CoGrammar

GitHub Workshop 4: Classes and Career Goals

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Lecture Housekeeping

- The use of disrespectful language is prohibited if asking a question. This is a supportive, learning environment for all – please engage accordingly! **(FBV: Mutual Respect.)**
- No question is 'silly' – **ask away!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: **Open Class Questions**

Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

What is the purpose of the __init__ method in a Python class?

A. Initialise the class variables
B. Define the class methods
C. Create a new instance of the class
D. Terminate the class instance

What keyword is used to define a method in Python?

A. func
B. define
C. function
D. def

Recap of Week 10: OOP

Classes

- We can create classes in python using the ‘class’ keyword

Class Attributes

- Attributes can be assigned to classes that allow us to store data to use within the class

Class Instances

- We can create instances of classes with each instance having different values stored inside of their attributes

Recap of Week 10: OOP

Defining a Class

```
class MyClass():

    def __init__(self, value1, value2):
        self.value1 = value1
        self.value2 = value2
```

Creating an Instance

```
new_instance = MyClass(3,5)
```

Recap of Week 10: OOP

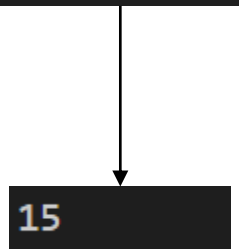
Adding a Method

```
class MyClass():

    def __init__(self, value1, value2):
        self.value1 = value1
        self.value2 = value2

    def get_total(self):
        return self.value1 * self.value2
```

Recap of Week 10: OOP

Calling the Method

```
new_instance = MyClass(3,5)

print(new_instance.get_total())
```



```
15
```

Recap: Methods, __init__(), and self

- ★ Methods are functions associated with objects of a particular class. Recall that lower() is a string method, meaning that it's called on string objects.
- ★ Also, notice that methods come after the object.

Recap: Methods, __init__(), and self

- ★ We create objects by calling the class name as a function. This function is referred to as a constructor function (or constructor, or abbreviated as ctor, pronounced “see-tore”) because it constructs a new object.
- ★ We also say the constructor instantiates a new instance of the class.

Recap: Methods, __init__(), and self

- ★ Calling the constructor causes Python to create the new object and then run the __init__() method.
- ★ Classes aren't required to have an __init__() method, but they almost always do.
- ★ The __init__() method is where you commonly set the initial values of attributes.
- ★ When a method is called on an object, the object is automatically passed in for the self parameter.

Recap: Methods, __init__(), and self

- ★ The rest of the arguments are assigned to parameters normally.
- ★ You don’t have to name a method’s first parameter self; you can name it anything. But using self is conventional.

ZooWonders

- **Background:** Your local zoo need a program that will provide a lively and instructive environment where guests can engage with virtual animals and discover more about their habitats and habits.

- **Challenge:** Construct a virtual zoo administration system with classes and objects. In the program, define each animal's traits and behaviours by representing it as an object.

- **Objectives:**
  - Create classes for various animal types with attributes and behaviours.
  - Instantiate objects for different animals within the virtual zoo.
  - Develop a user interface for visitors to interact with the virtual animals.

Creating Animals

This is a basic animal class representing a lion. We define a constructor method to add a name, age, weight and description for the lion. Remember: Not all the animals you create will have the same attributes.

```
class Lion():

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

    def make_sound(self):
        print("ROAR!")
```

Building a Pack

We can create a pride of lions, starting by adding all the details into a text file. Each line represents one lion with a name, age, weight, and description.

```
Jax; 9; 185; Big and Fluffy
Big Joe; 15; 200; Very Big and Very Angry
Spot; 4; 136; Young and Curious
```

Building a Pack

We can then run the function shown below to read all the data from the text file and populate a list representing our lion pride. This function can be improved to build packs of other animals too.

```
lion_pride = []

def build_pride(pride):
    with open('lions.txt', 'r') as file:
        for line in file:
            split_line = line.strip().split(';')
            lion = Lion(split_line[0], split_line[1], split_line[2], split_line[3])
            pride.append(lion)
    return pride
```

Output Example

```
Welcome to ZooWonders!
Please select an area of the zoo you would like to visit:
1. Birds of Paradise
2. Big Cat Park
3. Reptile Park
4. Giant Ocean
2
```



```
Animals in Big Cat Park
1. Lion
2. Tiger
3. Leopard
4. Cheetah
Please select an animal above to learn more:
```

Dashboard Output Example

Here is an example of how we can display information about the animals the user chooses.

```
We have 3 Lions in the Big Cat Park

1. Big Joe

Age = 12
Weight = 200kg

Big Joe is a fierce lion and the leader of the pack and the king of the park. There
won't be a lot of comotion when Big Joe is around as one big roar makes every
animal at ZooWonders go silent for a few seconds.
```

ZooWonders

Construct a virtual zoo administration system with classes and objects. In the program, define each animal's traits and behaviours by representing it as an object.

Important features:

1. **Menu:** Give the user a user-friendly interface to work with and navigate through your virtual Zoo.

1. **Animal Information:** Allow the user to view animals and information about them such as their age, name, habits, diet, etc.

1. **Interaction:** Allow the user to engage with the animals such as virtually feeding them and having the animals respond.

1. **User Experience:** Try to provide the user with a zoo experience. Think about zoo interactions and try to give you program a similar feel.

Advanced Challenge:

- When a user selects a part of the zoo to view, allow them to choose to do a tour that will take them through each animal one by one.

Summary

ZooWonders

- ★ Create a virtual zoo experience where users can view information about animals and interact with them in different ways.

Classes

- ★ Build a class for each animal type to model their own unique attributes and behaviours.

User Experience

- ★ Keep the user experience in mind and try to add elements to your program to make it feel like a real zoo.

How do you make a class variable private in python?

A. Use the keyword ‘private’ before the variable
B. private_str = private(“Hello”)
C. Add a single underscore before the variable name
D. Add two underscores, one before and one after the variable name.

True or False: A class is an object.

A. True
B. False

Questions and Answers

Questions around the Case Study