Advance Python Pt 2

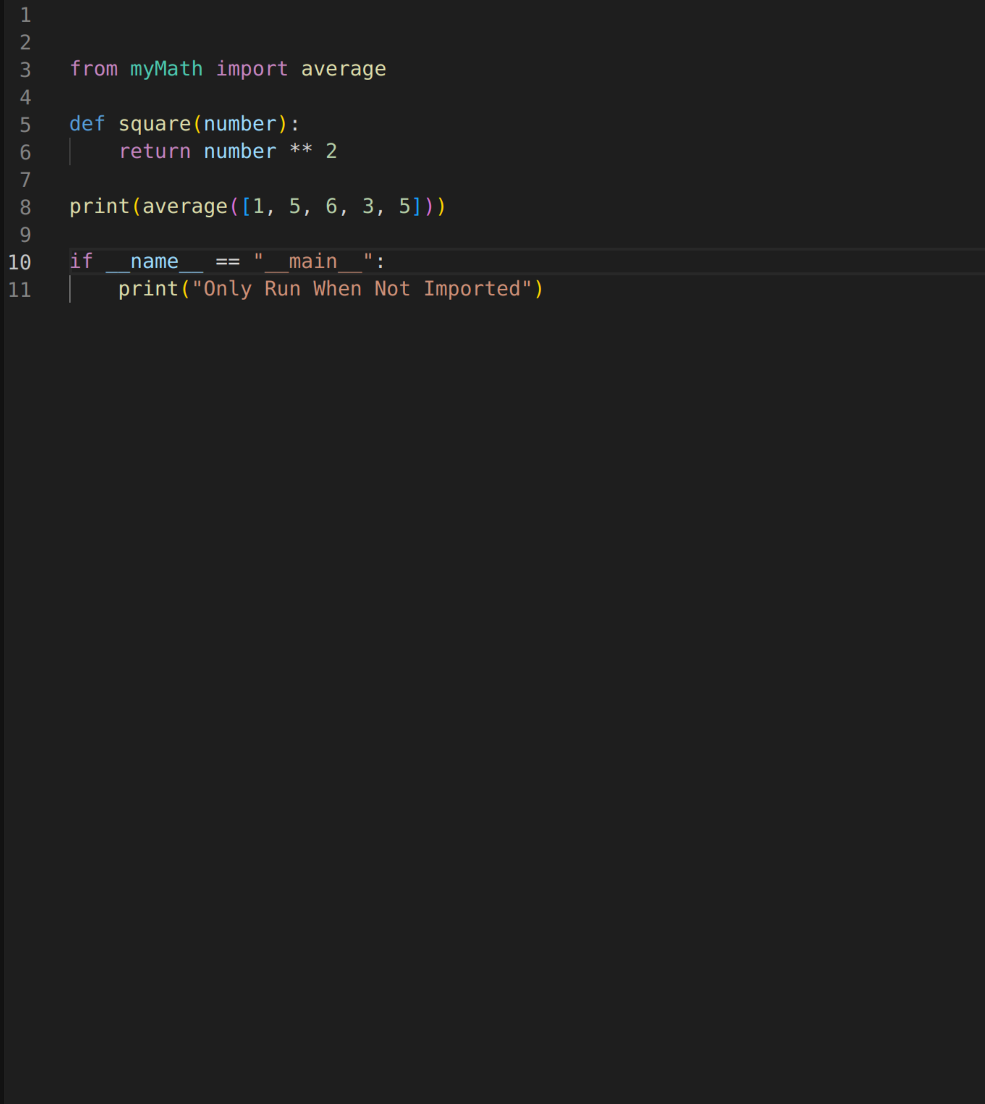
Research Data Services

By Avery Fernandez

Plans For Today

Learning the basics to python

- Classes how they work and their use cases
- Using Classes How to create and use classes



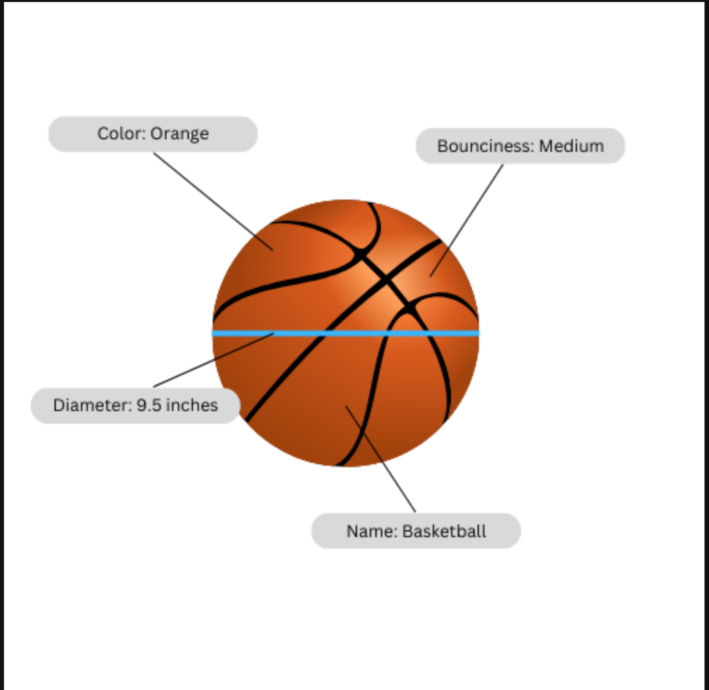
```python


from myMath import average

def square(number):
    return number ** 2

print(average([1, 5, 6, 3, 5]))

if __name__ == "__main__":
    print("Only Run When Not Imported")
```

Classes

What are they

Classes are the transition from just programming instructions to creating whole objects

Let’s look at this ball



First Class

Recreating the basketball in python

```
class ball:
    name = "basketball"
    diameter = 9.5
    bounciness = "medium"
    color = "orange"
```

Now if we want to use it, we can call upon it

```
myBall = ball()
print(myBall.name)
```

Output

```
basketball
```

Changing attributes

Now we can change any of the attributes at any time

```
myBall.color = "purple"
print(myBall.color)
```

Output

```
purple
```

The changes only apply to that object

```
secondBall = ball()
print(myBall.color)
print(secondBall.color)
```

Output

```
purple
orange
```

Exercise One

Looking at the world through python

- Pick any object around you
- Create a python class with atleast 4 of its attributes
- Print out all the attributes

Exercise One Code

```python
class waterBottle:
    material = "aluminum"
    brand = "yeti"
    company = "Mechanics Bank"
    lid = "black"
    fluid = "water"

myWater = waterBottle()
print(myWater.material, myWater.brand, myWater.company, myWater.lid, myWater.fluid)
```

Creating Custom Objects

Now that we know how to create an object, it would be nice to customize its attributes on creation

This is where initializers come in helpful

The initializer statement will set the attributes based on the parameters given

```python
class ball:
    def __init__(self, name, diameter, color, bounciness):
        self.name = name
        self.diameter = diameter
        self.color = color
        self.bounciness = bounciness

myBall = ball("basketball", 9.5, "orange", "medium")
print(myBall.name)
```

Output

```
basketball
```

NOTE self is just a variable name to reference the object, it can be anything from self to ball

Print Statements

What happens when we try to print out an object

```
print(myBall)
```

```
<__main__.ball object at 0x7f6a2e483f40>
```

We can add a parameter to the class so that printing it would do a specific function

```
class ball:
    def __str__(self):
        return f"The {self.name} is a(n) {self.color} ball with a diameter \
of {self.diameter} in and {self.bounciness} bounciness"

myBall = ball("basketball", 9.5, "orange", "medium")
print(myBall)
```

```
The basketball is a(n) orange ball with a diameter of 9.5 in and medium bounciness
```

Class Functions

We can create functions that are specific to the class and its data

```python
class ball:
    def ballVolume(self):
        return f"{4 / 3 * 3.14 * (self.diameter / 2) ** 3} in ^ 3"

myBall = ball("basketball", 9.5, "orange", "medium")
print(myBall.ballVolume())
```

```
448.69291666666663 in ^ 3
```

Exercise 2

Take the object we made from exercise 1

Make its attributes be made on creation

Create a print statement that explains the object and its attributes

Create a function that does something specific to the object

Exercise 2 Code

```python
class waterBottle:
    def __init__(self, material, brand, company, lid, fluid, height, diameter):
        self.material = material
        self.brand = brand
        self.company = company
        self.lid = lid
        self.fluid = fluid
        self.height = height
        self.diameter = diameter
    def __str__(self):
        return f"The {self.brand} water bottle is a(n) {self.material} bottle with a {self.lid} lid and \
{self.company} logo. It is filled with {self.fluid} and has a max capacity of {self.volume()} in ^ 3"
    def volume(self):
        return 3.14 * (self.diameter / 2) ** 2 * self.height

myWater = waterBottle("aluminum", "yeti", "Mechanics Bank", "black", "water", 8, 3.5)

print(myWater)
print(myWater.volume())
```