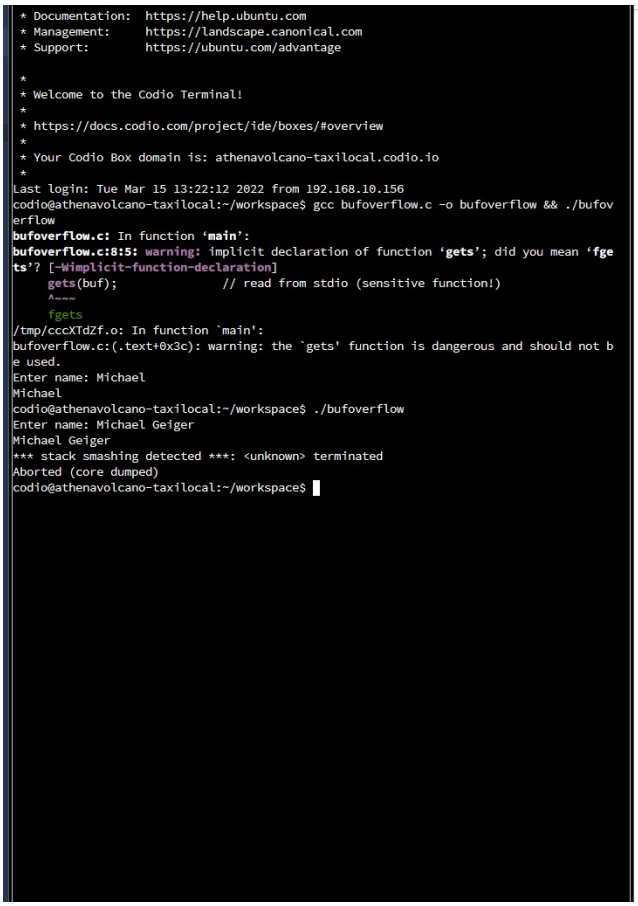
Exercise 1 – Buffer Overflow in C:



```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 *
 * Welcome to the Codio Terminal!
 *
 * https://docs.codio.com/project/ide/boxes/#overview
 *
 * Your Codio Box domain is: athenavolcano-taxilocal.codio.io
 *
Last login: Tue Mar 15 13:22:12 2022 from 192.168.10.156
codio@athenavolcano-taxilocal:~/workspace$ gcc bufoverflow.c -o bufoverflow && ./bufov
erflow
bufoverflow.c: In function 'main':
bufoverflow.c:8:5: warning: implicit declaration of function 'gets'; did you mean 'fge
ts'? [-Wimplicit-function-declaration]
     gets(buf);              // read from stdio (sensitive function!)
     ^~~~
     fgets
/tmp/cccXTdZf.o: In function `main':
bufoverflow.c:(.text+0x3c): warning: the `gets' function is dangerous and should not b
e used.
Enter name: Michael
Michael
codio@athenavolcano-taxilocal:~/workspace$ ./bufoverflow
Enter name: Michael Geiger
Michael Geiger
*** stack smashing detected ***: <unknown> terminated
Aborted (core dumped)
codio@athenavolcano-taxilocal:~/workspace$
```

1. Buffer Overflow Part I

Buffer Overflow in C

Remember to save your work to your GitHub Repository

In this example, you will compile and run a program in C. The program is already provided as bufoverflow.c - a simple program that creates a buffer and then asks you for a name, and prints it back out to the screen.

This is the code in bufoverflow.c:

```c
#include <stdio.h>

int main(int argc, char **argv)
{
char buf[8]; // buffer for eight characters
printf("Enter name: ");
gets(buf); // read from stdio (sensitive function!)
printf("%s\n", buf); // print out data stored in buf
return 0; // 0 as return value
}
```

Now use the rocket icon to compile and run the code. To test it, enter your first name (or at least the first 8 characters of it) you should get the output which is just your name repeated back to you.

Run the code a second time (from the command window this can be achieved by entering `./bufoverflow` on the command line). This time, enter a string of 10 or more characters.

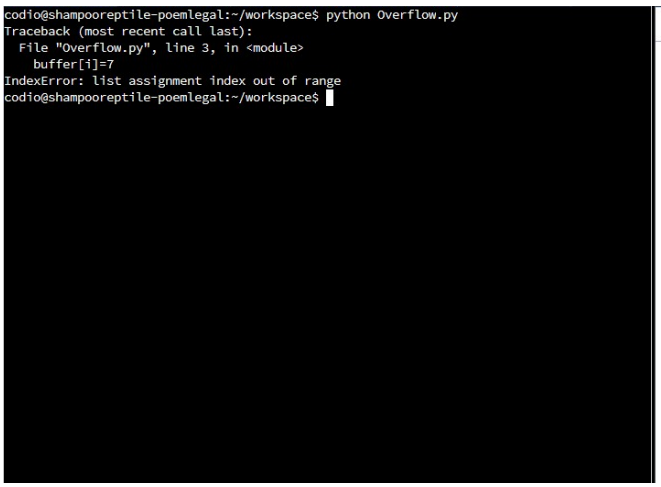
- What happens?
- What does the output message mean?

Now move on to Part II of this exercise - **Buffer Overflow in Python**

Be prepared to discuss your thoughts on both exercises at the next seminar session.

➔ If the input of the string is longer than 8 characters, the input gets dumped. This can be seen in the 'Aborted (core dumped)' message.

Exercise 2 – Buffer Overflow in Python:



```
codio@shampooreptile-poemlegal:~/workspace$ python Overflow.py
Traceback (most recent call last):
  File "Overflow.py", line 3, in <module>
    buffer[i]=7
IndexError: list assignment index out of range
codio@shampooreptile-poemlegal:~/workspace$
```

Collapse

1. Buffer Overflow Part II

Buffer Overflow in Python

Remember to save your work to your GitHub Repository

Now carry out a comparison of your work in C, with one in Python, following these instructions.

You will be using the file Overflow.py:

```python
buffer=[None]*10

for i in range (0,11):

    buffer[i]=7

print(buffer)
```

```
codio@shampooreptile-poemlegal:~/workspace$ pylint Overflow.py
************* Module Overflow
Overflow.py:4:0: C0303: Trailing whitespace (trailing-whitespace)
Overflow.py:5:0: C0304: Final newline missing (missing-final-newline)
Overflow.py:1:0: C0103: Module name "Overflow" doesn't conform to snake_case naming style (i
nvalid-name)
Overflow.py:1:0: C0114: Missing module docstring (missing-module-docstring)

--------------------------------
Your code has been rated at 0.00/10

codio@shampooreptile-poemlegal:~/workspace$
```

```
codio@shampooreptile-poemlegal:~/workspace$ flake8 Overflow.py
Overflow.py:1:7: E225 missing whitespace around operator
Overflow.py:2:17: E231 missing whitespace after ','
Overflow.py:3:14: E225 missing whitespace around operator
Overflow.py:4:1: W293 blank line contains whitespace
Overflow.py:5:14: W292 no newline at end of file
codio@shampooreptile-poemlegal:~/workspace$
```

→Pylint provides recommendations for design changes. Since the implementation with pylint was carried out after the code was changed, no further errors can be shown otherwise. However, Pylint can also detect errors in the code. A similar test was also carried out with Flake8.Flake8 shows exactly on which line : character the problem occurs.

Exercise 3 – Testing with Python:

```
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     product = 1
9     for i in range(1, n+1):
10      product = product * i
11    return product
```

2. Testing with Python

Exploring Linters to Support Testing in Python: Question 1

Run `styleLint.py`.

- What happens when the code is run?
- Can you modify this code for a more favourable outcome?
- What amendments have you made to the code?

➔ Before changing the code an error occurs because of missing indentation. To get a more favourable outcome the indentation was fixed and a more generic design was used.

```
codio@samuelperfect-actorpattern:~/workspace$ pylint pylintTest.py
************* Module pylintTest
pylintTest.py:26:13: E0001: Missing parentheses in call to 'print'. Did you mean print(encod
ed)? (<unknown>, line 26) (syntax-error)
codio@samuelperfect-actorpattern:~/workspace$
```

➔ Pylint shows errors (in this case an syntax error in line 26) and suggest solutions.

```
codio@samuelperfect-actorpattern:~/workspace$ flake8 pylintTest.py
pylintTest.py:5:1: W293 blank line contains whitespace
pylintTest.py:12:3: E111 indentation is not a multiple of 4
pylintTest.py:14:7: E111 indentation is not a multiple of 4
pylintTest.py:16:7: E111 indentation is not a multiple of 4
pylintTest.py:17:7: E111 indentation is not a multiple of 4
pylintTest.py:17:14: E225 missing whitespace around operator
pylintTest.py:19:7: E111 indentation is not a multiple of 4
pylintTest.py:23:11: E111 indentation is not a multiple of 4
pylintTest.py:24:11: E111 indentation is not a multiple of 4
pylintTest.py:26:14: W292 no newline at end of file
codio@samuelperfect-actorpattern:~/workspace$
```

4. Testing with Python

Exploring Linters to Support Testing in Python: Question 3

Ensure *flake8* is in your virtual box -

```
pip install flake8
```

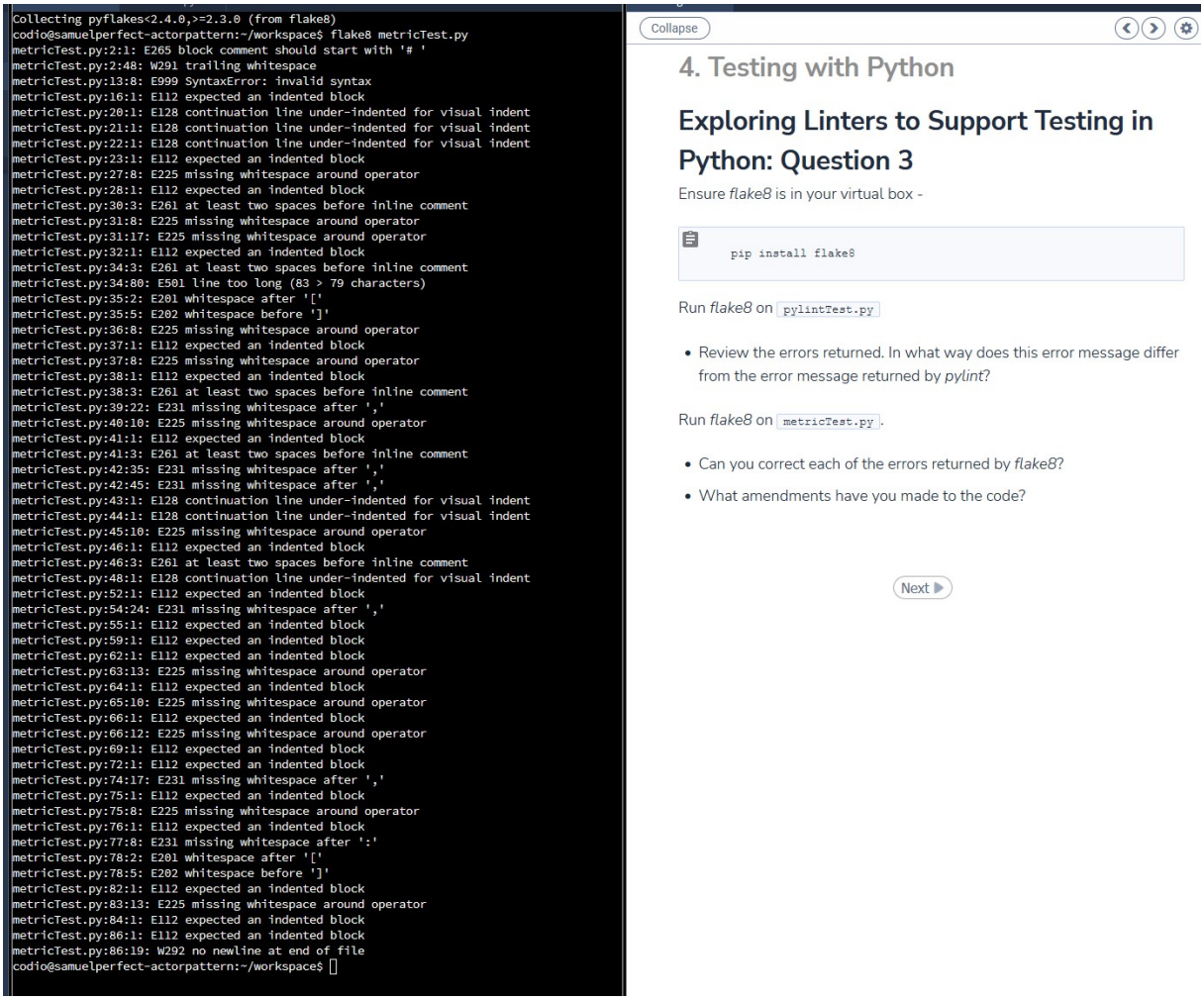
Run *flake8* on `pylintTest.py`

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

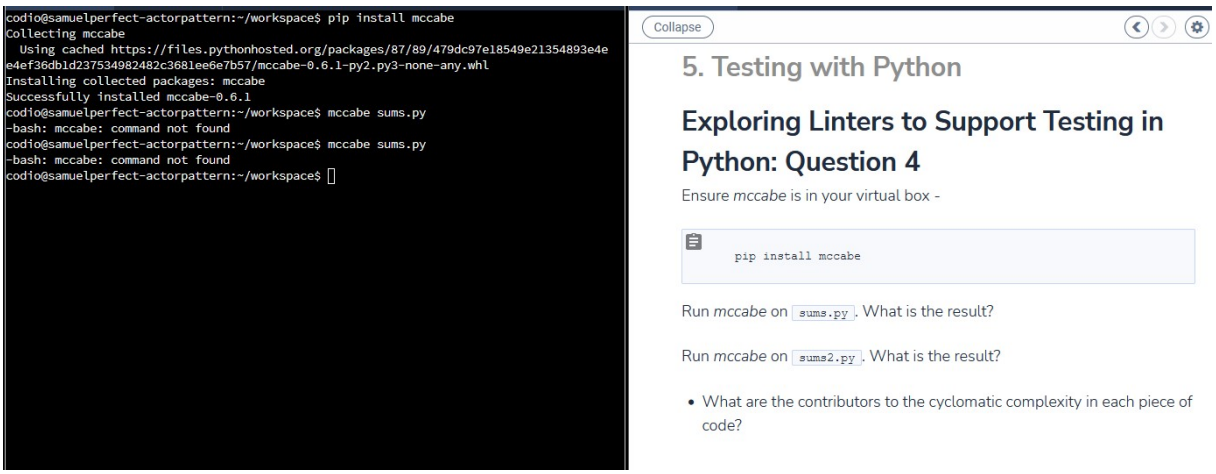
Run *flake8* on `metricTest.py`.

- Can you correct each of the errors returned by *flake8*?
- What amendments have you made to the code?

➔ Flake8 shows exactly in which line : character an error or an unexpected character like a missing space or one too much is found.



➔ To correct the errors returned by flake8, missing spaces need to be edited and those which are too many need to be deleted. Also indentations need to be fixed. Some of the listed errors can be ignored, such as command changes or some whitespaces (Sometimes it is better to separate some lines to make it better readable). Also the too long line error is a question of agreement and design and in should be decided out of the situation.



➔ mccabe made some trouble by using it (see above).

➔ McCabe checks the McCabe, also known as the cyclomatic, complexity. It will flag a problem if the function is considered to be too complex. Complexity occurs as a consequence of too much branching logic, which includes if/elif/else and for/while loops.