Spring 2022

Class 07 - Control Flow Practice

Today's Goals

1. Announcements

2. Practice function writing

3. Practice function diagramming

Announcements

- EX03 - Released on Tuesday: Structured Wordle - Due Monday
- Tuesday 2/8: Watch async lessons in lecture and get help.
  - Want to watch lecture content with help around? Bring your headphones and move through content with UTAs around to answer questions!
- Thursday 2/10: Quiz 1 - Primary emphases: loops, logic, functions, lists
- Email Pairings in Sakai's PostEm Tab
  - Copy BOTH of your UTAs on e-mails
  - If you feel inclined to attach a screenshot or copy paste something-- *come to office hours!*
  - UTAs (nor myself) can help with technical or code problems via e-mail.

Function #1

- Create a file in your `lessons` directory named `love_functions.py`

- Define a function with the following signature expectations:

1. Function Name: `love`
2. Parameters
   - `name: str`
3. Return Type: `str`
4. Docstring: `"""Given a name as a parameter, returns a loving string."""`

- In the function body, have a single return statement:
  - `return f"I love you {name}!"`

Expected implementation:

```python
def love(name: str) -> str:
    """Given a name as a parameter, returns a loving string."""
    return f"I love you {name}!"
```

How to use in the Python REPL:

- In the terminal, begin a Python REPL:

```
python
```

- Import the function:

```
>>> from lessons.love_functions import love
```

- Call it:

```
>>> love("Mom")
>>> love(", my dear friend")
```

Function #2

- Still in the same file lessons/love_functions.py, declare a function named `spread_love`, with the following signature expectations:

1. Function name: `spread_love`
2. Two parameters:
   - `to: str`
   - `n: int`
3. Return type: `str`
4. Docstring: `"""Generates a string that repeats a loving message n times."`

- Implementation:

1. Declare a string variable named `love_note` and assign it the empty string.
2. Declare a counter variable that is initialized to zero.
3. Write a `while` loop that will iterate while your counter variable is less than your parameter `n`. Don't forget to increment your counter variable!
4. Inside the `while` loop, concatenate `love_note`'s current value to the result of calling the `love` function with `to` as the argument, then concatenate `"\n"` for a line break.
5. After the while loop completes, return the generated `love_note`

Expected implementation:

```python
def spread_love(to: str, n: int) -> str:
    """Generates a string that repeats a loving message n times."""
    love_note: str = ""
    i: int = 0
    while i < n:
        love_note += love(to) + "\n"
        i += 1
    return love_note
```

How to use in the Python REPL:

- In the terminal, begin a Python REPL:

```
python
```

- Import the function:

```
>>> from lessons.love_functions import spread_love
```

- Call it:

```
>>> spread_love("Mom", 100)
>>> print(spread_love("Mom", 100))
```

Challenge Question #1: What returned when the following function definition is called with...

`mystery(4)`

```
def mystery(n: int) -> str:
    """A useless function."""
    i: int = 0
    while i < n:
        if i % 2 == 1:
            return "ooh"
        i += 1
    return "ahh"
```

Diagramming Practice

```python
"""CQ  A main Function."""


def main() -> None:
    """The program's entrypoint."""
    print("main()")
    y: float = double(2.0)
    print(halve(y))


def halve(x: float) -> float:
    """Hlave a value."""
    print(f"halve({x})")
    return x / 2.0


def double(x: float) -> float:
    """Double a value."""
    print(f"double({x})")
    return x * 2.0


if __name__ == "__main__":
    print("__name__ is '__main__'")
    main()
```

```python
"""CQ A main Function."""


def main() -> None:
    """The program's entrypoint."""
    print("main()")
    y: float = double(2.0)
    print(halve(y))


def halve(x: float) -> float:
    """Hlave a value."""
    print(f"halve({x})")
    return x / 2.0


def double(x: float) -> float:
    """Double a value."""
    print(f"double({x})")
    return x * 2.0


if __name__ == "__main__":
    print("__name__ is '__main__'")
    main()
```