

实验二 Python变量、简单数据类型

班级：21计科3班

学号：B20210302317

姓名：覃思敏

Github地址：(<https://github.com/xx12qq/Experiment.git>)

CodeWars地址：(<https://www.codewars.com/users/xx12qq>)

实验目的

1. 使用VSCode编写和运行Python程序
2. 学习Python变量和简单数据类型

实验环境

1. Git
2. Python 3.10
3. VSCode
4. VSCode插件

实验内容和步骤

第一部分

实验环境的安装

1. 安装Python，从Python官网下载Python 3.10安装包，下载后直接点击可以安装：[Python官网地址](#)
 2. 为了在VSCode集成环境下编写和运行Python程序，安装下列VSCode插件
 - Python
 - Python Environment Manager
 - Python Indent
 - Python Extended
 - Python Docstring Generator
 - Jupyter
 - indent-rainbow
 - Jinja
-

第二部分

Python变量、简单数据类型和列表简介

完成教材《Python编程从入门到实践》下列章节的练习：

- 第2章 变量和简单数据类型

- 练习2.1:

```
message = "Hello Python world!"  
print(message)
```

输出：Hello Python world!

- 练习2.2:

```
message = "Hello Python world!"  
print(message)  
message = "Hello!"  
print(message)
```

输出：Hello Python world! Hello!

- 练习2.3:

```
name = "Hello Eric,would you like to learn some Python today?"  
print(name)
```

输出:Hello Eric,would you like to learn some Python today?

- 练习2.4:

```
name = "miKe"  
print(name.lower())  
print(name.upper())  
print(name.title())
```

输出: mike MIKE Mike

- 练习2.5:

```
print("Albert Einstein once said,A person who never made a mistake never tried  
anything new.")
```

输出:Albert Einstein once said,A person who never made a mistake never tried anything new.

- 练习2.6:

```
famous_person = "Albert Einstein"
message = "once said,A person who never made a mistake never tried anything new."
full_name = f"{famous_person} {message}"
print(full_name)
```

输出:Albert Einstein once said,A person who never made a mistake never tried anything new.

- 练习2.7:

```
name = '\t\tsdzfsas\nasdffxczf\t\t'
name = name.rstrip()
name = name.lstrip()
print(name)
```

输出: sdzfsas asdffxczf

- 练习2.8:

```
filename = 'python_notes.txt'
filename = filename.removesuffix('.txt')
print(filename)
```

输出:python_notes

- 练习2.9:

```
print(7+1)
print(10-2)
print(16/2)
print(4*2)
```

输出: 8 8 8.0 8

- 练习2.10:

```
i = 16
message = f"My favorite number is {i}"
print(message)
```

输出:My favorite number is 16

- 练习2.11:

```
#你们好呀  
print("I come from China")
```

输出:I come from China

- 练习2.12:

```
import this
```

输出:The Zen of Python, by Tim Peters

- Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated. Flat is better than nested. Sparse is better than dense. Readability counts. Special cases aren't special enough to break the rules. Although practicality beats purity. Errors should never pass silently. Unless explicitly silenced. In the face of ambiguity, refuse the temptation to guess. There should be one-- and preferably only one --obvious way to do it. Although that way may not be obvious at first unless you're Dutch. Now is better than never. Although never is often better than *right* now. If the implementation is hard to explain, it's a bad idea. If the implementation is easy to explain, it may be a good idea. Namespaces are one honking great idea -- let's do more of those!

第三部分

在[Codewars网站](https://www.codewars.com/)注册账号，完成下列Kata挑战：

第1题：求离整数n最近的平方数 (Find Nearest square number)

难度：8kyu

你的任务是找到一个正整数n的最近的平方数 例如，如果n=111，那么nearest_sq(n) (nearestSq(n)) 等于121，因为111比100（10的平方）更接近121（11的平方）。如果n已经是完全平方（例如n=144，n=81，等等），你需要直接返回n。代码提交地址 <https://www.codewars.com/kata/5a805d8cafa10f8b930005ba>

```
def nearest_sq(n):  
    return round(n**.5) ** 2
```

第2题：弹跳的球 (Bouncing Balls)

难度：6kyu

一个孩子在—栋高楼的第N层玩球。这层楼离地面的高度h是已知的。他把球从窗口扔出去。球弹了起来, 例如: 弹到其高度的三分之二（弹力为0.66）。他的母亲从离地面w米的窗户向外看,母亲会看到球在她的窗前经过多少次（包括球下落和反弹的时候）？

一个有效的实验必须满足三个条件：

- 参数 "h" (米) 必须大于0
- 参数 "bounce "必须大于0且小于1
- 参数 "window "必须小于h。

如果以上三个条件都满足，返回一个正整数，否则返回-1。注意:只有当反弹球的高度严格大于窗口参数时，才能看到球。 代码提交地址 <https://www.codewars.com/kata/5544c7a5cb454edb3c000047/train/python>

```
def bouncing_ball(h, bounce, window):
    n=1
    if window>=h or bounce>1 or bounce<=0 or h<=0:
        return -1

    if window>=h*bounce:
        return 1

    while window<h*bounce:
        n+=2
        h*=bounce
    return n
```

第3题：元音统计(Vowel Count)

难度：7kyu

返回给定字符串中元音的数量（计数）。对于这个Kata，我们将考虑a、e、i、o、u作为元音（但不包括y）。输入的字符串将只由小写字母和/或空格组成。

代码提交地址：<https://www.codewars.com/kata/54ff3102c1bad923760001f3>

```
def get_count(sentence):
    vowels = "aeiouAEIOU"
    count = 0
    for char in sentence:
        if char in vowels:
            count += 1
    return count
```

第4题：偶数或者奇数 (Even or Odd)

难度：8kyu

创建一个函数接收一个整数作为参数，当整数为偶数时返回“Even”当整数为奇数时返回“Odd”。代码提交地址：<https://www.codewars.com/kata/53da3dbb4a5168369a0000fe>

```
def even_or_odd(number):  
    if number % 2 == 0:  
        return "Even"  
    else:  
        return "Odd"
```


第四部分

使用Mermaid绘制程序流程图

安装Mermaid的VSCode插件：

- Markdown Preview Mermaid Support
- Mermaid Markdown Syntax Highlighting

使用Markdown语法绘制你的程序绘制程序流程图（至少一个），Markdown代码如下：

 程序流程图

```
graph TD  
    A[开始] --> B[判断number是否为偶数]  
    B -->|yes| C[返回Even]  
    B -->|no| D[返回Odd]  
    C --> E[返回Even]  
    D --> E[返回Odd]  
    E -->|yes| F[返回Even]  
    E -->|no| G[返回Odd]
```

查看Mermaid流程图语法--> [点击这里](#)

使用Markdown编辑器（例如VScode）编写本次实验的实验报告，包括[实验过程与结果](#)、[实验考查](#)和[实验总结](#)，并将其导出为 **PDF格式** 来提交。

实验过程与结果

请将实验过程与结果放在这里，包括：

- [第二部分 Python变量、简单数据类型和列表简介](#)
- [第三部分 Codewars Kata挑战](#)
- [第四部分 使用Mermaid绘制程序流程图](#)

注意代码需要使用markdown的代码块格式化，例如Git命令行语句应该使用下面的格式：

 Git命令

显示效果如下：

```
git init  
git add .
```

```
git status
git commit -m "first commit"
```

如果是Python代码，应该使用下面代码块格式，例如：

 Python代码

显示效果如下：

```
def add_binary(a,b):
    return bin(a+b)[2:]
```

代码运行结果的文本可以直接粘贴在这里。

注意：不要使用截图，**Markdown**文档转换为**Pdf**格式后，截图可能会无法显示。

实验考查

请使用自己的语言并使用尽量简短代码示例回答下面的问题，这些问题将在实验检查时用于提问和答辩以及实际的操作。

1. Python中的简单数据类型有那些？我们可以对这些数据类型做哪些操作？
2. 为什么说Python中的变量都是标签？
3. 有哪些方法可以提高Python代码的可读性？
4.
 - Python中的简单数据类型有：整数(int)、浮点数(float)、字符串(str)、布尔值(bool)、空值(None)等。我们可以对这些数据类型做以下操作：
 - 算术运算：加、减、乘、除、取模、幂等
 - 比较运算：等于、不等于、大于、小于、大于等于、小于等于
 - 逻辑运算：与、或、非
 - 字符串操作：连接、分割、替换、重复等
 - 列表操作：添加、删除、修改、遍历等
 - 元组操作：添加、删除、修改、遍历等
 - 字典操作：添加、删除、修改、遍历等
2. 因为Python中的变量在内存中实际上是一个指针，指向存储变量的内存地址。因此，我们可以将变量看作是一个标签，用于标识内存中的某个位置。
3. 提高Python代码的可读性可以通过以下方法：使用有意义的变量名 使用清晰的函数名和参数名 使用简洁明了的注释 使用 consistent 代码风格 使用清晰的代码结构

实验总结：

使用VSCode编写和运行Python程序：

1. 安装VSCode和Python：

- 首先，你需要在你的计算机上安装Visual Studio Code (VSCode)和Python。

2. 打开VSCode：

- 启动VSCode应用程序。

3. 创建新的Python文件：

- 在VSCode中，点击左侧的文件夹图标，选择一个文件夹作为你的工作目录。
- 右键点击该目录，选择“New File”创建一个新文件，将其命名为以 `.py` 结尾的文件，例如 `main.py`。

4. 编写Python代码：

- 在新创建的Python文件中，开始编写Python代码。

5. 保存文件：

- 定期保存你的代码，可以使用快捷键 `Ctrl + S` 或者选择菜单中的 "File" -> "Save"。

6. 运行Python程序：

- 在VSCode的终端中输入 `python 文件名.py` (例如 `python main.py`) 来运行你的Python程序。

7. 调试程序：

- 使用VSCode提供的调试工具，可以逐步执行代码、设置断点等，帮助你找出程序中的错误。

学习Python变量和简单数据类型：

1. 变量：

- 学习如何声明变量，并了解不同类型的变量（整数、浮点数、字符串等）。

2. 简单数据类型：

- 学习Python的基本数据类型，包括整数、浮点数、字符串、布尔值等。

3. 数据结构：

- 掌握Python中常用的数据结构，如列表、元组、集合、字典等，以及它们的使用方法。

4. 程序语言的语法：

- 学习Python的基本语法，包括条件语句、循环、函数等。

5. 算法：

- 开始解决简单的问题，并学习如何设计和实现算法以解决复杂问题。

6. 编程技巧：

- 学习一些常用的编程技巧，如错误处理、模块化、代码复用等。

7. 编程思想：

- 开始理解和应用一些基本的编程思想，如面向对象编程、函数式编程等。