# 金融计算机语言讲义

高强 (mutecamel@gmail.com) 首都经济贸易大学金融学院



## 第 5 课 面向对象编程入门

### 学习任务

本周是第 7 周。请一边阅读教材一边完成**课程测验**任务，同时一边阅读一边完成以下**非上机题**任务。课程测验得分将构成平时成绩 (占期末总评 40%)；所有非上机题将构成期末闭卷考试 (占期末总评 60%) 的题库。所以请务必认真对待学习任务，不能按时完成任务者后果自负。

#### 1 阅读

阅读教材 “Introduction to Python / OOP I: Introduction to Object Oriented Programming” 一节，学习有关面向对象编程 (OOP) 的基础知识。

#### 2 简答题

Python 既支持函数式编程，又支持面向对象编程，请简要介绍两者的关键区别。

#### 3 简答题

Python 语言将运行时 (runtime) 的内存 (memory) 数据以对象 (object) 为单位进行管理。请简要介绍对象 (object) 所包含的四个基本要素。

#### 4 简答题

请问执行以下 Python 代码会不会报错?

```
>>> def add(x, y):
...     return x + y
```

接着执行以下代码会不会报错?

```
>>> add(300, 400)
```

接着执行以下代码会不会报错?

```
>>> add('Python', 'ista')
```

接着执行以下代码会不会报错?

```
>>> add('300', 400)
```

若以上步骤中会有报错，出错的原因分别是什么？请重新定义一个 “正确” 的 `add` 函数，使以上代码不出现报错。

### 5 简答题

```
>>> x = [3, 1, 2]
>>> y = [3, 1, 2]
>>> z = x
```

以上代码运行后，`x`、`y`、`z` 哪些指向的是同一个对象？为什么？

### 6 简答题

什么是对象 (object) 的属性 (attribute)？如果有一个名称 `x` 指向着对象 (object) 整数 (int) `42`，如何通过 `x` 访问到 `42` 这个对象的 `imag` 属性？

### 7 简答题

Python 对象 (object) 的方法 (method) 与 Python 函数 (function) 之间有什么联系和区别？

### 8 简答题

运行代码

```
>>> x = ['foo', 'bar']
```

之后，`x.append` 所指向的对象 (object) 是否可调用 (callable)？`x.__doc__` 是否可调用 (callable)？`x.__class__` 是否可调用 (callable)？通过什么 Python 代码可以判断他们是否可调用 (callable)？

### 9 简答题

运行以下代码之后，`q` 会等于什么值？

```
>>> q = [7, 9, 8]
>>> q.remove(9)
```

运行以下代码之后，`s` 会等于什么值？

```
>>> s = 'This is a string'
>>> s.replace('This', 'That')
```

这能说明什么问题？

### 10 简答题

运行以下代码

```
>>> q = [7, 9, 8]
```

之后，可以说名称 `q` 现在指向着一个列表 (list) 对象 (object)。那么，`q` 所捆绑 (bind) 的方法 (method) `q.append`，其所指向的还是不是对象 (object)？为什么？通过什么可以看出来？

## 11 简答题

Python 对象 (object) 普遍捆绑 (bind) 有大量 “双下划线方法” (double-underscore methods，简称 dunder methods)，用于各种 Python 语法的内部实现 (implementation)。例如，表达式 `x + y` 在 Python 内部其实是调用了 `x.__add__(y)` 实现的。请列举任意 *三个* 你所了解的某种类型 (type) 的双下划线方法 (dunder methods)，及其等价的 Python 语法。

## 12 探索

Python 的内置函数 (built-in functions) 实际上是一个名为 `__builtins__` 的模块 (module) 的属性 (attributes)。执行以下代码

```
>>> dir(__builtins__)
```

将返回 `__builtins__` 模块 (module) 下所有属性 (attributes) 的名称 (name) 列表 (list)，*全部* 内置函数 (built-in functions) 的名称 (name) (以小写字母开头) 就包含在其中，除此之外还包括许多其他内置对象，大多为代表各种错误的内置错误类型 (built-in error types) (以大写字母开头)。

将任意一个对象 (object) 传递给内置函数 (built-in function) `help`，都能够在终端 (terminal) 输出其文档 (docstring)。例如执行

```
>>> help(id)
```

将显示内置函数 (built-in function) `id` 的文档

```
id(obj, /)
    Return the identity of an object.

    This is guaranteed to be unique among simultaneously existing objects.
    (CPython uses the object's memory address.)
```

`type`、`id`、`callable`、`dir`、`help` 这几个可接受 *任意* 对象 (object) 为参数 (argument) 的内置函数 (built-in function) 被称为自省函数 (introspection functions)，利用他们可以在运行时 (runtime) 获得 *任意* 对象 (object) 的基本信息。

请使用 `dir` 和 `help` 自行探索 Python 所有的内置函数 (built-in functions) (小写字母开头)。在这些内置函数 (built-in function) 中，有一些确实是函数 (function)，例如 `sorted`，通过运行 `help(sorted)` 将会看到

```
sorted(iterable, /, *, key=None, reverse=False)
    Return a new list containing all items from the iterable in ascending
order.

    A custom key function can be supplied to customize the sort order, and the
    reverse flag can be set to request the result in descending order.
```

而另一些其实是类 (class)，例如 `str`，通过运行 `help(str)` 将会看到

```
class str(object)
 |  str(object='') -> str
 |  str(bytes_or_buffer[, encoding[, errors]]) -> str
 |
 |  Create a new string object from the given object. If encoding or
 |  errors is specified, then the object must expose a data buffer
 |  that will be decoded using the given encoding and error handler.
 |  Otherwise, returns the result of object.__str__() (if defined)
 |  or repr(object).
...
```

类 (class) 的文档都是以 `class` 开头的。按照文档说明的方法调用 `str`，例如执行 `str(42)`，将像调用函数 (function) 那样返回一个新创建的类型 (type) 为 `str` 的对象 (object)。

请从内置函数 (built-in functions) 中把所有这种内置类 (built-in class) 找出来。

### 13 简答题

请列举至少 10 个 Python 内置类型 (built-in class) 的名称 (name)。

### 14 探索

请使用 `dir` 和 `help` 自行探索 Python 所有内置类型 (built-in class) 所附带的方法，尤其是浏览并了解 `str`、`list`、`dict`、`set` 等类型的所有方法。

### 15 简答题

在 `str` 类型 (class) 的各种方法 (method) 中，请列举你认为最常用的五个，并举例说明其用途、用法。

### 16 简答题

在 `list` 类型 (class) 的各种方法 (method) 中，请列举你认为最常用的五个，并举例说明其用途、用法。

### 17 简答题

在 `dict` 类型 (class) 的各种方法 (method) 中，请列举你认为最常用的五个，并举例说明其用途、用法。

### 18 简答题

在 `set` 类型 (class) 的各种方法 (method) 中，请列举你认为最常用的五个，并举例说明其用途、用法。

### 19 探索

在运行时 (runtime) 通过 `import` 语句 (statement) 所导入的模块 (module) 也是对象 (object)。与普通对象 (object) 的属性 (attributes) 多为绑定的 (binded) 方法 (method) 不同，模块 (module) 对象的属性 (attributes) 多为子模块 (sub-module)、函数 (function) 和类 (class)。模块 (module) 是完成一类专业 (specialized) 工作所需的各种预先定义 (编程) 好的函数 (function) 和类 (class) (及其方法 (method)) 的一种组织整理方式。例如 Python 标准库 (standard library) 中所自带的 `datetime` 模块 (module)，组织整理了各种处理日期时间有关的函数 (function) 和类 (class)。执行 `import datetime` 之后，执行 `help(datetime)` 就可以在终端 (terminal) 里查看到 `datetime` 模块的文档。执行 `dir(datetime)` 就可以看到 `datetime` 模块里的内容，其中重要的有 `datetime.date`、`datetime.time` 、`datetime.datetime` 三个类 (class)，分别用于处理日期、时间、日期-时间等类型的数据及其有关计算。

在包罗万象的 Python 标准库 (standard library) 里，特别常用的几个模块有：1. `datetime`，2. `random`，3. `os`，4. `re`，5. `pathlib`，6. `json`，7. `sqlite3` 等。请自行阅读他们的文档有个大概的了解。

### 20 简答题

请分别用一句话介绍以下几个 Python 标准库 (standard library) 中的常用模块 (module) 的主要用途：1. `datetime`，2. `random`，3. `os`，4. `re`，5. `pathlib`，6. `json`，7. `sqlite3`。

### 21 阅读

阅读教材 “Introduction to Python / OOP II: Building Classes” 一节，学习有关自定义类 (class) 的基础知识。

### 22 简答题

请简要介绍类定义 (class definition) 及其实例 (instance) 的概念，以及它们之间的关系。

### 23 探索

严格来说，Python 运行时 (runtime) 的内存 (memory) 里一切都是对象 (object)。实例 (instance)、函数定义 (function definition)、类定义 (class definition)、模块定义 (module definition)、甚至函数 (function) 或方法 (method) 被调用 (called) 而运行 (run) 时所产生的层层堆栈 (stack frame) 等，都是对象 (object)。但通常我们说对象 (object) 也简单地指的是实例 (instance)，因为实例 (instance) 是最多地被我们使用的对象 (object)。

所有的对象 (object) 都有一个在内存中创建它时所依循的 “模板” 或 “蓝图”，被称为 “类型” (type)。内置函数 (built-in function) `type` 可以被用来获得任意一个对象 (object) 的类型 (type)。例如执行代码

```
>>> type(42) is int
```

返回的会是 `True`，其中 `is` 是用来判断左右两边所指向的是否为同一对象 (object) 的运算符 (operator)。可以看到，整数 (int) `42` 的类型是类 (class) `int`。那么，“模板” 或 “蓝图” 本身，还有没有模板的 “模板”、蓝图的 “蓝图” 呢？执行代码

```
>>> type(int) is type
```

返回的会是 `True`，说明内置类型 (built-in class) `int` 的类型 (type) 是内置类型 (built-in class) `type`。那么，`type` 本身，还有没有 `type` 的 `type` 呢？执行代码

```
>>> type(type) is type
```

返回的会是 `True`。这说明，终极 “模板” 就是 `type`。从这个意义上来讲，因为所有的对象 (object) 都可以找到 “模板” 或 “蓝图”，所以任何一个对象 (object) 确实都是实例 (instance)。

我们说 “实例” (instance)，主要是相对于它的 “模板” 或 “蓝图” 而言的。执行代码

```
>>> isinstance(42, int)
>>> isinstance(int, type)
>>> isinstance(type, type)
```

返回的都会是 `True`，其中 `isinstance` 是内置函数 (built-in function)，用来判断第一个参数 (argument) 是否是第二个参数 (argument) 的实例 (instance)，或者说，是判断第二个参数 (argument) 是否是第一个参数 (argument) 的类型 (type)。所以，以下代码永远都会返回 `True`，

```
>>> isinstance(x, type(x))
```

无论名称 (name) `x` 指向什么对象 (object)。

类 (class) 和类 (class) 之间还可以有一种继承 (inheritance) 关系。如果类 `SecureConnection` 继承了类 `Connection`，例如

```
>>> class Connection:
...     secret = 42
...     def tell(self):
...         return f'My secret is {self.secret}.'
...
>>> class SecureConnection(Connection):
...     def tell(self):
...         return f"I won't tell you that my secret is {self.secret}."
...     def clear(self):
...         self.secret = ''
...
>>> conn = SecureConnection()
>>> conn.tell()
"I won't tell you that my secret is 42."
>>> conn.clear()
>>> conn.tell()
"I won't tell you that my secret is ."
```

那么 `SecureConnection` 被称为是 `Connection` 的子类 (subclass)，将继承 `Connection` 的所有属性 (`secret` 和 `tell`)，并在其基础上可以重新定义属性 (`tell`)，也可以定义新属性 (`clear`)。

内置函数 (built-in function) `issubclass` 可以用来判断一个类 (class) 是不是另一个类 (class) 的子类 (subclass)。例如，

```
>>> issubclass(SecureConnection, Connection)
>>> issubclass(bool, int)
>>> issubclass(ValueError, Exception)
```

返回的都会是 `True`。

Python 下所有的类 (class) 都是内置类型 (built-in class) `object` 的子类。例如，

```
>>> issubclass(SecureConnection, object)
>>> issubclass(str, object)
>>> issubclass(dict, object)
>>> issubclass(KeyError, object)
```

返回的都会是 `True`。

虽然 `object` 和 `type` 都有 “万物之母” 的意味，但要注意 `object` 是所有类 (class) 的基类 (base class)，而 `type` 是所有类 (class) 的 “类”，`object` 是 `type` 的一个实例 (instance)，而不是子类 (subclass)。例如，

```
>>> isinstance(object, type)  # 返回 True
>>> issubclass(object, type)  # 返回 False
```

综上，从 “实例” (instance) 的角度来理解，Python 对象 (object) 可以分为三个层次。最顶层是 `type`，`type` 是 `type` 自己的实例 (instance)。第二层是所有的类 (class)，所有的类 (class) 都是 `type` 的实例 (instance)。在第二层，类 (class) 和类 (class) 之间可以有各种复杂的继承 (inheritance) 关系，像族谱那样。`object` 处于第二层，是所有其他类 (class) 的基类 (base class)。第三层才是第二层中各种类 (class) 的 “实例” (instance)。我们最常接触和操作的字符串 (str) `'hello'`、整数 (int) `42`、浮点数 (float) `2.5`、布尔值 (bool) `True` 和 `False`、列表 (list) `[7, 9, 8]`、元组 (tuple) `('a', 2, True)`、字典 (dict) `{'a': 8, 'b': 3}` 等等都是处于第三层的 “实例” (instance)。我们所使用的内置函数 (built-in functions)、从模块 (module) 中导入的现有函数 (function)、以及我们自己用 `def` 语句定义的函数 (function)，也都是处于第三层中的 “实例” (instance)。第三层之下再没有了，因为在第三层中的任何对象 (object) 都不可能作为 “模板” 或 “蓝图” 去生成更加具体的实例 (instance)。所以第三层就是底层。

在学用 Python 的初级阶段，只在第三层 “实例层” 中工作就已经足够了。我们可以定义函数、编写脚本，借助 Python 完成我们的具体现实工作。在学用 Python 的中、高级阶段，我们可以进入第二层编写各种类 (class)，把常见的各类专业工作的逻辑进行抽象，编写成可供他人使用的软件包 (package) 发表在 Python Package Index。在 Python 的非常高级的阶段，在有最灵活的编程需要的时候，才会进入最顶层去编写类 (class) 的 “类”，这叫做 Python 元编程 (meta programming)。

## 24 简答题

对象 (object) 和实例 (instance) 这两个概念之间究竟是什么关系?

## 25 简答题

Python 对象 (object) 可以分为哪三个层次？简要介绍你对三个层次的理解。

## 26 简答题

面向对象编程 (OOP) 的思维方式有什么用?

## 27 简答题

在定义类 (class) 时，`__init__` 方法 (method) 有什么用处？这个方法在什么时候会被调用 (called)?

## 28 简答题

Python 所有自定义类 (class) 的实例 (instance) 都附带着一个 `__dict__` 属性 (继承自 `object`)，对于这个属性的类型 (class) 和用途你有什么认识?

## 29 简答题

在自定义类 (class) 的实例方法 (instance method) 的时候，第一个形参 (parameter) 必须是 `self`。这个 `self` 是什么意思？有什么用?

---