



CS 1112: Introduction To Programming

Introduction to Python Classes

Creating our own data types!

Dr. Nada Basit // `basit[at]Virginia[dot]edu`

Friendly Reminders

- Your **safety** and **comfort** is important!
 - If you choose to wear a mask you are welcome to do so
 - *We will interpret wearing a mask as being considerate and caring of others in the classroom (not that you are sick), and realize that some may choose to mask to remain distanced*
- Remember to always be **kind, respectful, supportive, compassionate and mindful of others!** 😊
- Be an **active** participant in your learning!
You're welcome and **encouraged** to ask questions during class!
- If you feel **unwell**, or think you are, **please stay home**
 - *Contact us! We will work with you!*
 - Get some rest 😊
 - View the recorded lectures – *please allow 24-48 hours to post*



Announcements

- **PA04** is due by 11:00pm on Wednesday (*March 19 ~ After Spring Break*)!
 - Submit on Gradescope: your **.py** file, and a **PDF** of your reflection
 - Please be mindful about submitting the right kind of files (.py file and .pdf file) as well as submitting the .py file that is named correctly (see assignment document for full details)
 - A note about submitting on Gradescope: you can submit an **UNLIMITED** number of times prior to the deadline. Look at the score you got, if you have some points taken off, that's ok, go back and fix your code and resubmit! Do this as often as you like BEFORE the assignment deadline. You cannot resubmit after the deadline.
 - **REMEMBER ALSO:** You have a grace period of 24 hours to submit your PAs!
- **Quiz 5** will come out this week (probably 3/5) and due Monday (*March 17 ~ After Spring Break*)

Announcements : Exam 1

- **Exam 1** has been **graded!** Many thanks to our wonderful TAs who helped get this exam graded as fast as possible 😊
- **Mean score:** 85 (out of 100 points)!
- This class is about **learning** and **improving**. We want you to learn from your mistakes by performing **Exam 1 error corrections**.
- **Exam 1** error corrections (10 points back) due by **March 26, 2025 (11pm)**
 - In-person (Professor or TA) during office hours (or schedule time with me)
 - In-person (Professor or TA) before/after class
- **Exam 1 grader error** corrections due by **March 26, 2025 (11pm)**
 - Fill out the form “**Regrade Request – Exam 1**” (under Assignments)

Classes

A form of encapsulation

A means to create a custom (complex) data type!!

Classes

- Introduction to structure
- Encapsulation
- Creation of custom (non-native) data types
- Building regions of code larger than functions
 - Functions exist inside classes
 - There are “special” functions that are specific to classes / creation of a data type
- Example on how to use classes and how to create instances of these classes

Constructor (`__init__`)

```
class Alien:
    # fields: name, numArms, planet

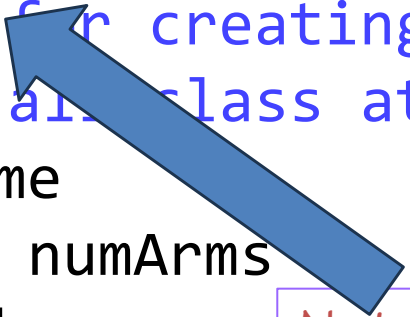
    def __init__(self, name, numArms, home): # constructor
        # responsible for creating objects of Aliens (this class)
        # must handle all class attributes (fields)
        self.name = name
        self.numArms = numArms
        self.planet = home
```

A **constructor** is a special kind of method (in Object Oriented vocabulary, functions are called methods)
The parameter list defines the rest of the class (class **attributes**)

Constructor (`__init__`)

```
class Alien:
    # fields: name, numArms, planet

    def __init__(self, name, numArms, home): # constructor
        # responsible for creating objects of Aliens (this class)
        # must handle all class attributes (fields)
        self.name = name
        self.numArms = numArms
        self.planet = home
```



Note: **"self"** is always the first parameter
"self" refers to the current instance of a class

Given classes are like **blueprints**, and you can make **many** copies (instances) of this object, "self" is used to describe the **current object** calling this method ("whose name"?, etc...)

Printing function (`__str__`)

```
def __str__(self):  
    # to-string method - display the state of the obj  
    # (also useful for testing class implementation)  
    return self.name + ' ' + self.planet + ' ' + str(self.numArms)
```

Note: this method is invoked when you use **print**!

```
=====
```

```
def function_x(self, a, b, c):  
    # Other methods (behaviors) exist within the class  
    # All objects of this class exhibit the same behaviors
```

Generally, functions are a “**behavior**” of the object. If object is student, a behavior could be “enroll_in_course” at a University.... And so on.

Example: Shape Class

```
class Shape:
    def __init__(self, xcor, ycor): # constructor
        self.x = xcor
        self.y = ycor

    def __str__(self): # to-string method
        return 'x: ' + str(self.x) + ' y: ' + str(self.y)

    def move(self, x1, y1):
        self.x = self.x + x1 # move x-axis
        self.y = self.y + y1 # move y-axis
```

Let's Look At Some Examples

- `classStructure.py`
- `classExamples.py`
- Alien example (if time)

Reminder: CS Laptop Loaner Program

- This course requires students to have a **laptop**
- I realize that not everybody might have one (nor necessarily need one for their desired major / path...)
- If you do not have a laptop for any reason... *not to worry!*
- The CS department's Systems staff has a notebook / laptop loaner program and will be able to loan you a notebook / laptop computer for the duration of the semester if you don't have one or if you cannot afford one.
 - Also available if your laptop is broken and under repair, we can arrange for you to receive a loaner laptop for a week or two until your own laptop is fixed

Interested? Link: https://www.cs.virginia.edu/wiki/doku.php?id=cs_laptop_loaner

I am happy to be your sponsor. Please let me know.