Week 3: COMP-801 - Integrated Computing Practice

Agenda

- Feedback to **Active Reading 2**
- Review of tools, concepts, and techniques
  - `git` commands, `docstring`, test cases, `f-string`
  - function defintion and call, flow of execution, iteration
- **Dare2Design** activity
- Help session:
  - Lab 2 collaborative learning work
  - Markdown basics

Feedback to AR2

- See AR2 Feedback in the Canvas Module for Week3
- AR assignment MUST be completed BEFORE starting on
  - program development assignments (**lab#** or **h#**)
- New deadline for ARs starting this week: **Monday midnight**

📖 Remote git Repositories

- What is a remote repository?
- How does a local repository know what remote repository it is linked to?
- What does `push` command do?
  - What is `origin`?
  - What is `main`?

📖 git Commands

- How do we prepare to commit changes we made in the working directory?
- How do we commit the changes?
  - What should the commit message say?
- How do we display the commit history?
- Where is the commit history stored?

📖 Document Your Code

- What is a **docstring**? What's the **docstring**'s syntax?
- Where do we write docstrings in a Python module?
- What are the components of a docstring?
  - when it documents a module?
  - when it documents a function (or method) definition?
  - when it documents a class definition?
- How do we use VS Code **restructuredText** to format **docstrings**?

`docstring` Example

```
def my_filter(words, prefix):
    """
    Return a list of strings in `words` that do not start with `prefix`.

    :param words: list of strings that do not contai white spaces
    :param prefix: string with no white spaces
    :return: list of strings
    Example:
    my_filter(['ball', 'break', 'bad', 'bet'], 'ba'])
    returns ['break', 'bet']
    """
```

📖 Test Cases

- What is a test case?
- What is a "happy path" test case?
- What is an "edge case"?
- What is an illegal argument test case?

📖 Python f-string Example

```python
"""Test input list with three elements."""
input_lst = [3, 2, 7]
expected_result = 2
actual_result = minimum(input_lst)
err_msg = (
    f'minimum({input_lst}) must be {expected_result}, '
    f'not {actual_result}'
)
assert expected_result == actual_result, err_msg
```

📖 Function Definition

- What is the syntax of a function definition?
  - Identify all components
  - Explain what each does
- Give an example of a function definition
- How is a function definition documented in a docstring?

📖 Function Call

- What is the syntax of a function call?
  - Identify all components
  - Explain what each does
- How is a function call different from a function definition?
- Give an example of a function call
  - How is a function call used? Where?

📖 Example of Function Definition and Call

```
def increment(operand):
    increased = operand + 1
    return increased

return_value = increment(2)
print(return_value)
```

function name

parameters

function body (indented)

return value

function call

arguments

📖 Flow of Execution

- What is the flow of execution?
- What is the default flow of execution?
- What statements change the flow of execution? How?

📖 Iteration

- What is the syntax of the `for` ... `in` ... loop?
  - Name and explain each component.
- What is the syntax of the `while` ... loop?
  - Name and explain each component.

📖 `for` … `in` … loop

```
1 item_prices = [3.56, 2.53, 8.92]
2 bill_amount = 0.0
3 for price in item_prices:
4     bill_amount = bill_amount + price
5 print(bill_amount)
```

iteration variable

iterable (str, list, dict, etc.)

loop body (indented)

next statement
(outside the loop)

Dare2Design Activite

- Individually
  - Write the code based on the design description
- In teams of 2 members
  - Review and discuss your designs
- Teams report out to the entire class

📖 Accumulation Pattern

- What is the **accumulation pattern**?
- What are the components of the pattern?
- Give an example of the accumulation pattern to illustrate it.

💪 📖 Design Descriptions - The Idea

Just the idea:

- The problem's input has a list of `words` and a string `prefix`.
- To solve the problem
  - Define a list variable that will return the result.
  - Iterate through each word in the `words` list
  - Check if each word starts with the `prefix`

    If it does not, append the word to the result

3. After the iteration ends, return the results

💪 📖 Design Descriptions - Steps 1 & 2

Define Accumulator and Iterate

- **Step 1**: Create variable `modified_list` and initialize it with empty list.
  - This is done to store the words which doesn't start with the given `prefix`
- **Step 2**:
  - Start a `for-loop` to iterate over the list `words`
  - Define the loop variable `a_word` of type string so that every word can be accessed to perform necessary operations

💪 📖 Design Descriptions - Steps 3, 4, 5

Transform and Accumulate

Inside the `for-loop`, at each iteration through the loop:

- **Step 3**: Check if `a_word` in the list starts with given `prefix`.
  - Use a `str` operation or method to do the checking.
- **Step 4**: If the word doesn't start with given prefix then append `a_word` to `modified_list`
- **Step 5**: Repeat the same steps - 3, 4 - for every `a_word` in `words`

💪 📖 Design Descriptions - Step 6

Return

- **Step 6**: After the for loop execution is completed return the `modified_list` .

Lab2: Getting Started

Use collaborative learning to do Lab2, working with the assigned peer.

- Verify you have all the tools you need on your laptop
  - `git-bash` (or `terminal`), `VS Code`, `git`, `GitHub` account
- Get the Lab2 codebase from the GitHub Classroom invitation link

Lab2: Document Your Code

- Follow **Lab 2** instructions to complete the modules' `docstrings`.

Version control this development step.

Lab2: Write the Tests

Write testing functions for the `only_integers()` method.

- Refer to **Testing Requirements** in the Test-Driven and Incremental Development resource
- Implement the 2nd testing function following the example of the 1st testing function.

Version control this development step.

Lab2: Draft the Design

Write the design of the `only_integers()` method.

- Refer to **Design Requirements** in the Test-Driven and Incremental Development resource
- Write the design in DESIGN.md

Version control this unit of development.

Lab2: Write the Implementation

Teams work on the implementation of `only_integers()`

- Refer to **Implementation Requirements** in the Test-Driven and Incremental Development resource
- Write the implementation in `sentence.py`

Version control this unit of development.

Lab2: Finish Development

Teams work to finish the development

- Check the `Problems` panel and fix the warnings and errors
- Use `black formatter` extension to help with automatic fixing
- Install and run `pycodestyle` to check and fix additional errors

Version control this unit of development.

Markdown

- Lightweight markup language with plain-text-formatting syntax
- We use it for readme, design documents, change logs, etc.

Markdown Tutorial

Markdown Summary

Title Untitled-1

```
# Title

## Heading

Normal text, in **bold** and *italics*.

```python
# Some Python code
def function_name():
    return
```
- Also, lists
  1. first
  2. second
- (Ordered and unordered, even nested)
```

Preview Untitled-1

Title

Heading

Normal text, in **bold** and *italics*.

```python
# Some Python code
def function_name():
    return
```

- Also, lists
  1. first
  2. second
- (Ordered and unordered, even nested)