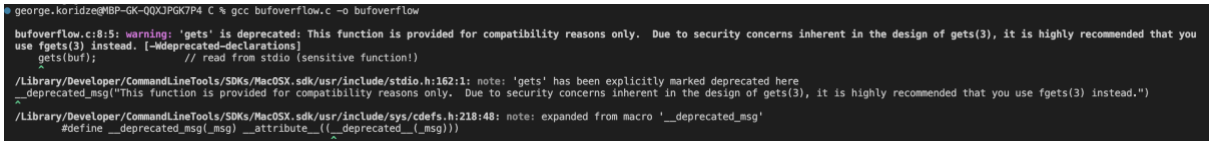
Activity - Exploring Python tools and features

Part 1



Expectation:

First Test:

- The program should have read input using gets(buf) and stored it in the 8-character buffer (buf).
- It then prints the input correctly, as the input size matches the buffer's capacity (8 characters).
- This illustrates normal operation where no buffer overflow occurs.

Second Test:

- Entering a string longer than 8 characters overflows the buffer. The additional characters exceed the allocated memory for buf.
- The output might print unexpected or corrupted data and can potentially crash the program.

In reality:

The use of gets() is inherently unsafe because it does not limit the input size, making it prone to buffer overflows. Modern compilers often warn against its use, recommending safer alternatives like fgets(). So Visual Studios resulted in the error shown above and did not allow me to run it. I changed the code to ‘fgets()’:

bufoverflow.c

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    char buf[8]; // buffer for eight characters
    printf("Enter name: ");
    fgets(buf, sizeof(buf), stdin); // Safe alternative to gets()
    printf("%s\n", buf); // Print out data stored in buf
    return 0; // Return 0 as return value
}
```

Which resulted in:

```
george.koridze@MBP-GK-QQXJPGK7P4 C % gcc bufoverflow.c -o bufoverflow

george.koridze@MBP-GK-QQXJPGK7P4 C % ./bufoverflow

Enter name: Giorgi
Giorgi

george.koridze@MBP-GK-QQXJPGK7P4 C % ./bufoverflow

Enter name: GiorgiVarME
GiorgiV
```

Part 2

```
george.koridze@MBP-GK-QQXJPGK7P4 python % python3 Overflow.py

Traceback (most recent call last):
  File "/Users/george.koridze/Desktop/Essex/SSD/ePortfolio/Unit3/python/Overflow.py", line 3, in <module>
    buffer[i]=7
    ~~~~~~^^^
IndexError: list assignment index out of range
```

1) This error occurs because the range(0, 11) loop tries to access indices 0 to 10, but buffer is only allocated to hold 10 elements (indices 0–9). Accessing buffer[10] is invalid.
2) Install ‘pip3 install pylint’

3) Run command and identify errors:

```
george.koridze@MBP-GK-QQXJPGK7P4 python % pylint Overflow.py

zsh: command not found: pylint
george.koridze@MBP-GK-QQXJPGK7P4 python % 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: C0114: Missing module docstring (missing-module-docstring)
Overflow.py:1:0: C0103: Module name "Overflow" doesn't conform to snake_case naming style (invalid-name)

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

4) Modify the loop to ensure it respects the buffer bounds:

```
buffer = [None] * 10
for i in range(len(buffer)):  # Use len(buffer) to restrict access within bounds
    buffer[i] = 7
print(buffer)
```

5) Error handled:

```
george.koridze@MBP-GK-QQXJPGK7P4 python % python3 Overflow.py

[7, 7, 7, 7, 7, 7, 7, 7, 7, 7]
```

Summary:

Error Handling:

- In Python, exceeding the bounds of a list immediately raises an exception (IndexError), making the issue explicit.
- In C, buffer overflows can silently overwrite adjacent memory, potentially leading to undefined behavior or security vulnerabilities.

Static Analysis (Pylint vs. Compilers):

- Python's Pylint flags logical issues before runtime.
- In C, static analysis tools (e.g., GCC warnings or dedicated tools like Valgrind) are needed to catch such errors.