安徽理工大学
ANHUI UNIVERSITY OF SCIENCE & TECHNOLOGY

• 人工智能专业 学科基础教育必修模块

Python与机器学习

Python and Machine Learning

Chapter 5: Python Object-Oriented Programming

• Lecturer ： 孟亦凡

• E-mail：myf@aust.edu.cn

安徽理工大学
ANHUI UNIVERSITY OF SCIENCE & TECHNOLOGY

Chapter 5： Python Object-Oriented Programming

Target

- 理解面向对象编程的核心概念
- **Understand the core concepts of Object-Oriented Programming**
- 掌握Python中类的定义与使用
- **Master the definition and usage of classes in Python**
- 学会使用Python标准库和第三方库
- **Learn to use Python standard libraries and third-party libraries**

安徽理工大學
ANHUI UNIVERSITY OF SCIENCE & TECHNOLOGY

Chapter 5： Python Object-Oriented Programming

CONTENTS

5.1 What is Object-Oriented Programming
什么是面向对象编程

5.1 What is Object-Oriented Programming

➢ The real world is composed of objects（现实世界是由对象组成的）

Each object has:

- **Attributes - Characteristics that describe the object（描述对象的特征）**
- **Behaviors - Operations that the object can perform（对象能够执行的操作）**

Real-world examples:

- **Car: Attributes(color, brand, speed), Behaviors(start, accelerate, brake)**
- **Student: Attributes(name, student_id, grades), Behaviors(study, take_exam, submit_homework)**

➢ Procedure-Oriented Programming（面向过程编程）

- **Focus on "how to do" - steps and functions（关注"怎么做" - 步骤和函数）**
- **Data and operations are separated（数据与操作分离）**

```
# 面向过程示例
student_name = "张三"
student_score = 85
def calculate_grade(score):
    if score >= 90: return "A"
    elif score >= 80: return "B"
    else: return "C"
grade = calculate_grade(student_score)
```

➢ Object-Oriented Programming：Three features

- **Encapsulation（封装）**
  - **Wrapping data and behaviors in a class（将数据和行为包装在类中）**
  - **Hiding internal implementation details（隐藏内部实现细节）**
  - **Interacting with objects through public interfaces（通过公共接口与对象交互）**
- **Inheritance（继承）**
  - **Subclasses inherit attributes and methods from parent classes（子类继承父类的属性和方法）**
  - **Achieving code reuse and hierarchical relationships（实现代码重用和层次关系）**

➢ Object-Oriented Programming：Three features

- **Polymorphism（多态）**
  - **Different objects respond differently to the same message（不同对象对同一消息做出不同响应）**
  - **Improving code flexibility and extensibility（提高代码的灵活性和可扩展性）**

➢ Object-oriented applications in machine learning

- **Object-Oriented Design in Scikit-learn**

```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
# 创建模型对象
model1 = LinearRegression()    # 线性回归对象
model2 = KNeighborsClassifier() # K近邻分类器对象
# 统一的方法调用
model1.fit(X_train, y_train)   # 训练
predictions = model1.predict(X_test)  # 预测
```

➢ Object-oriented applications in machine learning

- **Object-Oriented Design in Scikit-learn**
- **Advantages:**
  - **Consistent API: All models have fit() and predict() methods**
  - **Easy to extend: Can create custom model classes（可以创建自定义模型类）**
  - **Modularity: Separates data preprocessing（预处理）, model training, evaluation（评估）**

5.2 Python Classes
Python的类

5.2 Python Classes

➢ Class Definition & Create Syntax

```
class ClassName:
    """类的文档字符串"""
    # 类属性
    class_attribute = value

    def __init__(self, parameters):
        # 实例属性
        self.attribute = value

    def method(self, parameters):
        # 方法实现
        return result
```

Definition

```
# 实例化对象
object_name = ClassName(arguments)
# 访问属性和方法
print(object_name.attribute)
object_name.method(arguments)
```

Create

5.2 Python Classes

➢ __init__方法

- Constructor Method（构造方法）
  - Automatically called when creating an object（在创建对象时自动调用）
  - Used to initialize object attributes（用于初始化对象的属性）
- self参数
  - Reference to the object instance itself（指向对象实例本身的引用）
  - Must be the first parameter of methods（必须是方法的第一个参数）

5.2 Python Classes

➢ __init__方法

```
class Student:
    def __init__(self, name, student_id, major):
        self.name = name
        self.student_id = student_id
        self.major = major
        self.grades = []  # 初始化为空列表

    def add_grade(self, course, score):
        self.grades.append({"course": course, "score": score})
# 创建学生对象
student1 = Student("李四", "2024001", "计算机科学")
student1.add_grade("Python编程", 95)
```

5.2 Python Classes

➢ Class Attributes（类属性）与Instance Attributes（实例属性）

- Class Attributes(类属性)
  - Attributes shared by all instances（所有实例共享的属性）
  - Defined inside the class but outside methods（在类内部但在方法外部定义）
- Instance Attributes（实例属性)
  - Attributes unique to each instance(每个实例独有的属性)
  - Defined in the __init__ method（在__init__方法中定义）

5.2 Python Classes

➢ Class Attributes（类属性）与实例属性

```
class Car:
    # 类属性
    wheels = 4
    country = "中国"

    def __init__(self, brand, color):
        # 实例属性
        self.brand = brand
        self.color = color
        self.speed = 0
# 使用
car1 = Car("比亚迪", "红色")
car2 = Car("特斯拉", "蓝色")
print(Car.wheels)     # 4 - 通过类访问类属性
print(car1.wheels)    # 4 - 通过实例访问类属性
print(car1.brand)     # "比亚迪" - 实例属性
```

5.2 Python Classes

➢ Classes Inheritance Syntax （类继承语法

```
class ParentClass:
    # 父类定义
    pass
class ChildClass(ParentClass):
    # 子类定义
    pass
```

super()是一个特殊函数，让你能够调用父类的方法。

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

    def speak(self):
        return "Some sound"
class Dog(Animal):
    def __init__(self, name, breed):
        super().__init__(name)  # 调用父类构造方法
        self.breed = breed

    def speak(self):  # 方法重写
        return "Woof!"
class Cat(Animal):
    def speak(self):  # 方法重写
        return "Meow!"
# 使用
dog = Dog("Buddy", "Golden Retriever")
cat = Cat("Whiskers")
print(dog.speak())  # "Woof!"
print(cat.speak())  # "Meow!"
```

5.3 Python Libraries
Python的库

5.3 Python Libraries

➢ Python库的分类

- 标准库Standard Library
- Comes with Python installation（随Python安装包自带）

- No additional installation required（无需额外安装）

- Such as： math, datetime, os, json

5.3 Python Libraries

➢ Python库的分类

- **第三方库Third-party Libraries**
- **需要手动安装**
- **Installed via pip tool**
- **示例：numpy, pandas, matplotlib**

```
# 使用pip安装
pip install numpy pandas matplotlib
# 安装特定版本
pip install scikit-learn==1.2.0
# 从requirements文件安装
pip install -r requirements.txt
```

```
# 直接导入
import math
print(math.sqrt(16))

# 导入并重命名
import numpy as np
import pandas as pd

# 导入特定函数/类
from sklearn.linear_model import
LinearRegression
from matplotlib import pyplot as plt

# 导入所有（不推荐）
from math import *
```

5.3 Python Libraries

➢ Python库的分类

- 自定义库Custom Libraries
- Self-written modules and packages（自己编写的模块和包）
- Reusable code（可重复使用的代码）

5.3 Python Libraries

➢ What is PIP?

- **Python's official package manager**
- **Used to install and manage third-party Python libraries（用于安装和管理第三方Python库）**
- **Automatically installed with Python 3.4+（随Python 3.4+版本自动安装）**

5.3 Python Libraries

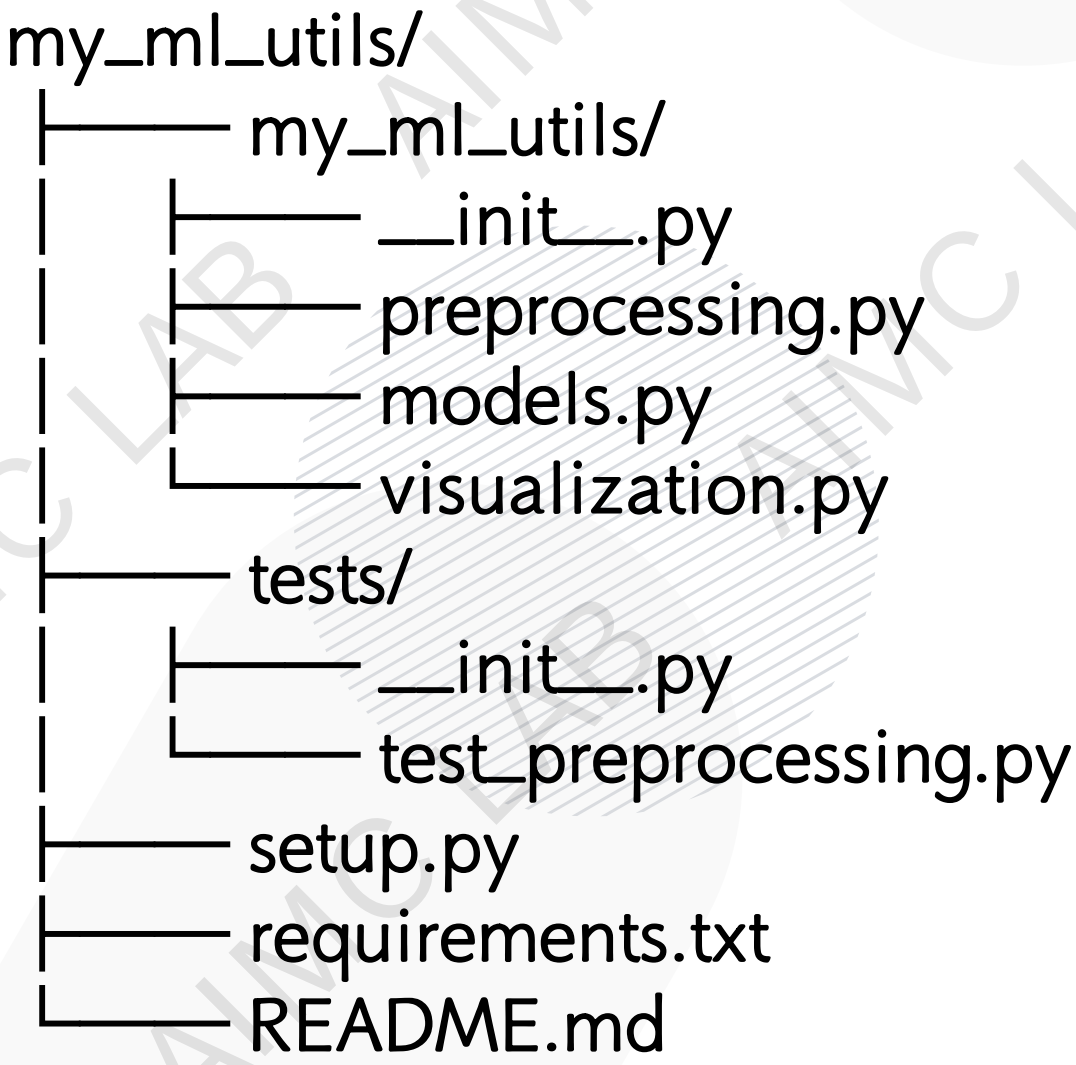
➢ Common PIP Commands

```
# 安装包
pip install package_name
# 安装特定版本
pip install package_name==1.2.3
# 升级包
pip install --upgrade package_name
# 卸载包
pip uninstall package_name
# 查看已安装的包
pip list
# 导出环境依赖
pip freeze > requirements.txt
# 从文件安装依赖
pip install -r requirements.txt
```

5.3 Python Libraries

➢ Create My Libraries

- **Basic Library Structure**



5.3 Python Libraries

➢ Create My Libraries

- **Core File Descriptions**
  - **__init__.py：Marks directory as Python package**
  - **setup.py：Installation configuration for the library**
  - **README.md：Usage documentation for the library**

5.3 Python Libraries

➢ Create My Libraries

```
from setuptools import setup, find_packages
setup(
    name="my_ml_utils",
    version="0.1.0",
    description="A collection of machine learning utilities",
    author="Your Name",
    author_email="your.email@example.com",
    packages=find_packages(),
    install_requires=[
        "numpy>=1.19.0",
        "pandas>=1.1.0",
        "matplotlib>=3.3.0",
        "scikit-learn>=0.24.0"
    ],
    python_requires=">=3.7",
    classifiers=[
        "Development Status :: 3 - Alpha",
        "Intended Audience :: Developers",
        "License :: OSI Approved :: MIT License",
        "Programming Language :: Python :: 3",
        "Programming Language :: Python :: 3.7",
        "Programming Language :: Python :: 3.8",
        "Programming Language :: Python :: 3.9",
    ],
)
```

5.3 Python Libraries

➢ Create My Libraries

- **Basic Structure of setup.py**
- **name: 包名称，PyPI唯一**
- **version: 版本号（主次修）**
- **description: 简短描述**
- **author: 作者的姓名。**
- **author_email: 联系邮箱。**

```
from setuptools import setup, find_packages
setup(
    name="my_ml_utils",
    version="0.1.0",
    description="A collection of machine learning utilities",
    author="Your Name",
    author_email="your.email@example.com",
    packages=find_packages(),
    install_requires=[
        "numpy>=1.19.0",
        "pandas>=1.1.0",
        "matplotlib>=3.3.0",
        "scikit-learn>=0.24.0"
    ],
    python_requires=">=3.7",
    classifiers=[
        "Development Status :: 3 - Alpha",
        "Intended Audience :: Developers",
```

5.3 Python Libraries

➢ Create My Libraries

- **packages: 指定包含在包中的Python包（即目录）列表。**
- **install_requires: 列出所依赖的第三方库。当用户安装你的包时，这些依赖会自动安装。**

```
from setuptools import setup, find_packages
setup(
    name="my_ml_utils",
    version="0.1.0",
    description="A collection of machine learning utilities",
    author="Your Name",
    author_email="your.email@example.com",
    packages=find_packages(),
    install_requires=[
        "numpy>=1.19.0",
        "pandas>=1.1.0",
        "matplotlib>=3.3.0",
        "scikit-learn>=0.24.0"
    ],
    python_requires=">=3.7",
    classifiers=[
        "Development Status :: 3 - Alpha",
        "Intended Audience :: Developers",
        "License :: OSI Approved :: MIT License",
        "Programming Language :: Python :: 3",
        "Programming Language :: Python :: 3.7",
        "Programming Language :: Python :: 3.8",
        "Programming Language :: Python :: 3.9",
    ],
)
```

5.3 Python Libraries

➢ Create My Libraries

- **python_requires: 指定你的包支持的Python版本范围。**
- **classifiers: 额外元数据，比如开发状态、目标受众、许可证、支持的Python版本等。这些信息会在PyPI上显示，帮助用户找到你的包。**

```
from setuptools import setup, find_packages
setup(
    name="my_ml_utils",
    version="0.1.0",
    description="A collection of machine learning utilities",
    author="Your Name",
    author_email="your.email@example.com",
    packages=find_packages(),
    install_requires=[
        "numpy>=1.19.0",
        "pandas>=1.1.0",
        "matplotlib>=3.3.0",
        "scikit-learn>=0.24.0"
    ],
    python_requires=">=3.7",
    classifiers=[
        "Development Status :: 3 - Alpha",
        "Intended Audience :: Developers",
        "License :: OSI Approved :: MIT License",
        "Programming Language :: Python :: 3",
        "Programming Language :: Python :: 3.7",
        "Programming Language :: Python :: 3.8",
        "Programming Language :: Python :: 3.9",
    ],
)
```

5.3 Python Libraries

➢ Create My Libraries

- **preprocessing.py Example**

```
"""
数据预处理工具模块
Data preprocessing utilities module
"""
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
class DataPreprocessor:
    """数据预处理器类"""

    def __init__(self):
        self.scaler = StandardScaler()
        self.is_fitted = False

    def fit_transform(self, data):
        """拟合并转换数据"""
        transformed_data = self.scaler.fit_transform(data)
        self.is_fitted = True
        return transformed_data
# 工具函数
def normalize_data(data):
    """归一化数据到[0,1]范围"""
    return (data - data.min()) / (data.max() - data.min())
```

5.3 Python Libraries

➢ Create My Libraries

- **Role of init.py**
  - **定义包的公共接口**
  - **Defines the public interface of the package**
  - **控制从包中导入的内容**
  - **Controls what gets imported from the package**

```
"""
My ML Utilities - 机器学习工具库
My ML Utilities - Machine Learning Utilities Library
"""
__version__ = "0.1.0"
__author__ = "Your Name"
# 导入关键功能到包级别
from .preprocessing import DataPreprocessor, normaliz
e_data
from .models import SimpleClassifier, SimpleRegressor
from .visualization import plot_correlation_matrix
# 定义__all__来控制from package import *的行为
__all__ = [
    'DataPreprocessor',
    'normalize_data',
    'SimpleClassifier',
    'SimpleRegressor',
    'plot_correlation_matrix'
]
```

5.3 Python Libraries

➢ Create My Libraries

- **Role of init.py**
  - **定义包的公共接口**
  - **Defines the public interface of the package**
  - **控制从包中导入的内容**
  - **Controls what gets imported from the package**

```
"""
My ML Utilities - 机器学习工具库
My ML Utilities - Machine Learning Utilities Library
"""
__version__ = "0.1.0"
__author__ = "Your Name"
# 导入关键功能到包级别
from .preprocessing import DataPreprocessor, normaliz
e_data
from .models import SimpleClassifier, SimpleRegressor
from .visualization import plot_correlation_matrix
# 定义__all__来控制from package import *的行为
__all__ = [
    'DataPreprocessor',
    'normalize_data',
    'SimpleClassifier',
    'SimpleRegressor',
    'plot_correlation_matrix'
]
```

5.3 Python Libraries

➢ Create My Libraries

- 打包和分发库
  - **Building the Package**
  - **Installing Local Development Version**
  - **Distributing to PyPI**

5.4 Packaging as Executable Files
打包为可执行文件

5.4 Packaging as Executable Files

➢ Why package Python programs into executable files?

- **Application：**
  - **Distributing to users without Python environment（分发給没有Python环境的用户）**
  - **Deploying to production servers（部署到生产服务器）**
  - **Creating desktop applications（创建桌面应用程序）**
  - **Protecting source code（保护源代码）**

5.4 Packaging as Executable Files

➢ Why package Python programs into executable files?

- **Advantages:**
  - **User-friendly: No need to install Python and dependencies**
  - **Easy deployment: Single file or folder can run**
  - **Environment isolation: Avoid environment dependency issues**

5.4 Packaging as Executable Files

➢ Comparison of common packaging tools

- **PyInstaller （简单上手）**
- **Cross-platform support (Windows, macOS, Linux)**
- **Packages as single executable**
- **Easy to use, active community**
- **Principle: Bundles Python interpreter and dependencies**

5.4 Packaging as Executable Files

➢ Comparison of common packaging tools

- **Nuitka （性能更好）**
- **Compiles Python to C++, then to machine code**
- **Higher performance, faster execution**
- **Better source code protection**
- **Cross-platform support**

5.4 Packaging as Executable Files

➢ Comparison of common packaging tools

特性	Nuitka	PyInstaller	cx_Freeze	py2exe
跨平台				(仅Windows)
单文件模式	(实验性)			
性能	(最佳)	(中等)	(中等)	(中等)
代码保护	(编译)	(打包)	(打包)	(打包)
易用性	(中等)	(简单)	(中等)	(中等)
编译时间	(较长)	(中等)	(中等)	(中等)
社区活跃	(增长中)	(活跃)	(稳定)	(较少)
文档质量	(良好)	(优秀)	(良好)	(良好)

5.4 Packaging as Executable Files

➢ PyInstaller Basic Usage

- **Installing PyInstaller**

```
pip install pyinstaller
```

- **Basic Packaging Commands**

```
pyinstaller your_script.py # 基本打包 - 生成文件夹
pyinstaller --onefile your_script.py # 打包为单个可执行文件
pyinstaller --onefile --windowed your_script.py # 打包时不显示控制台窗口 (Windows GUI程序)
pyinstaller --onefile --icon=app.ico your_script.py # 指定程序图标
```

5.4 Packaging as Executable Files

➢ PyInstaller Basic Usage

- **Packaging Results**
- project/
- ├── dist/ # 最终的可执行文件
- │ └── your_script.exe
- ├── build/ # 临时构建文件
- └── your_script.spec # 配置文件