or decode?")
6 word = (input("Please enter text: "))
7 letters = string.ascii_letters + string.punctuation + string.digits
8 encoded = ''
9 if choice == "encode":
10     for letter in word:
11         if letter == ' ':
12             encoded = encoded + ' '
13         else:
14             x = letters.index(letter) + shift
15             encoded = encoded + letters[x]
16 if choice == "decode":
17     for letter in word:
18         if letter == ' ':
19             encoded = encoded + ' '
20         else:
21             x = letters.index(letter) - shift
22             encoded = encoded + letters[x]
23
24 print(encoded)
25
```

Run flake8 on `metricTest.py`

- Can you correct each of the errors returned by flake8?

The errors are as follows:

- block comment should start with '#' flake8(E265)
- trailing whitespace flake8(W291)

- expected 2 blank lines, found 1 flake8(E302)
- missing whitespace after ',' flake8(E231)
- line too long (121 > 79 characters) flake8(E501)
- indentation is not a multiple of 4flake8(E111)
- IndentationError: unindent does not match any outer indentation levelflake8(E999)

• What amendments have you made to the code?

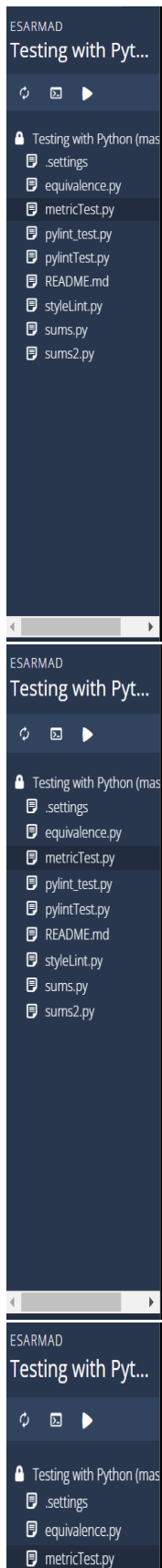
- For block comment with '#' error, we should put a space after '#'.
- For trailing whitespace error, we should remove spaces at the end of the line.
- For 2 blank lines, we should put two blank lines before functions,...
- For missing whitespace after ',', we should put ',' after the name of variable such as 'start_time, expected_time'.
- For too long line error, we should just put not more than 79 characters in a line.
- For indentation error, we should change the indentation into 4 spaces.
- For indentationError, we should change 'if d>45:' into 'if d > 45'.

metricTest.py

```

2  # CODE SOURCE: SOFTWARE ARCHITECTURE WITH PYTHON
3  """
4  Module metricTest.py
5  Metric example - Module which is used as a testbed for static checkers.
6  This is a mix of different functions and classes doing different things.
7  """
8
9  import random
10
11
12  def fn(x, y):
13      """ A function which performs a sum """
14      return x + y
15
16
17  def find_optimal_route_to_my_office_from_home(start_time, expected_time,
18                                              favorite_route='SBS1K',
19                                              favorite_option='bus'):
20
21      d = (expected_time, start_time).total_seconds() / 60.0
22      if d <= 30:
23          return 'car'
24      # If d > 30 but <45, first drive then take metro

```



```
25     if d > 30 and d < 45:
26         return ('car', 'metro')
27
28     """If d>45 there are a combination of optionsWriting Modifiable
29     and Readable Code"""
30
31     if d > 45:
32         if d < 60:
33             # First volvo, then connecting bus
34             return ('bus: 335E', 'bus: connector')
35         elif d > 80:
36             # Might as well go by normal bus
37             return random.choice(('bus: 330', 'bus: 331', ':'.join((
38                 favorite_option, favorite_route))))
39         elif d > 90:
40             # Relax and choose favorite route
41             return ':'.join((favorite_option, favorite_route))
42
43
44     class C(object):
45         """ A class which does almost nothing """
46
47         def __init__(self, x, y):
48             self.x = x
49             self.y = y
50
51         def f(self):
52             pass
53
54         def g(self, x, y):
55             if self.x > x:
56                 return self.x+self.y
57             elif x > self.x:
58                 return x + self.y
59
60
61     class D(C):
62         """ D class """
63         def __init__(self, x):
64             self.x = x
65
66         def f(self, x, y):
67             if x > y:
68                 return x-y
69             else:
70                 return x+y
71
72         def g(self, y):
73             if self.x > y:
74                 return self.x + y
75             else:
76                 return y - self.x
77
```

Question 4

```
pip install mccabe
```

Run mccabe on `sums.py`. what is the result?

```
$ python -m mccabe --min 1 sums.py
3:0: 'test_sum' 1
If 6 2
```

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

```
$ python -m mccabe --min 1 sums2.py
4:0: 'test_sum' 1
7:0: 'test_sum_tuple' 1
If 10 2
```

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

In 'sums.py', the cyclomatic complexity numbers for 'test_sum' function is 3 which means that the function is easy to understand and test. The reason behind it is that the function is small, and it has just used one assertion and one function invocation.

In 'sums2.py', the cyclomatic complexity numbers for 'test_sum' and 'test_sum_tuple' are less than 10. It means that the functions are considered as a medium complexity because of using 2 assertions and 2 function invocations. So, there are more lines of code.

Seminar 4 Preparation:

Select one of the methods for cryptography and create a python program that can take a short piece of text and encrypt it. Answer the following questions in your e-portfolio:

- Why did you select the algorithm you chose?
- Would it meet the GDPR regulations? Justify your answer.

The method I am using is called Data Encryption Standard (DES). It takes 64-bit plain text and 56-bit key as input and produces 64-bit ciphertext as output. DES is a symmetric block cipher. Besides, there are 256 possible keys, which means a brute force attack will never have any impact. It is easy to implement in both hardware and software and it is ubiquitous because most systems and libraries support it.

Under GDPR (General Data Protection Regulation), encrypting personal data is not obligatory for organizations, though European standards are different (Wikipedia, n.d). Countries considered GDPR as one of the security measures in their text of the Data regulations. For instance, US-based Information Commissioner's Office (NIST, 2001), as a national data protection authority, recommends that encryption solution should meet current standards such as FIPS 140-2. Here, they regarded the standard as "the security requirements that a cryptographic module should satisfy which include everything related to secure design and implementation of the module such as authentication and cryptographic key management" (NIST, 2001). The standard also provides a list of approved algorithms for encryption which includes the method (DES) I am focusing on.

Based on GDPR, key size selection is important. DES uses 56-bit key.

References

- NIST. (2001). Security Requirements for Cryptographic Modules. Available from: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf> [Accessed 15 July 2022]

```
1  from Crypto.Cipher import DES
2  from secrets import token_bytes
3
4  key = token_bytes(8)
5
6  def encrypt(msg):
7      cipher = DES.new(key, DES.MODE_EAX)
8      nonce = cipher.nonce
9      ciphertext, tag = cipher.encrypt_and_digest(msg.encode('ascii'))
10     return nonce, ciphertext, tag
11
12 nonce, ciphertext, tag = encrypt(input('Enter a message: '))
13 print(f'Ciphertext: {ciphertext}')
14
```

```
Enter a message: Life is short.
Ciphertext: b'\xe7\xc6\x07\xd1\x8c\xf5U!Z\xcbD\xc9\xcb\xec'
```