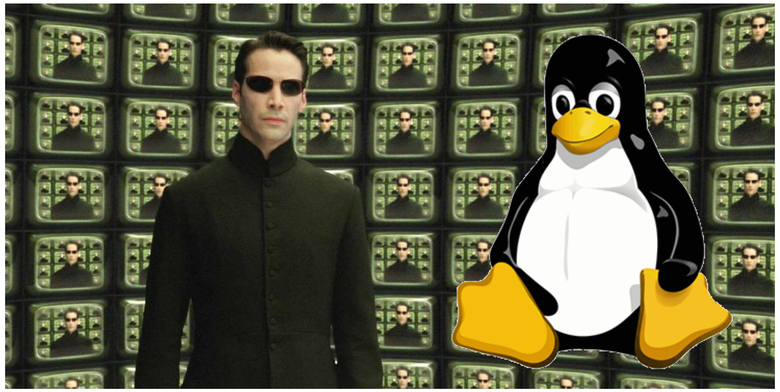
24.11.21 07:40

CSCI 120
Introduction to CompSci and Programming I
Lecture 8: Class I



Jetic Gū

Overview

- Focus: Python Programming
- Architecture: von Neumann
- Core Ideas:
    1. Python Class
    2. Lab

Class

Summary

P1
OOP

Algorithms

- The key to algorithms: data structure
- Complex data structures require carefully designed coding modules to realise
  - Python native: `list`
  - Python native: `dict`
- Data can by highly complex as well
  - Multiple columns, and multiple methods for you to work on these data

Review

Example: Employee info

- A company needs to manage its staff info
- Each staff has at least the following info
  - Name/ Sex/ Age
  - Supervisor
- Each of these entry will have different types
- How can we manage this data effectively?

Name: string

Gender: M/F/O

Age: int

Supervisor: person

Demo

P1
OOP

Class

- In python, you can declare custom classes with attributes and methods, to more effectively manage your data

- Declaring a class:

```
class Employee:
    statements
```

- After declaring the class, you can then create instances (or objects) of the class by calling the class name

```
john = Employee()
jetic = Employee()
```

Concept

P1
OOP

Class Methods

- You can declare methods for your classes, so each instance/object can perform stuff

- Example inside the `Employee` class

```
class Employee:
    def sayHello(self):
        print("This employee is saying hello to you!")

john = Employee()
john.sayHello()
```

- **Python class methods must have its first argument as** `self`

Concept

P1
OOP

Class Methods

- Usually, the first method you'd write is the constructor

- The constructor is called when you create the new object

```
class Employee:
  def __init__(self):
    self.name = "NoName"
    self.age = 0
    self.supervisor = None
john = Employee()
print(john.name)
```

- The constructor is called when you create the new object

Concept

Class Methods

- You can have other arguments in your constructor, as well as any other methods

```
class Employee:
  def __init__(self, name=None):    # name is an optional argument
    if name is not None:
      self.name = name
    else:
      self.name = "NoName"
    self.age = 0
    self.supervisor = None
john = Employee()
print(john.name)
jetic = Employee("jetic")
print(john.name)
```

Concept

Class Methods

- Arguments and self

```
class Employee:
    def __init__(…):
        …
    def sayHello(self):
        print(self.name + " says hello to you!")
jetic = Employee()
```

- Here, the following line:
  `jetic.sayHello()`
  is equivalent to:
  `Employee.sayHello(jetic)`

Concept

Extended Dict

A class example

Summary

P2
Extended Dict

Recall when we are counting words in Hamlet.txt

- When we are using dict, we must first determine if a key exists in the dict, then perform counting

```
counter = {}
if word in counter:
    counter[word] += 1
else:
    counter[word] = 1    # Otherwise it will throw KeyError
```

- Can we make it into a class?

Concept

P2
Extended Dict

Creating the counter class

```
class Counter:
    def __init__(self):
        self.co = {}

    def increase(self, key):
        if key in self.co:
            self.co += 1
        else:
            self.co = 1
    def get(self, key):
        return 0 if key not in self.co else self.co[key]
```

Use it in your Shakespeare programme!

Concept

Exercise

- Write the word counter class
- Use the word counter class to count words in Hamlet.txt
- Use the word counter class to redo Dict Practice 1 and Dict Practice 2

Exercise