AI . FREE Team 讀書會

IT Automation with Python

鄭中嘉（Eric）– 2022/03/26

Position: Where are we now? – Week 5

- Object Oriented Programming (Optional)
- Learning Objectives
  - Demonstrate object-oriented programming using classes and objects
  - Implement classes with custom attributes and methods
  - Write **docstrings** to document classes and methods
  - **Leverage inheritance** to reduce code duplication
  - Import and **use Python modules** to access powerful classes and methods

History 1/2: What has been shared?

Object-oriented Programming

- OOP Introduction
- What is Object-Oriented Programming?
- Classes and Objects in Python (**dir(“”)**, **help(“”)**)
- Defining New Classes (**Dot Notation**)

History 2/2: What has been shared?

Classes and Methods

- Instance Methods (**instance variable**)
- Constructors and Other Special Methods (**__init__**, **__str__**)

- **Topic 1: Documenting Functions, Classes, and Classes**
- **Topic 2: About Jupyter Notebook**

Topic 1:
Documenting Functions, Classes, and Classes

Docstring: A brief text that explains what something does

Docstring.py > Sensor_v2

```python
# ========================================================== #
# This code snippet shows the functionality of Docstring. #
# ========================================================== #

class Sensor_v1:

    def __init__(self):
        self.temp = 0.0
        self.pres = 0.0
        self.rh   = 0.0

    def measure(self, temp, pres, rh):···

    def display_temperature(self):···

    def display_pressure(self):···

    def display_relative_humidity(self):
        print(f"Relative Humidity: {self.rh}")
```

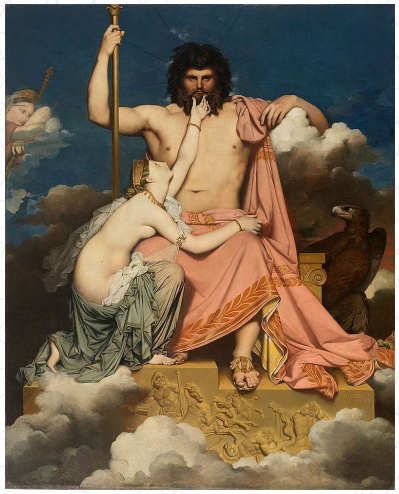
```python
class Sensor_v2:

    """
    This class contains methods to detect environmental variables.
    """

    def __init__(self):···

    def measure(self, temp, pres, rh):···

    def display_temperature(self):···

    def display_pressure(self):···

    def display_relative_humidity(self):
        """
        Display relative humidity.
        """
        print(f"Relative Humidity: {self.rh}")
```

Code: SSG/Docstring.py at main · Ratherman/SSG (github.com)

Topic 2:
About Jupyter Notebook

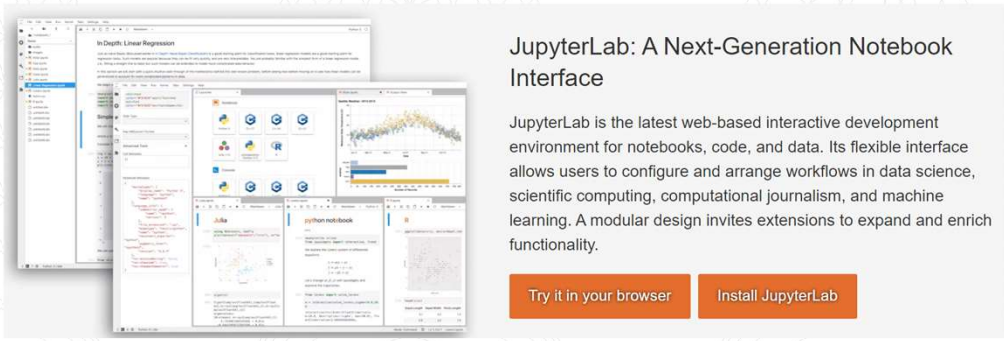
Open-Source, Where is “Jupyter Notebook” from?

古羅馬神話中的眾神之王



Project Jupyter | Home

Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.



Ref: Everything You Need To Know About Jupyter Notebooks | by Bharath K | Towards Data Science

Next Chapter

Code Reuse

- **Topic 3: Inheritance**
- **Topic 4: Composition**
- **Topic 5: Python Modules**

AI . FREE Team

Topic 3: Inheritance (is-a rule)


```python
class Model():

    """...

    def __init__(self, model_name, data_path, save_info, hparam):...

    def dataset(self):...

    def train(self):...

    def evaluate(self, plot_toggle=False):...

    def save(self):...
```

Inheritance to Reuse Code

```python
class ResNet(Model):

    def __init__(self, save_name, save_path, epoch, lr, bs):...

class LSTM(Model):

    def __init__(self, save_name, save_path, epoch, lr, bs):...
```

```python
class MLP(Model):

    def __init__(self, save_name, save_path, epoch, lr, bs):
        self.model_name = "MLP"
        self.data_path = "./dataset/"
        self.save_info = {
            "save_name": save_name,
            "save_path": save_path
        }
        self.hparam = {
            "epoch": epoch,
            "lr": lr,
            "bs": bs,
        }

        super(MLP, self).__init__(
            self.model_name,
            self.data_path,
            self.save_info,
            self.hparam
        )
```

Code: SSG/Inheritance.py at main · Ratherman/SSG (github.com)

Topic 4: Composition

- Inheritance: is-a rule
- Composition: has-a rule

```
class Vehicle:
    pass

class Car(Vehicle):
    pass
```

```
class Engine:
    pass

class Tire:
    pass

class Car(Engine, Tire):
    pass
```

Composition to Reuse Code

Q: What do you think about Composition?

Q: How do we relate Composition to our AI tech. ?

Ref: Python 的繼承 (Inheritance)與多形 (polymorphism) | 工程師小宇宙 (itcosmos.co)

AI . FREE Team

Topic 5:
Python Modules

Modules: Used to organize functions, classes, and other data together in a structured way.

Ex: import random ➔ random.randint(1, 10)

Ex: import datetime ➔ datetime.datetime.now()

AI . FREE Team

Wrap-Up 1/2

Class, Object, Attribute, Method

- Real-world concepts are represented by **classes**.
- Instances of classes are usually called **objects**.
- Objects have **attributes** which are used to store information about them.
- We can make object do work by calling their **methods**.

Wrap-Up 2/2

Dot Notation, Inheritance, Composition

- Access attributes and methods using **dot notation**
- Objects (/Classes) can be organized by **inheritance**
- Objects (/Classes) can be contained inside each other using **composition**

Code 1: Docstring.py

AI . FREE Team


```python
class Sensor_v1:

    def __init__(self):
        self.temp = 0.0
        self.pres = 0.0
        self.rh   = 0.0

    def measure(self, temp, pres, rh):
        self.temp = temp
        self.pres = pres
        self.rh   = rh

    def display_temperature(self):
        print(f"Temperature: {self.temp}")

    def display_pressure(self):
        print(f"Pressure: {self.pres}")

    def display_relative_humidity(self):
        print(f"Relative Humidity: {self.rh}")
```

Docstring.py > Sensor_v1

```python
class Sensor_v2:

    """
    This class contains methods to detect environmental variables.
    """

    def __init__(self):…

    def measure(self, temp, pres, rh):…

    def display_temperature(self):…

    def display_pressure(self):…

    def display_relative_humidity(self):
        """
        Display relative humidity.
        """
        print(f"Relative Humidity: {self.rh}")

JJs_old_sensor = Sensor_v1()
JJs_new_sensor = Sensor_v2()

print(help(JJs_old_sensor), end="\n\n")
print(help(JJs_new_sensor), end="\n\n")
```

Can you tell the difference?

AI . FREE Team

Code 2: Inheritance.py

Inheritance.py > ...

```python
# ========================================================== #
# This code snippet shows the functionality of Inheritance. #
# ========================================================== #

from time import time
import matplotlib.pyplot as plt

class Model():···

class MLP(Model):···

class ResNet(Model):···

class LSTM(Model):···

# =============== #
#  Neural Network #
# =============== #

JJs_NN = MLP(save_name="MLP_v1.h5", save_path="./trained_models/", epoch=30, lr=1e-3, bs=64)
print(JJs_NN.dataset())
print(JJs_NN.train())
print(JJs_NN.evaluate())
print(JJs_NN.save())
```