
Practice with Conditionals
Today is a Paper + Pencil or Tablet + Pencil day...
please keep laptops stowed away!
COMP110 - CL05
2024/02/01

Announcements

- Quiz 0 - Please review and understand questions and diagram points missed immediately, these concepts will be important to have an understanding of on moving forward.
  - Not sure about a correct answer? Come see us in office hours for conceptual help! We *love* working through quiz-style conceptual problems. Please do not ask questions about content in regrade requests.

Warm-up: What is the printed output?

Hint: Try diagramming!

```
1   def foo(x: int) -> int:
2       """A nonsensical function..."""
3       if x == 0:
4           return x + 1
5
6       if x == 1:
7           print(f"x: {x - 1}")
8
9       if x > 0:
10          return x * 2
11
12      return x
13
14
15  print(foo(x=-1))
16  print(foo(x=1))
```

Diagram the Following Program

```
"""Calling to and fro..."""


def ping(i: int) -> int:
    print("ping: " + str(i))
    if i <= 0:
        return i
    else:
        return pong(i=i - 1)


def pong(i: int) -> int:
    print("pong: " + str(i))
    return ping(i=i - 1)


print(ping(i=2))
```

Diagram the Following Program

```python
"""Mysterious 'rev' from source (src) to destination (dest)!"""


def rev(src: str, i: int, dest: str) -> str:
    """You happen upon a magical lil function..."""
    if i >= len(src):
        return dest
    else:
        return rev(src=src, i=i + 1, dest=src[i] + dest)


print(rev(src="lwo", i=0, dest=""))
```

Tuples

Fixed-length *sequences* of values.

Built-in Aggregation Functions

How could you implement an aggregation function?

Based on what we know now, functions and conditionals, by using recursion...

Diagram

```python
def sum(values: tuple[float, ...], i: int, total: float) -> float:
    """Sum all values in a tuple."""
    if i >= len(values):
        return total
    else:
        return sum(values=values, i=i+1, total=total+values[i])


print(sum(values=(1.0, 2.0, 3.0), i=0, total=0.0))
```

Diagram

```
1   def min(values: tuple[float, ...], i: int, smallest: float) -> float:
2       """Find the smallest value in a tuple."""
3       if i >= len(values):
4           return smallest
5       else:
6           if values[i] < smallest:
7               return min(values=values, i=i+1, smallest=values[i])
8           else:
9               return min(values=values, i=i+1, smallest=smallest)
10
11
12  print(min(values=(3.0, 1.0, 2.0), i=1, smallest=3.0))
```