Google igniteCS

An Introduction to Python

Part IV: Classes & Objects

Introduction

- **Mentors:** Corin and Sufyan
- **Goals:**
  - Review
  - Learn about **classes** and **objects**
  - Object focused activities
  - Discuss project ideas for next lecture!
- **Any questions before we start?**

What have we done so far?

- **Variables**
  - Strings, integers
- **Flow Control**
  - Conditionals (if-elif-else)
  - Loops (while, for)
- **Functions**
  - Built-in and User-defined Functions
- **Packages**
  - Math and Random Packages
- **Data Structures**
  - Lists

Try it yourself!

Review Exercise: Let’s make a bank!

1. Create a new file called **`bank.py`** by typing:

   ```
   idle3 bank.py &
   ```

2. Complete the following exercise in this file:

   a. `Use variables to store a list of accounts and balances`

   b. `Write three functions`

      - `One that takes an account name and initial deposit and adds that amount to the proper variables`
      - `One that takes in an account name and returns its balance`
      - `One that takes in two account names and an amount, and transfers that amount from one to another`

3. Access these functions from the **`python3`** interpreter

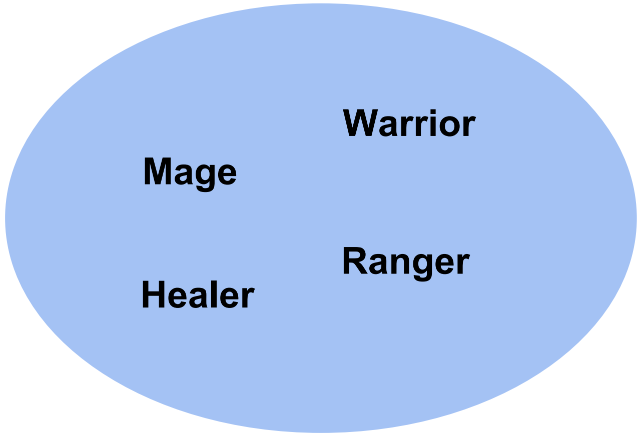
This isn’t good.

why?

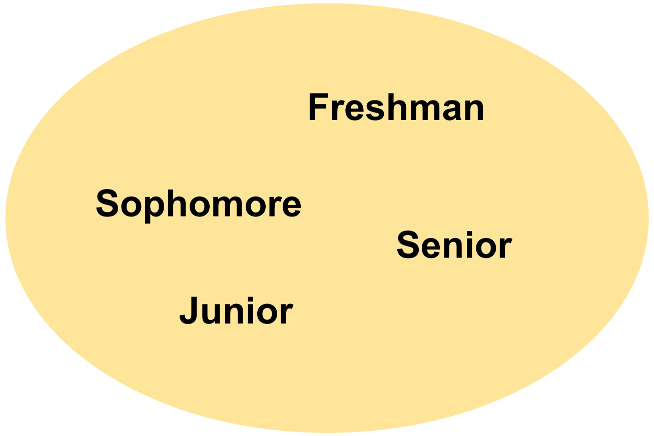
Introducing Classes & Objects

Classes

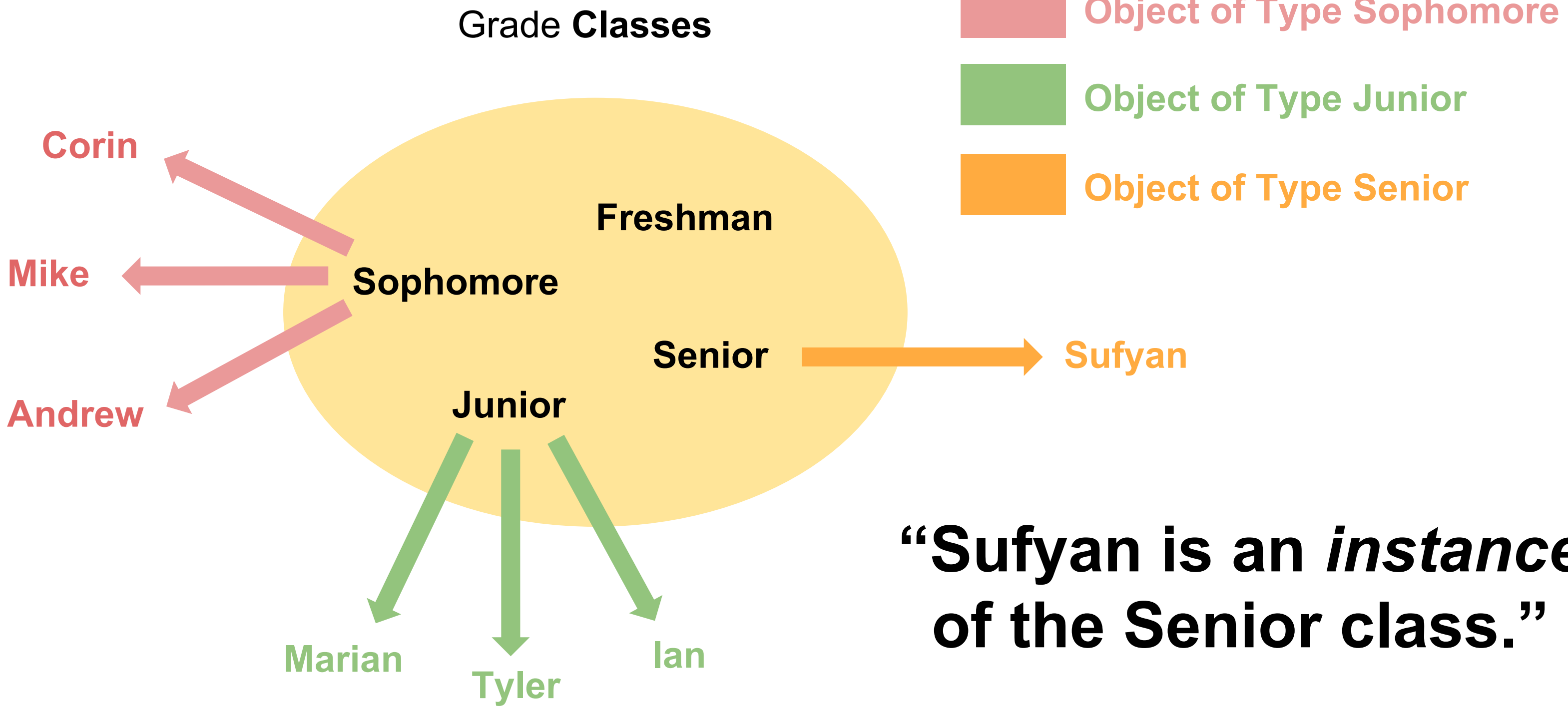
Video Game Character **Classes**



Grade **Classes**



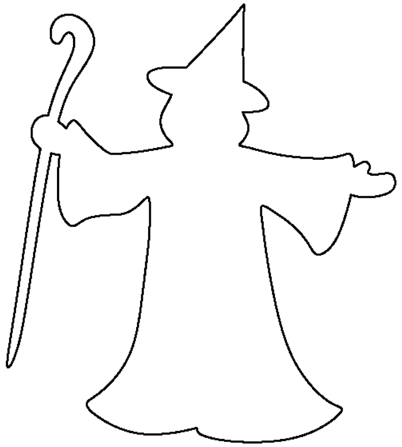
Objects



“Sufyan is an *instance* of the Senior class.”

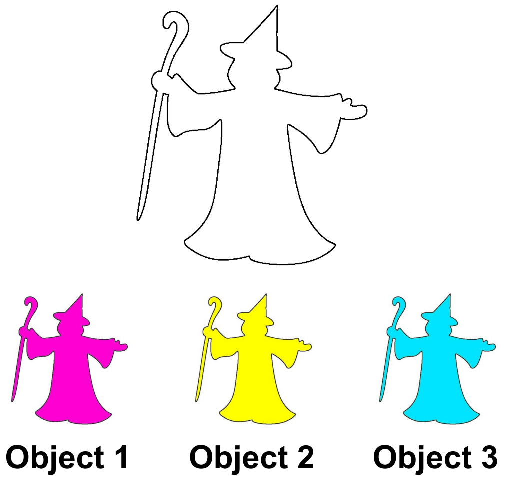
Formal Definition of a Class

- A **class** is a template for creating objects
- It contains **variables** (i.e. **attributes**) we need to describe the state of our object and **functions** (i.e. **methods**) that describe our object's behavior



Formal Definition of an Object

- An **object** is created from our class
    - It’s essentially a bundle of more specific information (i.e. attributes that have been set) and functions (i.e. methods)



Objects have attributes

These are the *intrinsic* properties of the object.

For example, a **`Person`** object would have the following intrinsic properties:

- name
- age
- birthday
- favorite color
- favorite Adele song

Classes have constructors

- The constructor of a class constructs an object and sets (i.e. “initializes”) the attributes of that object

Class Name

Constructor

```
class Person:

  def __init__(self, name, age):
        self.name = name
        self.age = age
```

we use the **self** keyword to reference the object we are currently in

Creating a Person Object

```
>>> person_object = Person(“Sufyan”, 21)
```

```
def __init__(self, name, age)
    self.name = name
    self.age = age
```

```
def __init__(self, “Sufyan”,
21)
    self.name = “Sufyan”
    self.age = 21
```

Accessing Attributes of the Person Class

- We can access an object’s attributes using the dot: **.**

```
>>> person_object = Person(“Sufyan”, 21)

>>> print(person_object.name)
    Sufyan

>>> print(person_object.age)
    21
```

Objects also have methods

- We can define functions that act upon the object’s attributes
- For example, we can define a **`greet`** method in the **`Person`** class that makes the object say their name

```
def greet(self):
    print(“Hello, my name is ” + self.name)
```

An Outline of Our Class So Far

- We would put this method inside of a class definition:

Constructor

```
class Person:

    def __init__(self, name, age):
        self.name = name
        self.age = age


  def greet(self):
      print(“Hello, my name is ” + self.name)
```

Methods

Calling a Method

```
>>> person_object = Person(“Sufyan”, 21)

>>> person_object.greet()

Hello, my name is Sufyan
```

```
def greet(self):
    print(“Hello, my name is ” + “Sufyan”)
```

Mutator methods

Finally, there are methods that change (i.e. mutate) attributes in our objects. These methods are called **mutators**. Let's define a **`happy_birthday`** method that increments our **`Person`** object's age by one.

```
def happy_birthday(self):
    self.age = self.age + 1
```

Calling a Method

```
>>> person_object = Person(“Sufyan”, 21)

>>> print(person_object.age)

    21

>>> person_object.happy_birthday()

>>> print(person_object.age)

    22
```

Complete Person Class

```
class Person:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print("Hello, my name is " + self.name)

    def happy_birthday(self):
        self.age = self.age + 1
```

Review

Class	A **class** is a template for creating objects, defining what information we need to understand the state of the object (attributes) and what functions that object needs to use (methods)
Constructor	A **constructor** is a specific method which we use to create an object of our class
Object	An **object** bundles together data and functions that operate on that data
Attribute	The data contained within an object are called **attributes**
Method	The functions contained within an object are **methods**

Review Exercise: Objects!

Try it yourself!

1. Create a new file called **`some_objects.py`** by typing:

   ```
   idle3 some_objects.py &
   ```

2. Complete the following exercise in this file:

   a. `Write three classes: Circles, Rectangles, and Triangles`

      - `You should define` **`constructors`** `for these classes which take in appropriate values (i.e. Circle's constructor should take in a radius, Rectangle's constructor should take two side lengths, and Triangle's constructor should take in three side lengths). These values should be stored as` **`attributes.`**

      - `Write a get_area()` **`method`** `for each class which returns the area of the shape by using the` **`attributes`** `we stored in the constructor and the appropriate area formula (e.g. Circle.get_area() ->` πr^2`)`

3. Create a bunch of objects using these classes with different parameters and print their areas

Summary

- **Classes** are the templates for our objects
- Classes are used to create (i.e. “instantiate”) **objects**, which have both attributes and methods
- **Attributes** are variables containing data stored in our objects
- **Methods** are functions in our objects that operate on attributes

Coming Up: Games | Cyber-security Hackathon

- Next session is on **Saturday December 2nd**
- Choose 1 of 2 options
    1. Make a small game using Python’s game plug-in
    2. Participate in cyber-security hackathon