

web安全-后端Python

一 python 介绍

二 python版本

三 python基本语法

注释

变量

输出、输入函数

Python 六个标准的数据类型

Python 数字

Python 字符串

Python 列表

Python 元组

Python 字典

python 流程语句

if-else 条件语句

基本用法

行和缩进

if-elif-else 语句

while 循环语句

continue 和 break 用法

for 循环语句

四 python 函数

函数的特点

基本用法

五 python模块

Python模块特点

python模块分类

系统内置模块

自定义模块

第三方模块

pip 工具的安装及使用

常见的 pip 命令

列出已经安装的所有模块

安装指定模块的最新版本

设置 pip 的下载源

使用清华镜像源安装第三方模块

卸载模块

六 python类和对象

类 (Class)

对象 (Object)

例子

Python魔术方法

`__init__`

`__del__`

`__getstate__`

`__setstate__`

`__reduce__`

一 python 介绍

Python 是一种解释型、面向对象、动态数据类型的高级编程语言。

它被设计为可读性强、简洁且易于学习，具有高效的高级数据结构，并且支持简单有效的面向对象编程。

Python官网: <https://www.python.org>

二 python版本

目前 python 的主要版本包括 python2 和 python3

Python2 已经在 2020 年 1 月 1 日之后不再被官方支持，建议使用 Python 3 进行开发和编程。

Python3 本身也有多个子版本，例如 Python 3.6、Python 3.7、Python 3.8、Python 3.9 以及更新的

版本。

每个子版本都包含了对语言本身的改进和新功能的添加。

三 python基本语法

注释

单行注释： `#`

多行注释： `三个单引号` 或 `三个双引号`

注意：

`"""`引号中的内容为注释部分`"""`

`"""`注释引号必须成对出现`"""`

A screenshot of a code editor window titled 'demo1.py'. The editor shows five lines of code. Line 1 is a single-line comment: `# print("hello world")单行注释`. Line 2 is empty. Line 3 is the start of a multi-line comment: `'''多行注释'''`. Line 4 is a single-line code: `print(1223)`. Line 5 is another single-line code: `print(789)'''`. The code is color-coded: comments are grey, and code is green.

变量

变量是用于存储数据的容器，其值可以在程序运行期间被改变。变量通常包括名称和值，名称用于标识变量，值则是变量存储的数据。

```
1 a = 10 # a 为变量 10 为值 =为赋值符号
```

输出、输入函数

```
1 print("Hello, Python!") # 输出括号中的内容
2 name = input('请输入你的姓名: ')
```

Python 六个标准的数据类型

Numbers (数字)

String (字符串)

List (列表)

Tuple (元组)

Set (集合)

Dictionary (字典)

Python 数字

数字数据类型用于存储数值。他们是不可改变的数据类型，这意味着改变数字数据类型会分配一个新的对象

```
1 var1 = 1
2 var2 = 10
```

Python 字符串

```
1 s = 'abcdef'
2 print(s[1:5])
3
4 str = 'Hello World!'
5 print(str) # 输出完整字符串
6 print(str[0]) # 输出字符串中的第一个字符
7 print(str[2:5]) # 输出字符串中第三个至第六个之间的字符串
```

Python 列表

```
1 list = ['Beyond', 786, 2.23, 'john', 70.2]
2
3 print(list) # 输出完整列表
4 print(list[0]) # 输出列表的第一个元素
5 print(list[1:3]) # 输出第二个至第三个元素
6
```

Python 元组

元组是另一个数据类型，类似于 List（列表）。元组用 () 标识。内部元素用逗号隔开。但是元组不能二次赋值，相当于只读列表。

```
1 tuple = ('Beyond', 786, 2.23, 'john', 70.2)
2
3 print(tuple) # 输出完整元组
4 print(tuple[0]) # 输出元组的第一个元素
5 print(tuple[1:3]) # 输出第二个至第四个（不包含）的元素
```

Python 字典

```
1 tinydict = {'name' : 'Beyond', 'code' : 6734, 'dept' : 'sales'}
2 print(tinydict) # 输出完整的字典
```

python 流程语句

在 python 中，命令语句的执行顺序是从上往下的，遇到命令运行错误时会终止。

if-else 条件语句

基本用法

if 条件：

条件为真执行的语句

else:

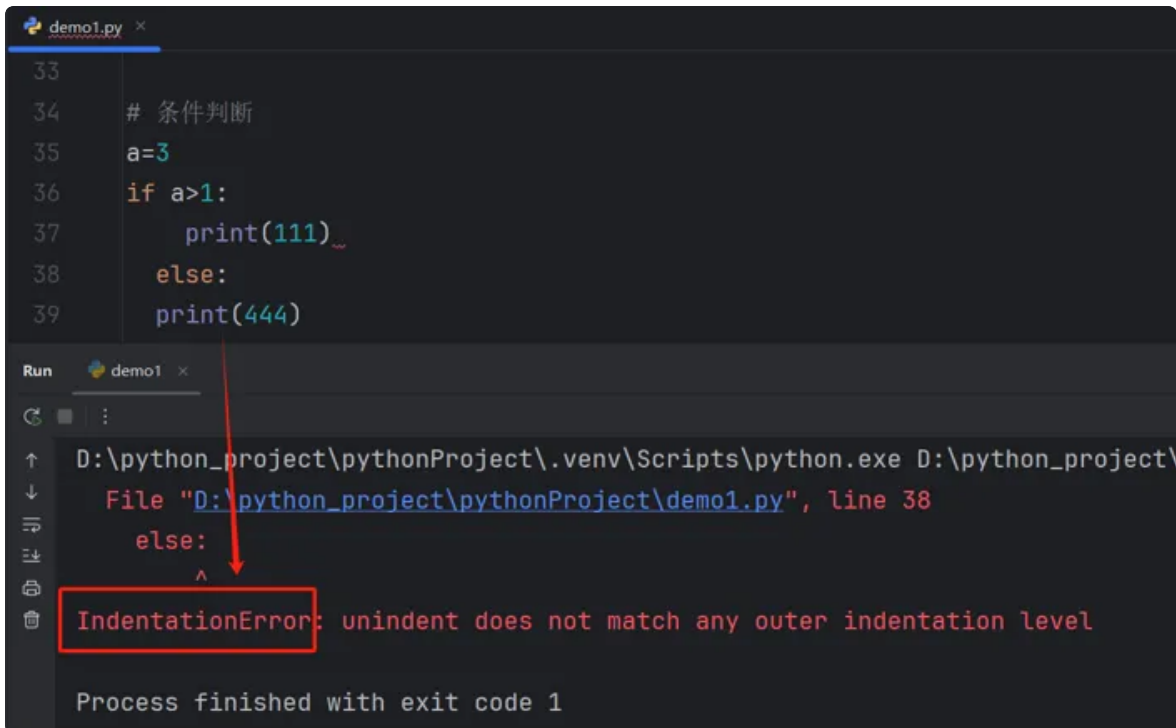
条件为假执行的语句

```
1 name = 'xiaolin'
2 if name == 'xiaolin': # 判断变量是否为 python
3     print('welcome boss') # 并输出欢迎信息
4 else:
5     print(name, '不是 xiaolin') # 条件不成立时输出变量名称 ``
```

行和缩进

学习 Python 与其他语言最大的区别就是，Python 的代码块不使用大括号 {} 来控制类，函数以及其他逻辑判断。python 最具特色的就是用缩进来写模块。

缩进的空白数量是可变的，但是所有代码块语句必须包含相同的缩进空白数量，这个必须严格执行。



```
demo1.py x
33
34 # 条件判断
35 a=3
36 if a>1:
37     print(111)
38 else:
39     print(444)

Run demo1 x
D:\python_project\pythonProject\.venv\Scripts\python.exe D:\python_project\
File "D:\python_project\pythonProject\demo1.py", line 38
    else:
    ^
IndentationError: unindent does not match any outer indentation level

Process finished with exit code 1
```

if-elif-else 语句

基本用法

if 条件 1 :

 条件 1 为真执行的语句

elif 条件 2 :

 条件 2 为真执行的语句

elif 条件 3 :

 条件 3 为真执行的语句

elif 条件 4 :

 条件 4 为真执行的语句

else:

 以上条件为假执行的语句

```
1 num = 5
2 ▾ if num == 3 : # 判断 num 的值
3     print('3')
4 ▾ elif num == 2 :
5     print('2')
6 ▾ elif num == 1 :
7     print('1')
8 ▾ else:
9     print('other') # 条件均不成立时输出
```

while 循环语句

基本用法

while 条件：

条件为真 重复执行的代码

```
1 count = 0
2 ▾ while count < 10 :
3     print('The count is :', count)
4     count = count + 1
5 print("Good bye!")
6
```

continue 和 break 用法

例 1:

```
1 i = 1
2 ▾ while i < 10 :
3     i += 1
4 ▾ if i % 2 != 0 : # 非双数时跳过输出
5     continue
6     print(i) # 输出双数 2、4、6、8、10
```

例 2:

```
1 i = 1
2 while 1 : # 循环条件为 1 必定成立
3     print(i) # 输出 1~10
4     i += 1
5     if i > 10 : # 当 i 大于 10 时跳出循环
6         break
```

for 循环语句

基本用法

for 临时变量 in 容器 :

重复执行的代码

例 1

```
1 for letter in 'Python':
2     print("当前字母 : %s" % letter)
```

例 2

```
1 fruits = ['banana', 'apple', 'mango']
2 for fruit in fruits :
3     print('当前水果 : %s' % fruit)
```

例 3

```
1 for i in range(1,10):
2     print('当前数字: ',i)
```

四 python 函数

函数的特点

- 1、函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。
- 2、函数使用 def 关键字开头，后接函数标识符名称和圆括号()。

3、任何传入参数和变量必须放在圆括号中间。圆括号之间可以用于定义参数。

4、函数主体内容以冒号开始，函数中的代码块必须缩进

`return [表达式]` 结束函数，选择性地返回一个值给调用方。不带表达式的 `return` 相当于 返回 `None`

基本用法

`def 函数名(参数):`

 "函数文档字符串"

 代码 1

 代码 2

```
1
2  # 例 1
3  def printme(name) :
4      "打印传入的字符串输出"
5      print('我的名字叫: ',str)
6
7  # 调用函数
8  printme('张三')
9
10
11
12  # 例 2
13  def sum(arg1, arg2) :
14      # 返回 2 个参数的和."
15      total = arg1 + arg2
16      print("函数内 : ", total)
17      return total
18
19  # 调用 sum 函数
20  total = sum(10, 20)
21  print(total) ``
22
```

五 python模块

是包含Python代码的文件，其扩展名通常为 `.py` 。模块是Python程序的基本组成部分，它们提供了封装代码的方式，使得代码更加组织化、易于重用和维护。模块可以包含函数、类和变量，也可以包含可执行的代码。

Python模块特点

1. **封装性**：模块可以将相关的函数、类和变量封装在一起，形成一个独立的代码单元。这有助于保持代码的整洁和组织性。
2. **重用性**：一旦一个模块被编写和测试完毕，它就可以被多个程序重复使用，而无需重复编写相同的代码。
3. **可维护性**：模块化的代码更容易维护和更新。如果需要修改某个功能，只需要修改相应的模块，而不需要修改整个程序。
4. **命名空间管理**：每个模块都有一个独立的命名空间，这有助于避免命名冲突。当两个模块包含相同名称的函数或变量时，它们不会相互干扰，因为它们是在不同的命名空间中定义的。

python模块分类

系统内置模块

系统内置模块是Python自带的标准库，它们提供了丰富的功能和工具，用于执行各种常见的任务。这些模块通常与Python解释器一起安装，无需额外下载或安装。

- **os模块**：提供了与操作系统交互的功能，如文件操作、目录操作等。
- **sys模块**：提供了与Python解释器及其环境相关的功能，如命令行参数、解释器版本等。
- **random模块**：用于生成随机数。
- **time模块**：提供了各种与时间相关的功能，如获取当前时间、格式化时间等。

自定义模块

自定义模块是开发者自己编写的模块，用于封装特定的功能或逻辑。自定义模块可以是任何有效的Python文件（以“.py”为后缀名），里面可以包含全局变量、函数、类等。在使用自定义模块时，需要确保模块的命名不与系统内置模块重名，否则将无法导入系统内置模块。自定义模块的使用方式与其他模块相同，通过**import语句**进行导入。

第三方模块

第三方模块是由**其他开发者或组织编写的**，并发布到Python包索引（PyPI）等公共仓库中的模块。这些模块通常提供了特定领域或特定任务的高级功能和工具，如数据分析、网络请求、图像处理等。要使用第三方模块，**需要先通过pip等包管理工具进行安装。**

- **Requests**：一个简单而强大的HTTP请求库，用于发送HTTP请求并处理响应。

- **Pandas**: 一个强大的数据分析和操作库，提供了高效的数据结构和数据分析工具。
- **NumPy**: 一个支持大规模多维数组和矩阵运算的库，提供了数学函数库来操作这些数组。

pip 工具的安装及使用

pip 是 Python 的包管理工具，通过 pip 可以方便地安装、卸载和管理 Python 的第三方库。

Python3.X 默认已安装 pip，Python2.x 可在终端输入 `python get-pip.py` 命令来安装 pip 工具。安装完 pip 之后，就可以通过 pip 命令来管理和安装 Python 的第三方库了。

常见的 pip 命令

列出已经安装的所有模块

```
pip list
```

安装指定模块的最新版本

```
pip install 模块名
```

例如：`pip install python-whois`

设置 pip 的下载源

很多时候，比如网络不给力，连接超时、防火墙阻挡等等各种原因，可能无法从 Python 官方的 PyPi 仓库进行 pip 安装，这时候可以选择国内的镜像源。常见的镜像源如下：

- 1 清华：<https://pypi.tuna.tsinghua.edu.cn/simple>
- 2 阿里云：<http://mirrors.aliyun.com/pypi/simple/>
- 3 中国科技大学 <https://pypi.mirrors.ustc.edu.cn/simple/>
- 4 华中理工大学：<http://pypi.hustunique.com/>
- 5 山东理工大学：<http://pypi.sdutlinux.org/>
- 6 豆瓣：<http://pypi.douban.com/simple/>

使用清华镜像源安装第三方模块

```
pip install python-whois -i https://pypi.tuna.tsinghua.edu.cn/simple
```

卸载模块

```
pip uninstall python-whois
```

六 python类和对象

在Python中，类和对象是面向对象编程（OOP）的两个核心概念。它们共同构建了一个强大的编程框架，使得开发者能够创建结构化、模块化和可复用的代码。

类（Class）

类是一个模板或蓝图，它定义了对对象的结构和行为。类包含了属性和方法，这些属性和方法定义了对对象将拥有的数据（状态）以及可以执行的操作（行为）。

- 属性：类的属性可以是数据属性或类属性。数据属性用于存储对象的特定信息（如实例变量），而类属性则是类级别的变量，被该类的所有实例共享。
- 方法：类的方法定义了对对象的行为。它们是绑定到类的函数，可以通过对象来调用。方法中的第一个参数通常是self，它代表调用该方法的对象本身。

对象（Object）

对象是类的实例。每个对象都是根据类模板创建的具体实体，它包含了类定义的属性和方法。

例子

```
1 class people:    #创建people类
2     #在类里面写了一个构造方法初始化在类实例化时会自动调用
3     def __init__(self):
4         print("初始化函数，打印构造函数")    #函数内打印内容
5
6 p1 = people()    #实例化对象
```

Python魔术方法

Python中的魔术方法，也被称为双下划线方法或特殊方法，是一类具有特殊命名规则的方法。它们的主要作用是给Python中的对象提供内置的、特殊的行为。这些方法的名称前后都带有两个下划线，例如__init__、__str__、__add__等。

魔术方法的命名是固定的，不能随意更改。如果你尝试定义一个不符合命名规范的魔术方法，它将不会被Python解释器识别为特殊方法。

`__init__`

- 触发机制：在实例化对象之后立即触发
- 参数：至少有一个self接收当前对象，其他参数根据需要进行定义
- 作用：初始化对象的成员

`__del__`

python的垃圾回收机制，回收不是"立即"的，由解释器在适当的时机，将垃圾对象占用的内存空间回收。

- 触发机制：当该类对象被销毁时，当对象不再被使用时自动触发
- 参数：一个self接收当前对象
- 作用：关闭或释放对象创建时资源

`__getstate__`

在对象被序列化`pickle.dump()`时调用，返回一个表示对象状态的字典。默认返回`__dict__`，但可以自定义以控制序列化的内容（例如排除敏感数据或不可序列化的属性）

`__setstate__`

在对象反序列化 `pickle.load()` 时调用，接收`__getstate__`返回的字典，用于恢复对象状态。常用于重建默认`__dict__`无法处理的复杂对象

`__reduce__`

定义对象如何被pickle模块序列化，返回一个元组 (callable, args, state)