COMP 110

CL06 - Practice with Functions

Practice Writing Functions

Write a mimic function: you input a string and it returns the same string back to you

- Function name: mimic
- Parameters: my_words: str
- Return type: str
- Doc string: """Given the string my_words, outputs the same string"""

Try calling it!

Expected Code:

```python
def mimic(my_words: str) -> str:
    """Given the string my_words, outputs the same string"""
    return my_words
```

Calling it:

```python
mimic("Hello!")

print(mimic("Hello!"))

my_words: str = "Hello!"
response: str = mimic(my_words)
print(response)
```

Practice Writing Functions

Write a different mimic function: you input a string and an index and it returns the letter at that index. If the index is too high for the string length, return “Index too high”.

E.g. mimic_letter(“hello”,0) returns “h”, mimic_letter(“howdy”,2) returns “w”, mimic_letter(“hi”,3) returns “Index too high”

Function name: mimic_letter

- Parameters: my_words: str, letter_idx: int
- Return type: str
- Doc string: """Outputs the character of my_words at index letter_idx"""

Expected Code:

```python
def mimic_letter(my_words: str, letter_idx: int):
    """Outputs the character of my_words at index letter_idx"""
    if letter_idx >= len(my_words):
        return("Index too high")
    #If we made it here, that means the letter_idx is valid
    return my_words[letter_idx]
```

Memory Diagrams: Change

- We will be replacing arrows with id numbers!
- Why?
  - Every object in python has an **id** number associated with their location in memory (also called an “address”)
  - We use arrows to represent variables that are *references* to locations in memory on the heap.
  - Using **id** is a cleaner and more literal representation of this.
- When objects stored on the heap (e.g. functions) are initialized, label them with a heap id, starting with **id:0** and counting up
- When referring to an object on the heap, instead of drawing an arrow, state their id number (**id: 0**).
- When *accessing* a variable name that holds a heap id, look at its associated id on the heap
  (if variable x has **id:0** as its value, look at the object on the heap with **id:0**).

Memory Diagram Example

```python
"""Example functions to learn definition and calling syntax"""

def my_max(num1: int, num2: int) -> int:
    """Returns the maximum value out two numbers"""
    if num1 >= num2:
        return num1 + 0
    else: #number1 < number2
        return num2

max: int = my_max(1,12)
other_max: int = my_max(13,3)
print(other_max)
```

Memory Diagram

```python
def main():
    """Main code of program"""
    y: float = double(2.0)
    print(halve(y))

def halve(x: float) -> float:
    """Returns half the value of x"""
    print(f"halve({x})")
    return x / 2.0

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

main()
```