Part I: Exercise

In this example, you will compile and run a program in C using the **Codio workspace** provided (Buffer Overflow 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 (also available in the Codio workspace):

```
#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 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?

Part II: Exercise

Now carry out a comparison of this code with one in Python (Buffer Overflow in Python), following these instructions:

In the Codio workspace, you will be using the file called Overflow.py:

```
buffer=[None]*10
for i in range (0,11):
    buffer[i]=7
print(buffer)
```

- Run your code using: Python overflow.py (or use the codio rocket icon)
- What is the result?
- Read about Pylint at http://pylint.pycqa.org/en/latest/tutorial.html
- Install pylint using the following commands:

    pip install pylint (in the command shell/ interpreter)

- Run pylint on one of your files and evaluate the output:

    pylint your_file

- (Make sure you are in the directory where your file is located before running Pylint)
- What is the result? Does this tell you how to fix the error above?

Part 1: Answer

- C program:

```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
}
```

- Below is the output run on a machine running a Linux Ubuntu distribution, with the input string within the 8 character bounds:

```
codio@absent-opera:~/workspace$ gcc bufoverflow.c -o bufoverflow && ./bufoverflo
w
bufoverflow.c: In function ‘main’:
bufoverflow.c:8:5: warning: implicit declaration of function ‘gets’; did you mea
n ‘fgets’? [-Wimplicit-function-declaration]
     gets(buf);              // read from stdio (sensitive function!)
     ^~~~
     fgets
/tmp/ccVsUKtN.o: In function `main':
bufoverflow.c:(.text+0x3c): warning: the `gets' function is dangerous and should
 not be used.
Enter name: mike
mike
```

- Beneath is the output on the same machine with a character set larger than 8 characters:

```
codio@absent-opera:~/workspace$ gcc bufoverflow.c -o bufoverflow && ./bufoverflo
w
bufoverflow.c: In function ‘main’:
bufoverflow.c:8:5: warning: implicit declaration of function ‘gets’; did you mea
n ‘fgets’? [-Wimplicit-function-declaration]
     gets(buf);              // read from stdio (sensitive function!)
     ^~~~
     fgets
/tmp/cclJjXfL.o: In function `main':
bufoverflow.c:(.text+0x3c): warning: the `gets' function is dangerous and should
 not be used.
Enter name: michaelAmy
michaelAmy
*** stack smashing detected ***: <unknown> terminated
Aborted (core dumped)
```

- The program still outputs the entered string, however a stack overflow (stack smashing) is detected by the OS because the bounds of the variable are exceeded, and therefore the memory area allocated (the stack) where the variable is stored is exceeded. The program is aborted and the program memory area wiped (dumped)

Part 2 : Answer

- Python program:

```
1 buffer=[None]*10
2
3 for i in range (0,11):
4
5     buffer[i]=7
6
7 print(buffer)
8
```

- Output:

```
Traceback (most recent call last):
  File "C:\Users\Michael Botha\Desktop\test.py", line 5, in <module>
    buffer[i]=7
IndexError: list assignment index out of range
```

- The index is out of range because the list has been defined as only having 10 elements yet the for loop is trying to access 11

- PEP8 is the style guide for writing standardised python code
- Pylint is used to quickly and easily determine if code has captured the essence of PEP8 with the following format of output:

```
Output:
   Using the default text output, the message format is :
  MESSAGE_TYPE: LINE_NUM:[OBJECT:] MESSAGE
  There are 5 kind of message types :
  * (C) convention, for programming standard violation
  * (R) refactor, for bad code smell
  * (W) warning, for python specific problems
  * (E) error, for probable bugs in the code
  * (F) fatal, if an error occurred which prevented pylint from doing
  further processing.
```

- Pylint is run from the command prompt: pylint *program.py*
- It will evaluate the source code and return any deviations from the PEP style:

```
#!/usr/bin/env python3

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)
```

```
robertk01 Desktop$ pylint simplecaesar.py
************* Module simplecaesar
simplecaesar.py:3:0: W0301: Unnecessary semicolon (unnecessary-semicolon)
simplecaesar.py:1:0: C0114: Missing module docstring (missing-module-docstring)
simplecaesar.py:5:0: C0103: Constant name "shift" doesn't conform to UPPER_CASE naming style (invalid-name)
simplecaesar.py:9:0: C0103: Constant name "encoded" doesn't conform to UPPER_CASE naming style (invalid-name)
simplecaesar.py:13:12: C0103: Constant name "encoded" doesn't conform to UPPER_CASE naming style (invalid-name)

-----------------------------------
Your code has been rated at 7.37/10
```

- Running pylint on the above index errored program:

```
C:\Users\Michael Botha>cd desktop

C:\Users\Michael Botha\Desktop>pylint test.py
************* Module test
test.py:1:16: C0303: Trailing whitespace (trailing-whitespace)
test.py:4:22: C0303: Trailing whitespace (trailing-whitespace)
test.py:6:15: C0303: Trailing whitespace (trailing-whitespace)
test.py:8:13: C0303: Trailing whitespace (trailing-whitespace)
test.py:1:0: C0114: Missing module docstring (missing-module-docstring)

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

- The code does not pick up the logical error in the code only syntactical deviations from the PEP standard