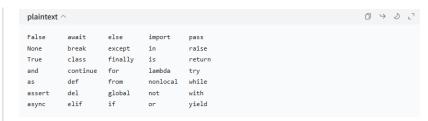
- 自动化邮件的代码案例,以及使用pdb模式理解代码。
 - 新建week04的python环境 conda env create
 - vscode中打开week04文件夹新建一个 main.py 文件, 里面写 Python 代码(自动化发邮件)
 - 在终端中运行代码 python main.py
 - 用pdb模式运行代码(调试模式): python -m pdb main.py
 - (pdb) 提示符下使用 I (显示代码)、n (执行当前行)、p (打印表达式)、s (步入调用)、pp (美观打印)、c (继续执行)等命令

Python基本概念

• **python保留字**:是 Python 语言里具有特定含义和用途的单词,不能将其用作变量名、函数名或者其他标识符。



- 语句 (statement) 和表达式 (expression):
 - 语句: 语句是 Python 程序中执行操作的基本单位,它通常用于完成某种行为或控制程序的流程,不直接返回一个值(虽然有些语句会有副作用,比如修改变量的值)。



表达式:表达式是由变量、常量、运算符和函数调用等组成的代码片段,它会计算并返回一个值。表达式通常作为语句的一部分,也可以单独使用。

- **缩进 (indent)**:缩进的作用是界定代码块的范围。与其他部分编程语言(像 C、Java 等)运用大括号 {} 来划分代码块不同,Python 采用缩进来明确代码块的起始与结束位置。
 - 在 Python 中,相同缩进层次的代码会被视作一个代码块。例如,if、for、while、def、class 等语句后面,通过缩进表示这些语句所管辖的代码范围。
- 局部变量 (local variable)、全局变量 (global variable)、LEGB 规则
 - 局部变量是在函数或类的方法内部定义的变量,其作用域仅限于定义它的函数或方法内部。也就是说,局部变量只能在定义它的函数或方法内部被访问和使用,函数或方法执行完毕后,局部变量就会被销毁。
 - **全局变量**是在函数和类的外部定义的变量,其作用域是整个模块。全局变量可以在模块的任何地方被访问,但如果要在函数内部修改全局变量的值,需要使用 global 关键字进行声明。

```
python ^
                                                                          0 4 0 7
# 全局变量: 用于记录所有轮次的总得分
total_score = 0
def play round():
   # 局部变量: 用于记录当前轮次的得分
   round_score = 0
   # 模拟玩家在本轮游戏中的操作,这里简单随机生成一个 1 到 10 之间的得分
   round_score = random.randint(1, 10)
   print(f"本轮游戏得分: {round_score}")
   # 使用 global 关键字声明要修改全局变量
   global total score
   total score += round score
def display_total_score():
   print(f"目前总得分: {total_score}")
# 进行三轮游戏
for _ in range(3):
   play round()
   display total score()
```

- total_score 是一个全局变量,它在函数外部定义,用于记录所有轮次的总得分。在 play_round 函数中,使用 global total_score 声明后,就可以在函数内部修改这个全局 变量的值。
- round_score 是 play_round 函数内部的局部变量,它只在该函数内部有效。每一轮 游戏开始时,round_score 会被重新初始化为 0,然后根据随机数生成当前轮次的得分。
- LEGB 规则

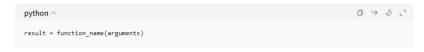
- LEGB 规则是 Python 中用于查找变量的优先级顺序,它代表了四个不同的作用域:
- Local (局部作用域):函数或方法内部的作用域,局部变量的作用域。
- Enclosing (闭包作用域): 当一个函数嵌套在另一个函数内部时,外部函数的作用域就是闭包作用域。闭包作用域允许内部函数访问外部函数的局部变量。
- Global (全局作用域): 模块级别的作用域, 全局变量的作用域。
- Built-in(内置作用域): Python 内置函数和内置异常的作用域,这些是 Python 解释器预先定义好的。
- 当 Python 解释器在查找一个变量时,会按照 LEGB 的顺序依次查找,即先在局部作用域查找,如果找不到,就到闭包作用域查找,接着是全局作用域,最后是内置作用域。如果在所有作用域中都找不到该变量,就会抛出 NameError 异常。

• 函数 (function) 的定义 (define) 和调用 (call)

函数的定义:定义函数时,需要明确函数的名称、参数和函数体。Python 中使用def关键字来定义函数。



函数的调用:定义好函数之后,就可以在代码的其他地方调用该函数。调用函数时,需要提供函数所需的参数(如果有参数的话),并可以接收函数的返回值。



• 字面值 (literal) (字符串 (str)、整数 (int)、列表 (list)、字典 (dict)、元组 (tuple))

- 字符串(str)是由零个或多个字符组成的序列,Python 里可以用单引号(')、双引号(")或者三引号('''或""")来表示字符串字面值。
- 整数(int)是没有小数部分的数字,可以是正数、负数或零。Python 中的整数字面值直接书写数字即可。
- 列表(list)是一种可变的、有序的数据集合,用方括号([])来表示,列表中的元素可以是不同类型,元素之间用逗号分隔。

```
python ^
# 包含不同类型元素的列表
mixed_list = [1, 'apple', True, 3.14]
# 空列表
empty_list = []
print(mixed_list)
print(empty_list)
```

字典(dict)是一种无序的、可变的数据集合,由键值对组成,用花括号({})表示。每个键值对之间用逗号分隔,键和值之间用冒号(:)分隔。

元组是一种不可变的、有序的数据集合,用圆括号())表示,元素之间用逗号分隔。如果元组只有一个元素,需要在元素后面加逗号。

- 运算符: 用于对数据进行操作的特殊符号。如: +- 'if' 'else'等等
- 形参 (parameter)、实参 (argument)、返回值 (return value)
 - 形参是在定义函数时,函数名后面括号中声明的变量,它的作用是接收调用函数时传递过来的值。形参就像是函数内部的占位符,在函数定义时并不具备实际的值,只有在函数被调用时,才会被赋予具体的值。(a和b为形参)

```
python ^

def add_numbers(a, b):
    return a + b
```

 实参是在调用函数时,传递给函数的具体值。实参是形参的具体实例,当函数被调用 时,实参的值会被赋给对应的形参。(3和5为实参)

```
python ^

result = add_numbers(3, 5)

print(result)
```

- 返回值是函数执行完毕后,通过 return 语句返回给调用者的结果。函数可以有返回值,也可以没有返回值(如果没有 return 语句,函数默认返回 None)。返回值可以是任意类型的数据,如数字、字符串、列表、字典等。
- 对象 (object)、类型 (type)、属性 (attribute)、方法 (method)

1. 对象(Object)

对象是类的实例,它将数据(属性)和操作这些数据的代码(方法)封装在一起。在现实世界里,对象可以类比成具体的事物,像一辆汽车、一只猫等。在编程里,对象是代码层面对于现实事物的抽象表示。

以下是一个 Python 示例:

在这个例子中,my_car 就是 Car 类的一个对象。

2. 类型 (Type)

类型定义了对象的类别,它决定了对象能够拥有的属性和方法。不同的类型有不同的行为和特征。在 Python里,每个对象都有其对应的类型。你可以使用 type() 函数来查看对象的类型。

延续上面的例子:

这里表明 my_car 对象的类型是 Car 类。

3. 属性 (Attribute)

属性是对象所拥有的数据,它描述了对象的状态和特征。属性可以是各种数据类型,如整数、字符串、列表等。在 Python 中,属性可以通过对象的实例变量来定义。

还是以上面的 Car 类为例:

```
python ^ ① 今 む い
print(my_car.brand) # 輸出 Toyota
print(my_car.color) # 輸出 blue
```

brand和 color就是 my_car 对象的属性。

4. 方法 (Method)

方法是对象能够执行的操作,它是与对象关联的函数。方法可以访问和修改对象的属性,并且可以执行特定的任务。在 Python 中,方法是定义在类中的函数。

以 Car 类的 start_engine 方法为例:

start_engine 就是 my_car 对象的一个方法,调用这个方法可以启动汽车的引擎。