

Python编程从入门到实践

第一部分 基础知识

第1章 起步

配置Python环境教程（采用conda 创建虚拟环境 防止环境混乱 因为不同的项目需要涉及不同版本的安装包）：

最新版最详细Anaconda新手安装+配置+环境创建教程 anaconda配置-CSDN博客

VS Code配置Python环境

10分钟搞定！VS Code配置Python开发环境指南（2025新版） - 知乎

可能会出现的问题：

Warning

PS E:\Python编程 从入门到实践> (D:\Anaconda\shell\condabin\conda-hook.ps1) ; (conda activate streamlit)

>>>>>>>>>>>>>>>> ERROR REPORT <<<<<<<<<<<<<<<<

```
Traceback (most recent call last): File "D:\Anaconda\Lib\site-  
packages\conda\exception_handler.py", line 16, in call return func(*args, **kwargs) ^^^^^^  
File "D:\Anaconda\Lib\site-packages\conda\cli\main.py", line 111, in main_sourced  
print(activator.execute(), end="") UnicodeEncodeError: 'gbk' codec can't encode character  
\u202a' in position 410: illegal multibyte sequence
```

因为系统环境变量被Unicode控制字符污染 可能是之前复制网页中的系统路径时无意中带入

成功截图：

```
example > helloworld.py
```

问题 输出 调试控制台 终端 端口

```
(streamlit) PS E:\Python编程 从入门到实践 & D:\Anaconda\envs\streamlit\python.exe "e:/Python编程 从入门到实践/example/helloworld.py"
Hello World!
```

第1章总结到此结束！！！

第2章 变量和简单的数据类型

之后会采用pycharm来进行实例运行 因为用习惯了😊

变量

💡 非常的浅显易懂

```
message = "Hello Python world!"  
print(message)  
message = "Hello Python Crash Course world!"  
print(message)  
#在程序中，可随时修改变量的值，而Python将始终记录变量的最新值
```

```
D:\Anaconda\envs\streamlit\python.exe "E:\Python编程 从入门到实践\example\2. 变量和简单的数据类型.py"  
Hello Python world!  
Hello Python Crash Course world!
```

message只是一个名字 而赋给它的值定义了它的类型 如果是5那就是整型 如果是""那就是字符串

Python是动态类型但**强类型语言**

```
10 + "a" # ✗ 直接报错
```

因此Python的变量不是“容器”，而是“标签” => Python用来做AI是非常合适的 因为AI需要随时调参讲究的是一个动态的变化 模型结构本身就是运行期对象，**语言必须允许结构动态变化**

💡 Tip

变量的命名规范

1. 变量名只能包含字母、数字和下划线。变量名能以字母或下划线开头，但不能以数字开头。

例： message_1 ✅ 1_message ✗

2. 变量名不能包含空格

3. 不要将Python关键字和函数名用作变量名 例如： print、 input等等

⚠ (剩余是工程上的规范 一是变量名应既简短又具有描述性，二是慎用小写字母l和大写字母O，因为它们可能被人错看成数字1和0)

#2.2.2 使用变量时命名错误

```
message = "Hello Python Crash Course reader!"  
print(mesage)
```

```
D:\Anaconda\envs\streamlit\python.exe "E:\Python编程 从入门到实践\example\2. 变量和简单的数据类型.py"  
Traceback (most recent call last):  
  File "E:\Python编程 从入门到实践\example\2. 变量和简单的数据类型.py", line 9, in <module>  
    print(mesage)  
           ^^^^^^  
NameError: name 'mesage' is not defined. Did you mean: 'message'?
```

⌚ Caution

上面的代码块我只截取了相关的部分 具体的整体章节代码在example文件夹下的2.变量和简单的数据类型.py中

通过Python解释器提供的报错信息可以发现是代码的第九行 一般变量xxx未定义的报错都是Python无法识别你提供的变量名 **根本在于使用变量前没有给它赋值** 这里的message可以当作一个新变量

字符串

字符串 (string) 就是一系列字符。在Python 中，用引号引起的都是字符串（可以是单引号也可以是双引号）

比较复杂的字符串

```
text = '''老师说: "今天的作业主题是'Python' 的字符串处理'，请大家认真完成，并在作业中写明: '我已经理解了双引号"和单引号'的区别'。" '''
print(text)

#这里采用了三引号 能够实现以下两个功能（一般用不上） PS: 中文的“和英文的”是有区别的
#1.可以跨行
#2.里面可以随便放'和"

#如果不想用'''三引号却又想在字符串里用"如何解决？
#采用转义字符
message = "The language \"Python\" is named after Monty Python, not the snake."
print(message)
输出: The language "Python" is named after Monty Python, not the snake.
#这里的\就是告诉解释器不需要扫描匹配"字符直接输出=>也就是\\"在解释器看来就是直接输出"
```

Important

字符串相关操作函数

1.title()函数

```
# title() 将每个单词的首字母转换为大写
name = "tian lin ying"
print(name.title())

#输出: Tian Lin Ying
```

2.upper()/lower()函数

```
# upper()/lower() 将每个单词大写/小写
name = "Tian Lin Ying"
print(name.upper())
print(name.lower())

#输出: TIAN LIN YING
tian lin ying
```

在字符串中使用变量

```
first_name = "lin ying"
last_name = "tian"
fullname = f"{first_name} {last_name}"
print(fullname)

#输出: lin ying tian
```

Tip

要在字符串中插入变量的值，可先在左引号前加上字母f，再将要插入的变量放在花括号内。

这种字符串称为**f字符串** (format-设置格式) Python通过把花括号内的变量替换为其值来设置字符串的格式

```
first_name = "lin ying"
last_name = "tian"
fullname = f"{first_name} {last_name}"
print(f"Hello, {fullname.title()}!")

#输出: Hello, Lin Ying Tian!
```

也可以使用**f字符串**来创建消息

```
first_name = "lin ying"
last_name = "tian"
fullname = f"{first_name} {last_name}"
message = f"Hello, {fullname.title()}!"
print(message)

#输出: Hello, Lin Ying Tian!
```

#将消息赋给了一个变量 让最后的函数调用print()简单得多

实际案例：

```
out_name = out_nameitems[attribute_box.currentIndex()]
sql_el = textwrap.dedent(f'''
    SELECT
        '当前表' as 表名,
        COUNT(*) as 记录数,
        SUM(a.{attribute_box.currentText()}) as {out_name}总数
    FROM {textbox_30.text()} a
    WHERE a.init_date between 20251001 and 20251031
    UNION ALL
    SELECT
        '历史表' as 表名,
        COUNT(*) as 记录数,
        SUM(a.{attribute_box.currentText()}) as {out_name}总数
    FROM {textbox_20.text()}@uf20_his a
    WHERE a.init_date between 20251001 and 20251031
'''')
```

```
out_sql.setPlainText(sql_e1)
```

这里的代码有一点复杂 因为涉及到了pyqt的知识 可以把花括号里的内容当作变量 因为很多部分基本上差不多的 因此脚本只需要替换变的地方即可=>用变量来实现，最后用f字符串进行拼接

使用制表符或换行符来添加空白

```
#制表符和换行符
print("Python")
print("\tPython")

#输出:
#Python
#      Python
#\t缩进4个字符

print("Languages:\nPython\nC\nJavaScript")

#输出:
#Languages:
#Python
#C
#JavaScript
#\n换行

print("Languages:\n\tPython\n\tC\n\tJavaScript")
#输出:
#Languages:
#      Python
#      C
#      Javascript
```

Important

字符串相关函数

1.rstrip()函数

```
#rstrip() 删除字符串右端空白
favorite_language = ' python '
print(favorite_language.rstrip())

#输出: (此处为空格) python
```

2.lstrip()函数

```
#lstrip() 删除字符串左端空白
favorite_language = ' python '
print(favorite_language.lstrip())

#输出: python (此处为空格)
```

3.strip()函数

```
#strip() 删除字符串两端空白  
favorite_language = ' python '  
print(favorite_language.strip())
```

#输出: python

4.removeprefix()函数

```
#removeprefix() 删除前缀  
nostarch_url = 'https://nostarch.com'  
simple_url = nostarch_url.removeprefix('https://')  
print(simple_url)
```

#输出: nostarch.com

5.removesuffix()函数

```
#removesuffix() 删除后缀  
filename = "python_notes.txt"  
print(filename.removesuffix('.txt'))
```

#输出: python_notes

数

Tip

整数

加减乘除

**表示乘方运算

浮点数

```
print(0.2 + 0.1)  
#输出: 0.30000000000000004
```

#结果包含的小数位数可能是不确定的 所有编程语言都存在这种问题，没有什么可担心的 后续会讲处理多余小数位的方式

整数和浮点数

1.将任意两个数相除，结果总是浮点数

2.只要有操作数是浮点数，默认得到的就是浮点数，即便结果原本为整数

数中的下划线

在书写很大的数时，可使用下划线将其中的位分组，使其更清晰易读

```
universe_age = 14_000_000_000
print(universe_age)

#输出: 14000000000
#在存储这种数时, Python会忽略其中的下划线
```

同时给多个变量赋值

```
x, y, z = 0, 0, 0
```

用逗号将变量名分开; 对于要赋给变量的值, 也需要做同样的处理。Python将按顺序将每个值赋给对应的变量。只要变量数和值的**个数相同**, Python就能正确地将变量和值**关联起来**。

常量

Python没有内置的常量类型, 但Python程序员会使用全大写字母来指出应将某个变量视为常量

```
MAX_CONNECTIONS = 5000
```

注释

在Python中, 注释用井号 (#) 标识。井号后面的内容都会被Python解释器忽略