AI. FREE Team 讀書會

IT Automation with Python



鄭中嘉 (Eric) - 2022/03/26





Position: Where are we now? – Week 5

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

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?



History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

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?



History

Position

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

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

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

Docstring: A brief text that explains what something does

```
Docstring.py > Sensor v2
     # This code snippet shows the functionality of Docstring. #
     # ----- #
     class Sensor v1:
        def init (self):
            self.temp = 0.0
            self.pres = 0.0
10
            self.rh = 0.0
11
12 >
        def measure(self, temp, pres, rh): ...
16
        def display temperature(self): ...
17 >
19
20 >
        def display_pressure(self): ...
22
         def display relative humidity(self):
23
24
            print(f"Relative Humidity: {self.rh}")
```

```
26 v class Sensor v2:
27
28 V
29
         This class contains methods to detect environmental variables.
30
31
         def __init__(self): ...
32 >
39
40 >
         def measure(self, temp, pres, rh): ...
48
         def display temperature(self): ...
49 >
54
55 >
         def display_pressure(self): ...
60
61 V
         def display_relative_humidity(self):
62 V
63
             Display relative humidity.
64
65
             print(f"Relative Humidity: {self.rh}")
```

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



Topic 2: About Jupyter Notebook

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

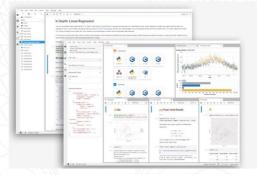
Project Jupyter | Home

Open-Source, Where is "Jupyter Notebook" from?

古羅馬神話中的眾神之王



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



JupyterLab: A Next-Generation Notebook Interface

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

Try it in your browser

nstall Junyterl ab

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



Next Chapter

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

Code Reuse

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



Topic 3: Inheritance (is-a rule)

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

```
class Model():
 5
 6 >
17
18 >
          def init (self, model name, data path, save info, hparam): ...
40
41 >
          def dataset(self): ...
50
51 >
          def train(self): ...
78
          def evaluate(self, plot_toggle=False): ...
79 >
99
100 >
          def save(self): ···
```

Inheritance to Reuse Code

```
108 ∨ class MLP(Model):
109
110 V
          def init (self, save name, save path, epoch, lr, bs):
111
              self.model name = "MLP"
112
              self.data path = "./dataset/"
113 V
              self.save info = {
114
                   "save_name": save_name,
115
                  "save path": save path
116
117 V
              self.hparam = {
118
                   "epoch": epoch,
119
                  "lr": lr,
120
                  "bs": bs,
121
122
123 V
              super(MLP, self).__init__(
124
                  self.model name,
125
                  self.data path,
126
                  self.save_info,
127
                  self.hparam
128
```

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



Topic 4: Composition

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

• 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):
```

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)



Topic 5: Python Modules

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

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

Ex: import random \rightarrow random.randint(1, 10)

Ex: import datetime

datetime.

datetime.now()



Wrap-Up 1/2

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 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

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 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

```
Position
History
Topic 1
Topic 2
Topic 3
Topic 4
Topic 5
Wrap-up
Code 1
```

Code 2

```
♣ Docstring.py > ♣ Sensor_v1
     class Sensor_v1:
                                                                     class Sensor_v2:
 6
                                                                27
         def __init__(self):
                                                                28
 8
             self.temp = 0.0
                                                                29
                                                                         This class contains methods to detect environmental variables.
             self.pres = 0.0
 9
                                                                30
10
             self.rh = 0.0
                                                                31
11
                                                                32 >
                                                                         def init (self): ···
12 V
         def measure(self, temp, pres, rh):
                                                                39
13
             self.temp = temp
                                                                40 >
                                                                         def measure(self, temp, pres, rh): ...
14
             self.pres = pres
                                                                48
15
             self.rh = rh
                                                                49 >
                                                                         def display temperature(self): ...
16
                                                                54
17 V
         def display temperature(self):
                                                                55 >
                                                                         def display pressure(self): ...
             print(f"Temperature: {self.temp}")
18
                                                                60
19
                                                                         def display relative humidity(self):
                                                                61
20 V
          def display pressure(self):
                                                                62
             print(f"Pressure: {self.pres}")
21
                                                                63
                                                                             Display relative humidity.
22
                                                                64
23 V
         def display relative humidity(self):
                                                                65
                                                                             print(f"Relative Humidity: {self.rh}")
24
             print(f"Relative Humidity: {self.rh}")
                                                                     JJs old sensor = Sensor v1()
                                                                     JJs new sensor = Sensor v2()
Can you tell the difference?
                                                                     print(help(JJs old sensor), end="\n\n")
                                                                     print(help(JJs new sensor), end="\n\n")
```



Code 2: Inheritance.py

Position

History

Topic 1

Topic 2

Topic 3

Topic 4

Topic 5

Wrap-up

Code 1

Code 2

```
♣ Inheritance.py > ...
     # This code snippet shows the functionality of Inheritance. #
     # ----- #
     from time import time
     import matplotlib.pyplot as plt
   > class Model(): ···
111
112 > class MLP(Model): ...
133
134 > class ResNet(Model): ...
155
156 > class LSTM(Model): ...
177
     # ======= #
178
179
     # Neural Network #
180
     # ======= #
181
182
     JJs_NN = MLP(save_name="MLP_v1.h5", save_path="./trained_models/", epoch=30, lr=1e-3, bs=64)
     print(JJs_NN.dataset())
183
184
     print(JJs NN.train())
     print(JJs_NN.evaluate())
     print(JJs_NN.save())
```