Introductory Computer Science

Week 4 — Object Oriented Programming

Dann Sioson
dj.sioson@alum.utoronto.ca

August 4, 2019

Table of Contents

Administrative Items

A bit more on tracing

Introduction to OOP

Terminology

Objects

self

"Security" in Python

__init__ and __str__

Table of Contents

Administrative Items

A bit more on tracing

Introduction to OOP

Terminology

Objects

self

"Security" in Python

__init__ and __str__

Administrative items

- A bit more on tracing

Administrative items

- A bit more on tracing
- Exercises

Administrative items

- A bit more on tracing
- Exercises
- Problem Sets

Administrative items

- A bit more on tracing
- Exercises
- Problem Sets
- Assignment

Itinerary

1. ~~Introduction and Git~~
2. ~~Programming with Python~~*
3. ~~Memory Model and Debugging~~*
4. Object Oriented Programming
5. Object Oriented Programming
6. Test day
7. Linked Lists

* = What will be tested

Table of Contents

Administrative Items

A bit more on tracing

Introduction to OOP

Terminology

Objects

self

"Security" in Python

__init__ and __str__

A bit more on tracing

```
def f1(x):          | def f2(y, x):     |  def f3(x, y):
    return x + 1    |     z = f1(y)     |      c = x
                    |     print(y + x)  |      x = f1(y)
                    |     return z + x  |      y = f1(x)
                    |                   |      z = f2(c, y) % 5
                    |                   |      return y, z, x
```

```
>>> x = f3(4, 5)
>>> print(x)
?
```

What's printed?

Table of Contents

Administrative Items

A bit more on tracing

Introduction to OOP

Terminology

Objects

self

"Security" in Python

__init__ and __str__

From week 2: Some types

- A lot of the data structures and types that was presented from week 2 was pretty neat

From week 2: Some types

- A lot of the data structures and types that was presented from week 2 was pretty neat
- But, if you program for a while these types can be boring (especially in the software engineering realm)

From week 2: Some types

- A lot of the data structures and types that was presented from week 2 was pretty neat
- But, if you program for a while these types can be boring (especially in the software engineering realm)
- What if we made our own? If there were only one way...

OBJECT

OBJECT ORIENTED

OBJECT ORIENTED PROGRAMMING

Introduction

- Functions were the initial focus as it a way in which we received output from input
- Object Oriented Approach:
  - `my_object.method(input)` has its own methods and data

Table of Contents

Administrative Items

A bit more on tracing

Introduction to OOP

Terminology

Objects

self

"Security" in Python

__init__ and __str__

Terminology

- Class: `Type` of an object. Something that is abstract but does not necessarily have something of substance

Terminology

- Class: `Type` of an object. Something that is abstract but does not necessarily have something of substance
- Object: Something that is `instantiated`

Terminology

- Class: `Type` of an object. Something that is abstract but does not necessarily have something of substance
- Object: Something that is `instantiated`
- Method: A function the belongs to a class

Terminology

- Class: `Type` of an object. Something that is abstract but does not necessarily have something of substance
- Object: Something that is `instantiated`
- Method: A function the belongs to a class
- We somewhat saw this:
  - `my_str = str(12.57)`
  - `my_str.ljust(10)`
  - `my_str` is an *object*, of the `str` class, and we called the ljust *method* on it

Table of Contents

Administrative Items

A bit more on tracing

Introduction to OOP

Terminology

Objects

self

"Security" in Python

__init__ and __str__

Leading Example

Person

Object Creation

- Begin with `class ClassName`
  - CamelCase is conventional for ClassNames

Object Creation

- Begin with `class ClassName`
  - CamelCase is conventional for ClassNames
- Defining a method `method_name`

Object Creation

- Begin with `class ClassName`
  - CamelCase is conventional for ClassNames
- Defining a method `method_name`
- After this, you are capable of creating a new object
  - `my_object = ClassName()`
- And the object can now access the methods defined
  - `my_object.method_name()`

Table of Contents

Administrative Items

A bit more on tracing

Introduction to OOP

Terminology

Objects

self

"Security" in Python

__init__ and __str__

Self

- `self` is the representation of the object itself, but not the class' representation

Self

- `self` is the representation of the object itself, but not the class' representation
- `self` generally has representative attributes (e.g. as a person, you have certain amount of fingers, you have a certain eye colour, you have certain medical conditions etc.)

Self

- `self` is the representation of the object itself, but not the class' representation
- `self` generally has representative attributes (e.g. as a person, you have certain amount of fingers, you have a certain eye colour, you have certain medical conditions etc.)
- Using `self` is a way to manipulate instantiated objects of the class within defined methods

Self

- `self` is the representation of the object itself, but not the class' representation
- `self` generally has representative attributes (e.g. as a person, you have certain amount of fingers, you have a certain eye colour, you have certain medical conditions etc.)
- Using `self` is a way to manipulate instantiated objects of the class within defined methods
- For every method in a class (in Python), it is necessary to have `self` as the first parameter

Table of Contents

Administrative Items

A bit more on tracing

Introduction to OOP

Terminology

Objects

self

"Security" in Python

__init__ and __str__

“Security” in Python

- As you probably know, Python is an absolutely wild language

“Security” in Python

- As you probably know, Python is an absolutely wild language
- There’s barely any constraints! You don’t even have to put semi-colons at the end of lines

“Security” in Python

- As you probably know, Python is an absolutely wild language
- There’s barely any constraints! You don’t even have to put semi-colons at the end of lines
- Going back to the previous example, would you want anybody to access your medical records?

“Security” in Python

- As you probably know, Python is an absolutely wild language
- There’s barely any constraints! You don’t even have to put semi-colons at the end of lines
- Going back to the previous example, would you want anybody to access your medical records?
- In Python, to combat this is using an underscore
  - `self._blood_type = "O"`

Table of Contents

Administrative Items

A bit more on tracing

Introduction to OOP

Terminology

Objects

self

"Security" in Python

__init__ and __str__

Built-In Methods

- __init__
  - Initialize: This is known as a constructor method (a method that returns an instantiation of an object)
  - Every class must have one (in Python)
  - Defines the code that runs when we first create a new object of this type
- __str__
  - Return what you want to output when an object of this class is cast to a string (or printed)