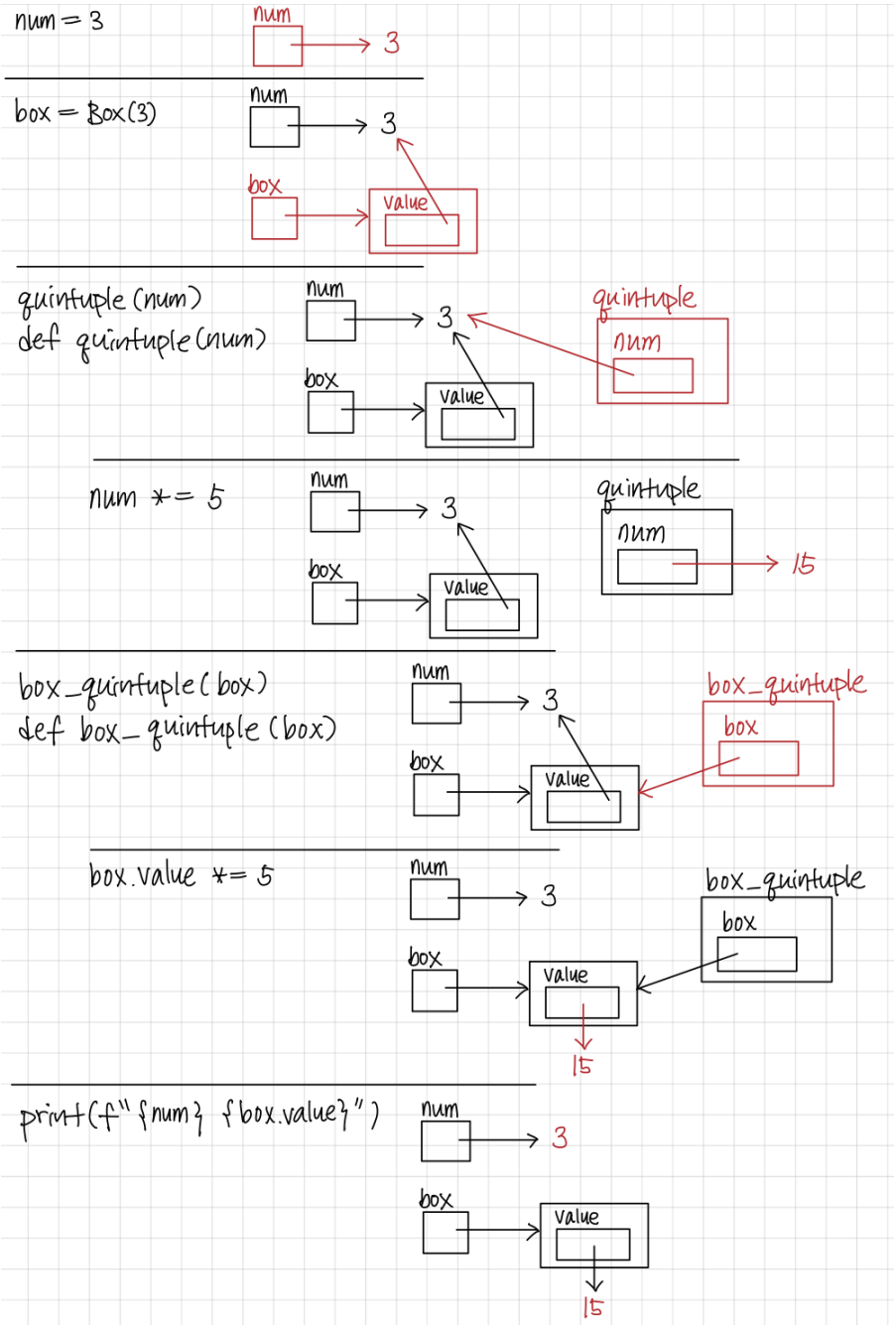
Homework 1

CS 131, Fall 2024
Carey Nachenberg

Soyeon Kim, 106292803

PYTH1

Part A



`num = 3` creates an integer object 3, and `num` holds the object reference. Inside `quintuple`, the parameter `num` holds the same object reference of 3. Since integers are immutable, `num *= 5` creates a new integer object 15 and holds the new object reference without changing the original object 3. After the function call, the integer object 15 will be released and the outside `num` will still hold 3.
`box = Box(3)` creates a new instance of `Box`, where `box.value = 3`, and `box` holds the object reference. Inside `box_quintuple`, the parameter `box` holds the same object reference. Since `Box` instances are mutable, `box.value *= 5` changes the value in place.

Part B

1. joke6, joke3, joke4
2. joke6, joke2

PYTH2

It is because `Foo` and `str` class has `__len__` method overloading. When `len` function is called, it will automatically find and call the `__len__` function of the parameter. However, `int` class does not have `__len__` method. Since Python is a dynamically typed language, it checks for the presence of a given method or attribute at runtime.

PYTH3

Part A

```
class DuckA:
   def quack(self):
     pass
```

Part B

```
class DuckB(Duck):
   def __init__(self):
     pass
```

Part C

`is_duck_a` is more Pythonic because Python typically relies on an object's behavior rather than on its type. `is_duck_a` utilizes duck typing by directly calling their methods without checking the object's type beforehand.

PYTH4

Part A

```
nums[i:i+k]
not max_sum or sum > max_sum
```

Part B

`nums[:k]`
`nums[i]`
`nums[i+k]`

PYTH5

Part A

```
class Event:
    def __init__(self, start_time, end_time):
        if start_time >= end_time:
            raise ValueError
        self.start_time = start_time
        self.end_time = end_time
```

Part B

```
class Calendar:
    def __init__(self):
        self.__events = []
    def get_events(self):
        return self.__events
    def add_event(self, event):
        if type(event) is not Event:
            raise TypeError
        self.__events.append(event)
```

Part C

It is because Python does not call the superclass Calendar. We need to explicitly call the super class' constructor inside the AdventCalendar constructor, or override the `__events` variable and `get_events` method.

```
class AdventCalendar(Calendar):
    def __init__(self, year):
        super().__init__()
        self.year = year
```

```
class AdventCalendar(Calendar):
    def __init__(self, year):
        self.year = year
        self.__events = []
    def get_events(self):
        return self.__events
```

PYTH6

Part A

I expect C-Lang to be faster because the script will be compiled into machine code ahead of time. On the other hand, I-Lang will be translated into machine code line by line during runtime.

Part B

Jamie will have the server running first because she can write and run the script immediately without compiling the code.

Part C

Connie can execute I-Lang Jamie's script because I-Lang is an interpreted language and the script is not tied to any specific machine architecture. So when Connie runs the script, the interpreter in her machine will handle its execution.
However, Connie cannot execute Tim's C-Lang executable. Without an emulator or translator, his pre-compiled Intel binary will not run natively on Connie's machine.

PYTH7

Numpy is faster because of its optimized pre-compiled C code and vectorization. C code enables more efficient memory management and processor instructions. Vectorization reduces the overhead of repeatedly interpreting Python code inside loops.

PYTH8

Part A

```
j.joke = I dressed as a UDP packet at the party. Nobody got it.
Joker.joke = I dressed as a UDP packet at the party. Nobody got it.
self.joke = I dressed as a UDP packet at the party. Nobody got it.
Joker.joke = I dressed as a UDP packet at the party. Nobody got it.
self.joke = Why do Java coders wear glasses? They can't C#.
Joker.joke = How does an OOP coder get wealthy? Inheritance.
j.joke = Why do Java coders wear glasses? They can't C#.
Joker.joke = How does an OOP coder get wealthy? Inheritance.
```

Part B

Same as Part A.

```
j.joke = I dressed as a UDP packet at the party. Nobody got it.
Joker.joke = I dressed as a UDP packet at the party. Nobody got it.
self.joke = I dressed as a UDP packet at the party. Nobody got it.
Joker.joke = I dressed as a UDP packet at the party. Nobody got it.
self.joke = Why do Java coders wear glasses? They can't C#.
Joker.joke = How does an OOP coder get wealthy? Inheritance.
j.joke = Why do Java coders wear glasses? They can't C#.
Joker.joke = How does an OOP coder get wealthy? Inheritance.
```