Codetrain

Introduction to Python Programming

Hello!

I am Awal Mubarak

Software Engineer, Codetrain.

Codetrain

1

Basic Concepts

Uderstanding the basics of python

Your First Program

```python
print("Hello World")
```

Run this code in the python shell

Datatypes

- Strings - “my name”, ‘my name’
- Integers- -5, 7, 0, 900
- Float - 7.76556, 0.98677
- Boolean - True, False

Variables

```
name = "awal"
number = 90
available = True
name = "Mubarak"
```

Cannot contain spaces or start with special characters except underscore

Comments

```
# this is a comment
```

Operators

They give you the power to manipulate or verify two or more data

Arithmetic Operators

Addition +

Subtraction -

Multiplication *

Division /

String Operators

Concatenation +

Comparison Operators

Equal to	==
Not Equal to	!=
Greater than	>
Less Than	<

Boolean Logic

and

or

Codetrain

2

Control Structures

This is where the magic occurs

If, elif and else statements

```
if expression:
    statements
```

e.g

```
if 3 > 8:
    print("big")
```

```
elif expression:
    statements
```

e.g

```
if 3 > 8:
    print("big")
elif 3 < 10:
     print ("small")
```

```
else:
    statements
```

E.g

```
if 3 > 8:
    print("big")
else:
     print ("small")
```

Lists

A list is created using **square brackets** with **commas** separating items.

```
names = [“zak”, “rich”, “ray”]
index -    0       1       2
```

A list can include several different data types

List Functions & Operations

```
ages = [“two”, 19, “nine”, True, 63]
```

```
ages[0]
ages[2]
ages[4]
```

Assigning values

```
ages[0] = 6
ages[3] = “five”
```

Codetrain

More Object Types

Tuples

Tuples are very similar to lists, except that they cannot be changed. Also, they are created using **parentheses**, rather than square brackets.

```
languages = (“python”, “php”, “java”)
index -              0          1         2
```

```
languages[0] = “c++”
```

Dictionaries

Dictionaries are data structures used to map arbitrary keys to values.

Lists can be thought of as dictionaries with integer keys within a certain range.

```
students = {“Sam” :  28, “Bob”: 21, “Naa”: 25}
students["Sam"]
students["Bob"]
```

For loops

```
words = ["hello", "world", "spam", "eggs"]

for word in words:
    print(word + "!")
```

For Iterating a list or running a code a certain number of times

Codetrain

Functions and Modules

Code Reusability

Code Reuse

- To decrease code size
- Easy code maintenance

Principle

- DRY (Don’t Repeat Yourself)
- WET (We Enjoy Typing or Write Everything Twice)

You’ve already Seen a Function

```
print("hello!")
```

`print` Is the function name and `"hello!"` Is argument for this function.

Every word with parentheses in front of it is a function. The word in front of the parentheses is function **names**, and the comma-separated values inside the parentheses is function **arguments**.

How do we create Functions?

functions are created / defined by using the `def` keyword.

```
def my_function():
    print("spam")
    print("spam")
    print("spam")

my_function()
```

Codetrain

Quick Tip

Just like variables, functions need to be defined before being called. Calling a function before defining it will throw an error.

Functions with Arguments

Sometimes you want a function that takes some arguments and then do something with it.

```
def say_my_name(name):
    print(name)


say_my_name(“Awal”)


//Awal
```

Return Statement

Sometimes you want your function to return a value so that you use it for other operations.

```
def add(num1, num2):
    answer = num1 +  num2
    return answer


result = add(8,9)
```

Modules

Modules are pieces of code that other people have written to fulfill common tasks, such as generating random numbers, performing mathematical operations, etc.

Codetrain

4

Object Oriented Programming

Everything is an Object, including you. **Ops!**

Every Object has **functions** and/or **properties**

In Real Life, Every object has functions and properties. Eg.

Dog

Properties

- Has 4 legs
- Has a tail
- Has a head
- Has name
- Has owner

Functions

- Can Eat
- Can Run
- Can Bite
- Can bark
- Can Code 😁

Properties and/or functions make up a class

Properties

There are just variables

```
legs = 4
```

Functions

There are just like the functions we looked at previously

```
def bark():
    print("woof")
```

Class

Made up of properties and functions

```
class Dog:
    legs = 4

    def bark(self):
        print("woof")
```

A Deep Dive into classes

```
class Dog:
    legs = 4

    def bark(self):
        print("woof")
```

#testing our class

```
bulldog = Dog()
bulldog.bark()
print(bulldog.legs)
```

- Classes are created with the `class` keyword.
- The word after the `class` keyword is the class name.
- Every Function in a class is indented
- Every function in a class must have the `self` argument
- Functions in classes are also called methods

The class constructor

```
class User:
    def __init__(self, name):
        self.name = name

    def like(self):
        print("liking")


max = User("max")
max.like()
print(max.name)
```

- Constructor runs as soon as you instantiate a class
- They are used in preparing a class for use

Class Inheritance

```
class Animal:
    def __init__(self, name):
        self.name = name

class Dog(Animal):
    def bark(self):
        print("woof")

class Duck(Animal):
    def quark(self):
        print("quaaarrk")
```

- **Inheritance** provides a way to share functionality between classes.
- Inheritance helps you save time
- With inheritance, you can easily make similar changes to many classes at a time

```
bulldog = Dog("killer")
bulldog.bark()
print(bulldog.name)
```

That’s how Magic is done!

Any questions?