面向对象方法学 -- Python

1. Python 简介

Python 是一个高层次的结合了`解释性`、`编译性`、`互动性`和`面向对象`的脚本语言，具有比其他语言更有特色的语法结构。

- **Python 是解释型语言** -- 没有编译环节；
- **Python 是交互性语言**
- **Python 是面向对象语言**
- **Python 是初学者的语言** -- 简单易学，应用广泛

2. Python 特点

- **优点**
  - **简单易学**
  - **免费开源**
  - **高层语言** -- 无需考虑底层细节
  - **可移植性**
  - **解释器**
  - **面向对象** -- Python 既支持面向过程的编程也支持面向对象的编程
  - **可扩展性** -- 如果你需要你的一段关键代码运行得更快或者希望某些算法不公开，你可以把你的部分程序用 C 或 C++ 编写，然后在你的 Python 程序中使用它们
  - **丰富的标准库**
  - **规范的代码** -- 采用强制缩进的方式使代码具有极佳的可读性
- **缺点**
  - **运行速度慢** -- 可使用 C++ 改写关键部分
  - **框架选择太多**
  - **国内市场较小**
  - **中文资料匮乏**

注： 以上论述摘自菜鸟笔记

3. 基本概念和基本机制

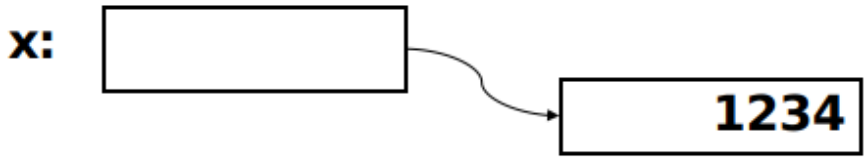
3.1 值、对象和变量

- **数据类型**
  - Number (数字) -- **int、float、bool、complex（复数）**
  - String (字符串)
  - List (列表)
  - Tuple (元组)
  - Set (集合)
  - Dictionary (字典)
- **注：**
  - 不可变数据 -- Number (数学)、String (字符串)、Tuple (元组)

- 可变数据 -- List (列表)、Dictionary (字典)、Set (集合)

3.2 变量模式

- **引用模式：**变量里保存对一个数据对象的引用，数据对象里保存着数据值



3.3 表达式计算

- **运算符**
  - `计算运算符` -- 加(+) 减(-) 乘(*) 除(/) 整除 // 取模(%)
  - `比较运算符` -- 大于(>) 小于(<) 等于(==) 大于等于(>=) 小于等于(<=) 不等于(!=)
  - `逻辑运算符` -- 布尔与（and） 布尔或（or） 布尔非（not）
  - `布尔运算符` -- 真（True） 假（False）
  - `赋值运算符`
  - `位运算符` -- 与(&) 或(|) 异或(^) 非(~)
  - `成员运算符` -- in / not in
  - `身份运算符` -- is / is not

3.4 语句和控制结构

- **条件控制**
  - if - elif - else
- **循环结构**
  - while 循环
    - while 循环使用 `else` 语句

```
count = 0
while count < 5:
   print (count, " 小于 5")
   count = count + 1
else:
   print (count, " 大于或等于 5")
```

  - for 循环

```
for <variable> in <sequence>:
    <statements>
else:
    <statements>
```

  - range() 函数
    - 遍历数字序列

```
for i in range(5):
  print(i)
```

- break/continue/pass

3.5 作用域和生存期

名字	定义	生存期
Local	本地作用域、局部作用域的local命名空间。在函数内（def或lambda）通过使用方式赋值，且没有通过global声明为全局变量。	函数调用时(非定义时)创建，调用结束时消亡。
Enclosing	python2引入了嵌套函数，实现闭包。这个就是嵌套函数的外部函数的命名空间。上层函数的本地作用域。	函数调用时(非定义时)创建，调用结束时消亡。
Global	全局作用域：即一个模块的命名空间。在模块文件顶层赋值的变量名，或者文件的def内生成的名为全局变量的变量名。	模块被import时创建，解释器退出时消亡。
Build-in	内置模块的命名空间。例如prrint(open)，print和open都是内置的变量。	python解释器启动时创建，解释器退出时消亡。

- **作用域**
  - 全局作用域 -- 全局存活，全局有效
  - 局部作用域 -- 局部存活，局部有效

```
def f1():
    x = 1
    y = 2
    print(locals())
    print(globals())
f1()
print(locals())
print(globals())
print(locals() is globals())

改全局名称:
x = 1
def f1():
    global x
    x = 2
f1()
print(x)

改局部名称:
x = 0
def f1():
    x = 1
```

```
    def f2():
        x = 2
        def f3():
            nonlocal x#改的是函数正上方的值
            x = 3
        f3()
    f2()
f1()
```

3.6 名字空间

- **内置名称空间：** python自带的名字，在python解释器启动时产生，存放一些python内置的名字;
- **全局名称空间：** 在执行文件时，存放文件级别定义的名字;
- **局部名称空间：** 在执行文件的过程中，如果调用了函数，则会产生该函数的名称空间，用来存放该函数内定义的名字，该名字在函数调用时生效，调用结束后失效。
- **加载顺序：** 内置名称空间------>全局名称空间----->局部名称空间
- **名字查找顺序：** 局部名称空间------>全局名称空间----->内置名称空间