Diagram Practice

Today is a Paper + Pencil or Tablet + Pencil day...
please keep laptops stowed away!

COMP110 - CL03

2024/01/23

Announcements

- QZoo - First Quiz is Thursday
  - Quiz Expectations on Course Site
  - Practice Problems Post Tonight
  - Randomized, Assigned Seats will Post by 11:59pm Tomorrow (Canvas Announce)

- EX01 - Tea Party Planner - Due Monday 1/29
  - Recommendation: Complete parts 0 - 3 as quiz practice!
  - You can submit parts 0 - 3 to the autograder to confirm correctness

```python
"""A program with a *two* function calls."""


def perimeter(length: float, width: float) -> float:
    """Calculates the perimeter of a rectangle."""
    return 2.0 * length + 2.0 * width


def square_perimeter(side: float) -> float:
    """Calculates the perimeter of a square."""
    return perimeter(length=side, width=side)


print(square_perimeter(side=4.0))
```

Let's review the problem from LS05 Due Yesterday

Trace & Diagram these two, short code listings.

```
3   """Be aware: venomous snake_case..."""
4
5
6   def addup(x: int, y: int) -> int:
7       """Add two numbers."""
8       return x + y
9
10
11  print(add_up(x=10, y=20))
```

```
3   """Be aware: name resolution"""
4
5
6   def addup(i: int, h: int) -> int:
7       """Add two numbers."""
8       return j + h
9
10
11  print(addup(i=10, h=20))
```

```python
"""A meandering little foraging program."""


def gather(bushes: int) -> int:
    """A function where creatures gather berries."""
    print("Gather!")
    return bushes * 4


def adventure(ponds: int) -> int:
    """Begin the adventure by hopping across ponds."""
    print("Adventure!")
    return gather(bushes=ponds * 2)


print(adventure(ponds=1))
```

Trace this Code Listing with a Diagram

```
"""Composing strings!"""


def bang(y: str) -> str:
    """Add excitement!"""
    print("bang!")
    return y + "!"


def qué(z: str) -> str:
    """What?"""
    print("¿qué?")
    return "¿" + z + "?"


print(qué(z=bang(y="bear")))
```

Trace this Code Listing with a Diagram

Named Constants

Putting a Name to "Magical Numbers"

- Programs often involve *constant values* in computations and other places
  - For example: π, e, SALES_TAX, GAME_TITLE, FOOT_IN_INCHES and so on
- Rather than sprinkling *literal values* for these constants in *many places* through a program, often called "Magic Numbers", defining **named constants** is encouraged
- By convention, named constants are ALL_CAPITAL_LETTERS with multiple words separated by underscores.
- For example:
  - PI: float = 3.14159
  - SALES_TAX: float = 0.07
- When defined at the *global level* the named constant is available throughout your Python module
  - Why? ... *Name resolution rules!*

Trace this Code Listing with a Diagram

```python
"""Functions of a circle..."""

PI: float = 3.14


def main() -> None:
    """Entrypoint of Program"""
    print(perimeter(radius=1.0))
    print(area(radius=1.0))
    return None


def area(radius: float) -> float:
    """Calculate area of a circle"""
    return PI * radius**2


def perimeter(radius: float) -> float:
    return 2 * PI * radius


main()
```