



AI Development Fundamentals

使用 **Python** 迈出第一步

Oct 2020

Microsoft Reactor | Ryan Chung

```
led by player to  
s.load_image("kg.png")  
  
[self]:  
    initialize Dog object and create Text o  
g, self).__init__(image = Dog.image  
x = games.mouse.x  
bottom = games.sc  
  
re = games.Text(value = 0, size = 24  
top = 5, right = gam  
  
reen.add(self.score)  
1 = games.Text(value = 0, size = 24  
top = 5, left = gam
```



Ryan Chung

Instructor / DevelopIntelligence
Founder / MobileDev.TW

@ryanchung403 on WeChat
Ryan@MobileDev.TW





Reactor



developer.microsoft.com/reactor/
@MSFTReactor on Twitter



使用 Python 迈出第一步

4 小时 18 分钟 剩余 • 学习路径 • 已完成 1 个模块，共 10 个

初级

开发人员

Visual Studio Code

想学习编程语言，但不知从何处着手？从此处开始！了解使用 Python 构建简单应用程序所需的基本语法和预期过程。

在此学习路径下，你将：

- 编写第一批 Python 代码
- 存储和操作数据以修改其类型和外观
- 执行代码库中提供的内置功能
- 向代码添加逻辑以实现复杂的业务功能

完成此学习路径后，你将为后续的 Python 学习路径提供坚实的基础。

先决条件

无

开始 >

🔖 书签

➕ 添加到集合

<https://aka.ms/FirstStepsWithPython-8>

流程

什么是 Python ?

使用 VS Code 设置开发
环境

创建你的第一个 Python
程序

if...elif...else

字符串操作和设置格式

对数值数据执行运算

导入标准库模块

while 循环访问代码块

列表管理数据序列

使用函数创建可重用功能

本机开发环境设置

- 安装Python

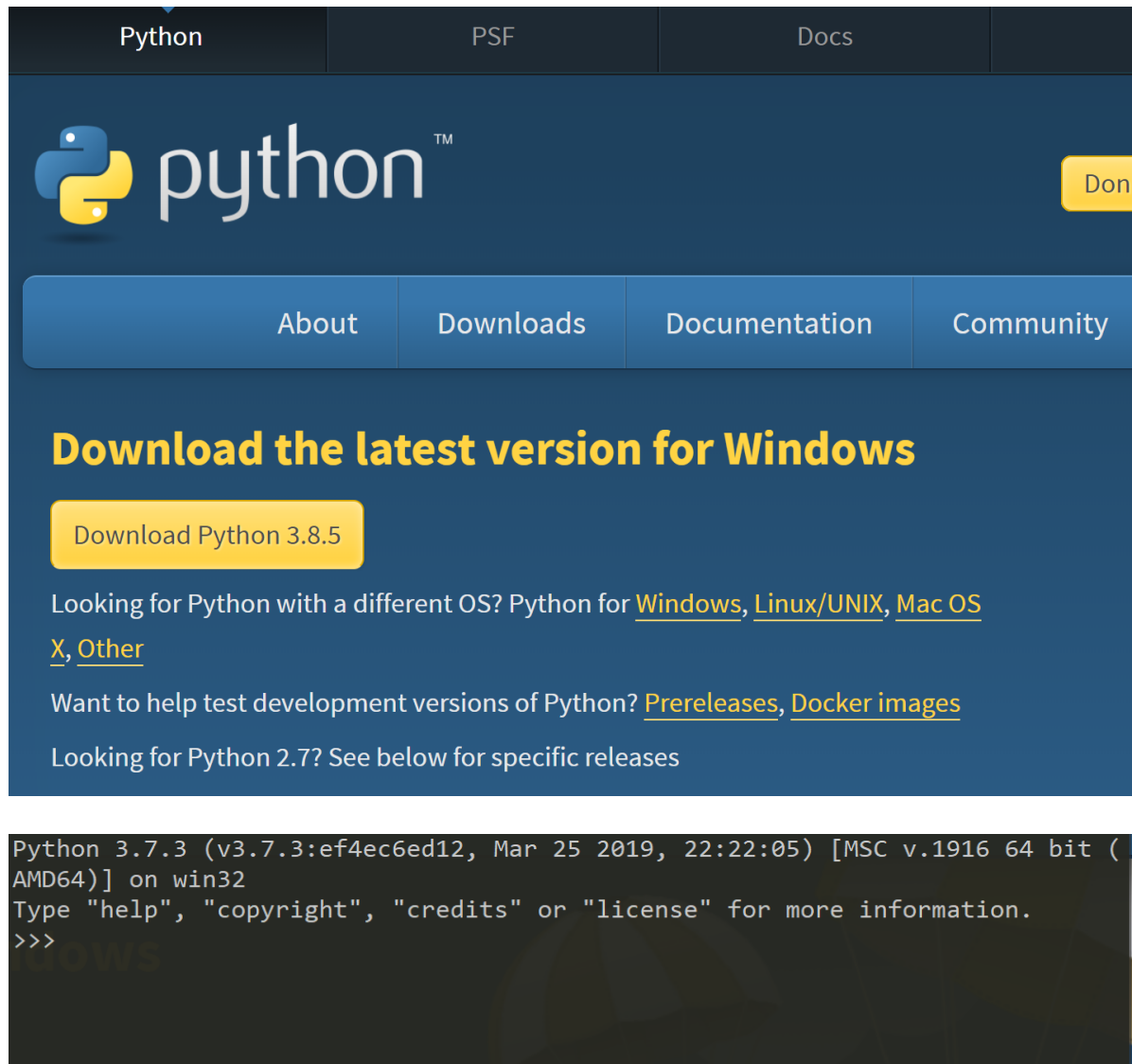
- 下载对应操作系统的版本

- 确认安装

- 开启命令提示字符
 - 输入python
 - 如有出现执行环境代表确认安装完成
 - 按下Ctrl+Z退出

- 版本确认

- 开启命令提示字符
 - 输入py --version



知识检查

1. 哪种工具可让你暂停程序执行、单步调试每行 Python 代码以观察其运行情况以及观察代码对存储在 RAM 中的值所做的更改？

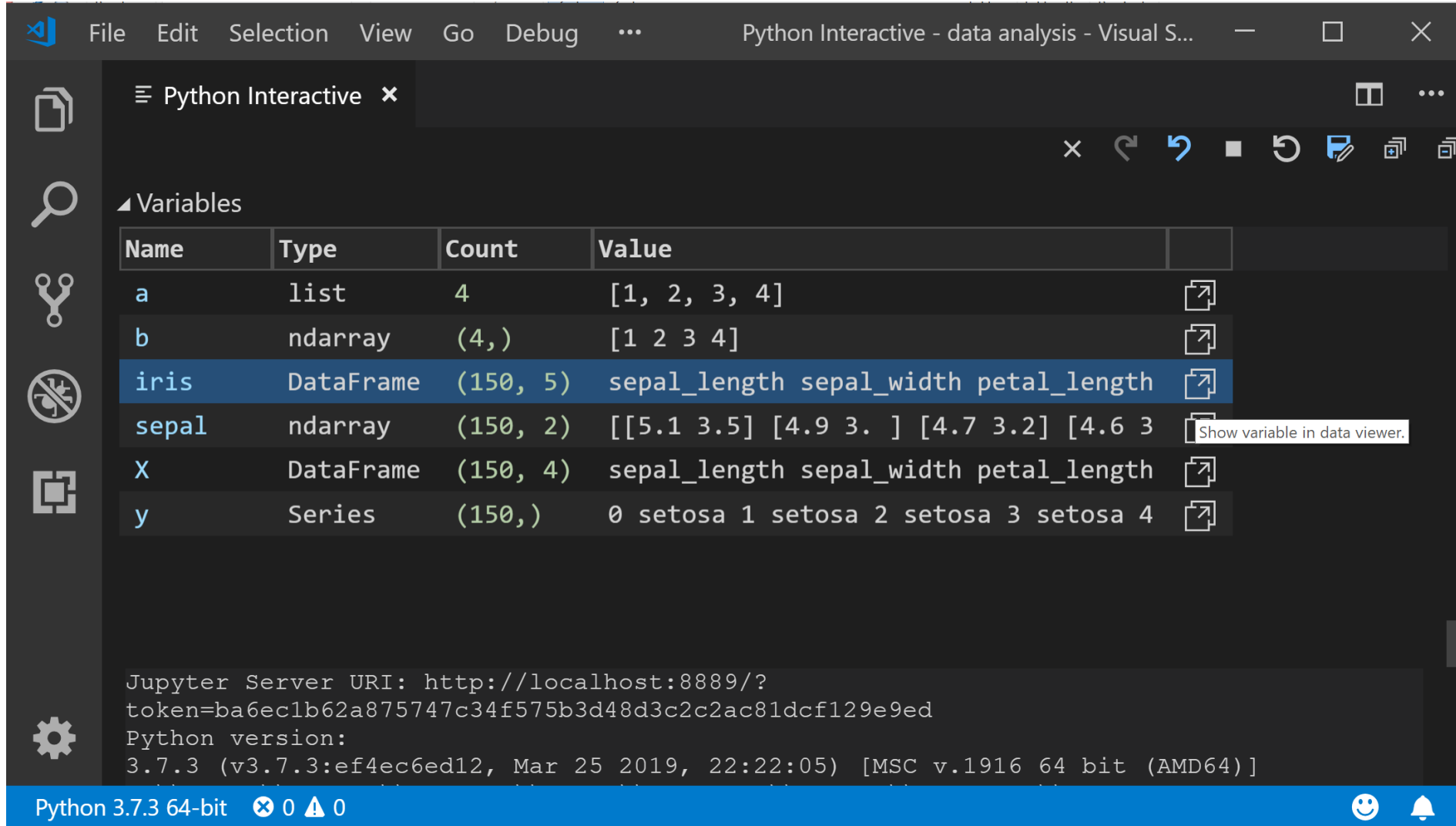
- ☐ 编译器
- ☐ 解释器
- ☐ 编辑器
- ☐ 调试程序

2. 以下哪个步骤_不是_软件开发过程的一部分？

- ☐ 软件供应商获取
- ☐ 测试代码
- ☐ 需求分析
- ☐ 编写代码

本机开发环境选择

• Microsoft Visual Studio Code



<https://code.visualstudio.com/>

安装扩充套件与设定

- 按下左边 Extensions图示或 Ctrl + Shift + X
 - Chinese (Simplified) Language Pack for Visual Studio Code
 - Python
 - Pylance
- 设定Ctrl+ 鼠标滚轴控制编辑器字号
 - editor.mouseWheelZoom
- 设定编辑时自动储存
 - 档案 -> 自动储存



Chinese (Simplified) Language Pack for Visual Studio Code

Microsoft | 4,835,250 | ★★★★★ | 儲存庫

Language pack extension for Chinese (Simplified)

安裝



Python ms-python.python

Microsoft | 23,279,154 | ★★★★★ | 儲存庫 | 授權 | v2020.7.96456

Linting, Debugging (multi-threaded, remote), Intellisense, Jupyter Notebooks, code formatting, refactoring, unit tests, snippets,...

停用 ▼ 解除安裝 已全域啟用此延伸模組。



Pylance ms-python.vscode-pylance 预览

Microsoft | 194,470 | ★★★★★☆ | 儲存庫 | 授權 | v2020.9.8

A performant, feature-rich language server for Python in VS Code

停用 ▼ 解除安裝 已全域啟用此延伸模組。

≡ 設定 ×

editor.mouseWheelZoom

使用者 工作區

▼ 文字編輯器 (1)

Editor: Mouse Wheel Zoom

☒ 使用滑鼠滾輪並按住 **Ctrl** 時，縮放編輯器的字型

Overview 综览

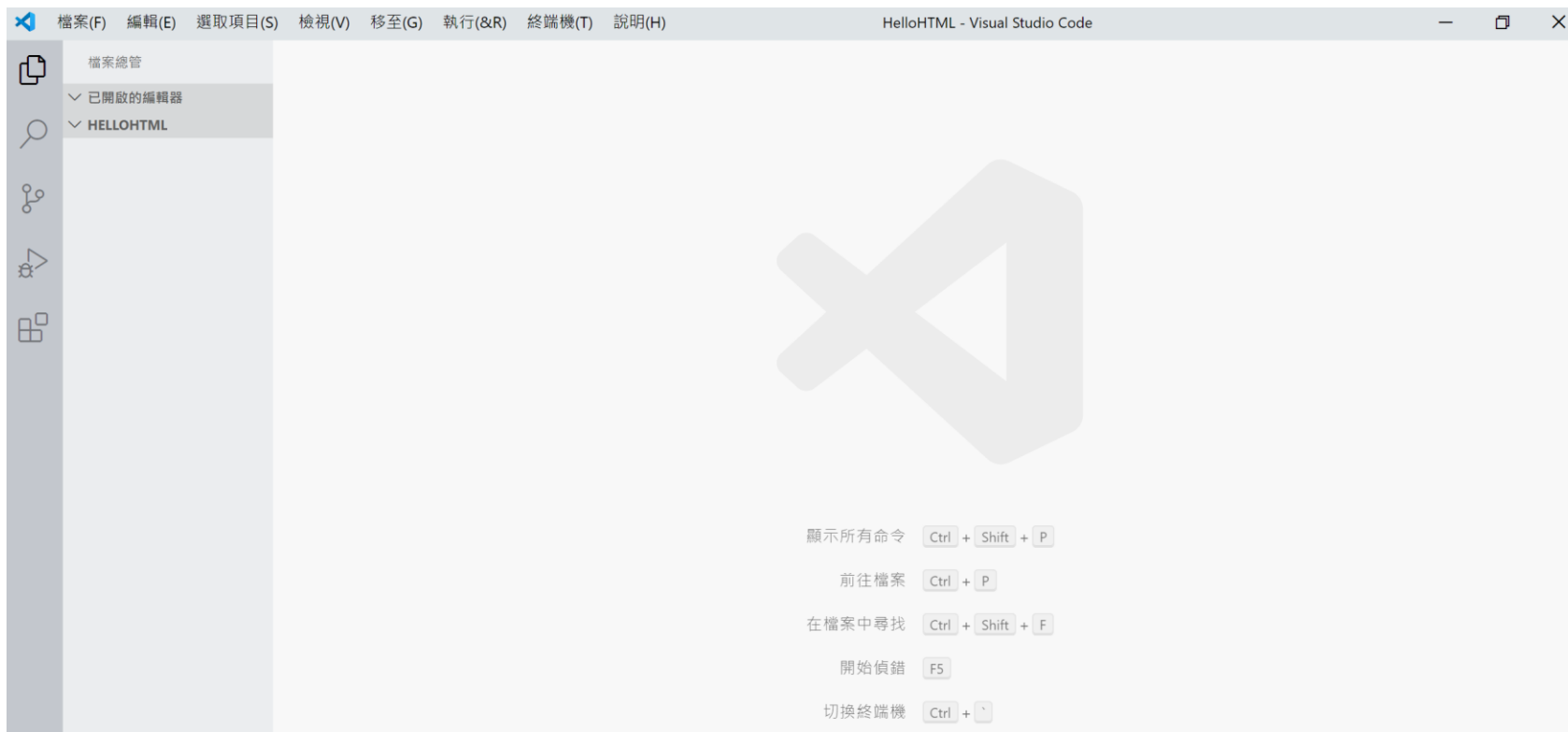
为什么使用Python?

- Easy to learn 容易学习
- Flexible 弹性大
- Powerful libraries 强大的函式库
- 处理数据的业务应用
- 动态 Web 应用
- 2D 和 3D 游戏
- 金融和科研应用
- 基于云的应用
- 移动应用

知名社群网路、影片平台以及搜寻引擎公司都大量使用**Python**在他们的核心技术，数据科学也是其中一个应用项目。

第一个 HelloWorld.py


- 在计算机中新增一个文件夹：HelloPython
- 打开VS Code, 档案(F) -> 开启文件夹...
- 选择刚才建立的文件夹
- 关闭「开始使用」分页




第一个 HelloWorld.py

- 左边档案总管区，按下鼠标右键 -> 新增档案
- 输入HelloWorld.py
- 档案前会出现Python图案




 Python Extension




Create a Jupyter Notebook

- Run "[Create New Blank Jupyter Notebook](#)" in the Command Palette (*Shift + Command + P*)
- Explore our [sample notebook](#) to learn about notebook features




Create a Python File

- Create a [new file](#) with a .py extension



Open a Folder or Workspace

- Open a [Folder](#)
- Open a [Workspace](#)

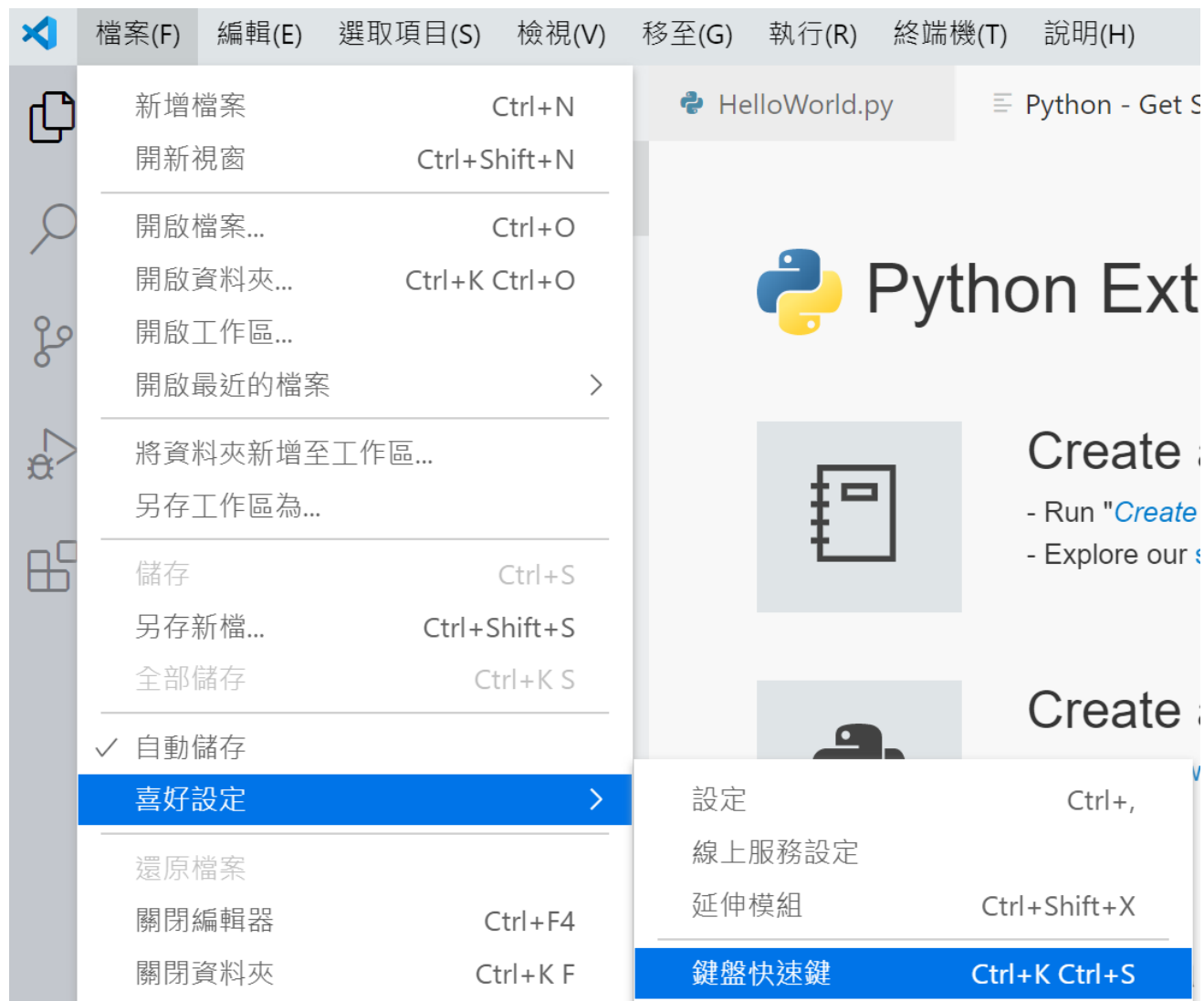


Use the Interactive Window to develop Python Scripts

- You can create cells on a Python file by typing "`%%`"
- Use "*Shift + Enter*" to run a cell, the output will be shown in the interactive window

設定執行快捷鍵

· 檔案 -> 喜好設定 -> 鍵盤快捷方式



设定执行快捷键

- 输入Python进行搜寻

Python			
命令	按键繫结关系	当	来源
Python: 在 Python 终端机中执行选定内容 / 行 Run Selection/Line in Python Terminal <code>python.execSelectionInTerminal</code>	Shift + Enter	editorTextFocus && !findInputFocussed && !python.datas...	预设
Python: Run Current Cell <code>python.datascience.runcurrentcell</code>	Ctrl + Enter	editorTextFocus && python.datascience.featureenabled &...	预设
Python: Run Current Cell And Advance <code>python.datascience.runcurrentcelladvance</code>	Shift + Enter	editorTextFocus && python.datascience.featureenabled &...	预设
Python: Run Current File in Python Interactive V Run Current File in Python Interactive Window <code>python.datascience.runFileInteractive</code>	Shift + Alt + Enter	—	使用者
Python: Run Selection/Line in Python Interactive Run Selection/Line in Python Interactive Window <code>python.datascience.execSelectionInteractive</code>	Shift + Enter	editorTextFocus && python.datascience.featureenabled &...	预设
<code>python.datascience.runcurrentcellandaddb...</code>	Alt + Enter	editorTextFocus && python.datascience.featureenabled &...	预设

知识检查

1. 交互模式的用途是什么？

- ☐ 构建整个应用程序。
- ☐ 调试代码。
- ☐ 测试应用程序。
- ☐ 一次运行一行代码。

错误提示

```
Print("Hello")
```

Print: Any

**"Print" is not defined Pylance
(reportUndefinedVariable)**

瞄孔問題 (Alt+F8) 沒有可用的快速修正

练习：提示使用者输入姓名

```
#print("Hello")  
print("What is your name?")  
name = input()  
print("Hello, "+name)
```

Ryan

按 'Enter' 键确认您的输入或按 'Esc' 键取消

```
[1] ▶ #print("Hello")...
```



```
What is your name?  
Hello, Ryan
```

练习：两数相加

```
first_number, second_number = input("Please input 2 numbers:").split()  
print(first_number+second_number)
```

1 2

Please input 2 numbers: (按 'Enter' 键确认或按 'Esc' 键取消)

```
[1] ▶ first_number, second_number = input("Please input 2  
numbers:").split()...
```



12



原来我们把字符串相加了

```
first_number, second_number = input("Please input 2 numbers:").split()  
print(type(first_number))  
print(first_number+second_number)
```

1 2

Please input 2 numbers: (按 'Enter' 键确认或按 'Esc' 键取消)

```
[4] ▶ first_number, second_number = input("Please input 2  
numbers:").split()...
```



<class 'str'>

12

转换为整数资料型态，再相加

```
first_number, second_number = input("Please input 2 numbers:").split()  
print(int(first_number)+int(second_number))
```

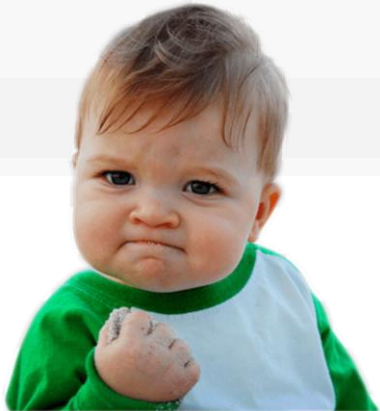
1 2

Please input 2 numbers: (按 'Enter' 键确认或按 'Esc' 键取消)

```
[5] ▶ first_number, second_number = input("Please input 2  
numbers:").split()...
```



3



输出内容加上说明

```
first_number, second_number = input("Please input 2 numbers:").split()  
print("The sum is "+str(int(first_number)+int(second_number)))
```

1 2

Please input 2 numbers: (按 'Enter' 键确认或按 'Esc' 键取消)

```
[6] ▶ first_number, second_number = input("Please input 2 numbers:").split()...
```



The sum is 3

练习

- 请使用者输入：
 - 日期
 - 早餐、午餐、晚餐、点心个别摄取的卡路里
- 系统输出：
 - XX日期的卡路里总和是YY

练习

```
this_date = input("请输入今天的日期:")  
this_breakfast, this_lunch, this_dinner, this_snack = input("请输入早餐/午餐/晚餐/点心的卡路里，并用,号隔开").split(",")  
print(this_date+"的卡路里总和是"+str(int(this_breakfast)+int(this_lunch)+int(this_dinner)+int(this_snack)))
```

10/18

请输入今天的日期: (按 'Enter' 键确认或按 'Esc' 键取消)

100,200,300,400

请输入早餐/午餐/晚餐/点心的卡路里，并用,号隔开 (按 'Enter' 键确认或按 'Esc' 键取消)

```
[3] ▶ this_date = input("请输入今天的日期:")...
```



10/18的卡路里总和是1000

知识检查

1. 哪项陈述是错误的？

- ☐ 默认情况下，所有变量都是字符串数据类型。
- ☐ `print()` 函数仅适用于整数和字符串数据类型。
- ☐ 如果为加法和串联使用相同的符号，Python 将根据两个值的数据类型推断应执行的操作。
- ☐ 在执行字符串串联前，必须将整数值转换为字符串值。

2. 你想要创建新的变量用于保存现有库存项目的计数。以下哪个变量名称会是你的最佳选择？

- ☐ `inventory_count_on_hand`
- ☐ `inventoryCountOnHand`
- ☐ `inventory_count`
- ☐ `inv-ct`

3. 函数名称旁边的左括号和右括号有何用途？

- ☐ 它们定义运算的顺序，就像在数学中一样。
- ☐ 括号是指示 Python 执行函数的函数调用运算符。
- ☐ 它们指示 Python 返回一个值。
- ☐ 它们指示 Python 在输出中显示内容。

if...elif...else

- 兵分多路
- ==判断
- 判断句后面加冒号
- 缩排与区段

exercise.py

```
value = 7
```

```
if value == 7:  
    print('The value is 7')
```

```
print('Finished!')
```

```
[1] ▶ value = 7...
```



```
The value is 7  
Finished
```

```
value = 6
```

```
if value == 7:  
    print('The value is 7')
```

```
print('Finished!')
```

```
[1] ▶ value = 6...
```



```
Finished
```

缩排是有意义的!

```
value = 6

if value == 7:
    print('The value is 7')

    print('Finished!')
```

```
value = 6

if value == 7:
    print('The value is 7')

    print('Finished!')
```

Unindent amount does not match previous indent

```
[2] ▶ value = 6...
```



加上else

```
value = 6
```

```
if value == 7:  
    print('The value is 7')  
else:  
    print('The value is not 7')
```

```
print('Finished!')
```

```
[1] ▶ value = 6...
```



```
The value is not 7  
Finished!
```


再加上elif (Else If)

```
value = 6

if value == 7:
    print('The value is 7')
elif value == 6:
    print('The value is 6')
else:
    print('The value is not 7 or 6')

print('Finished!')
```

```
[2] ▶ value = 6...
```



```
The value is 6
Finished!
```

elif 的重迭谬误

```
value = 6
```

```
if value < 8:  
    print('The value is less than 8')  
elif value < 7:  
    print('The value is less than 7')  
else:  
    print('The value is greater than 8')
```

```
[2] ▶ value = 6...
```



```
The value is less than 8
```

避免范围重迭

```
value = 58
```

```
if value >=80:  
    print('Score A')  
elif value >=70:  
    print('Score B')  
elif value >=60:  
    print('Score C')  
else:  
    print('Failed')
```

```
[1] ▶ value = 58...
```



Failed

资料型态

```
this_string = "Hello World"  
this_bool = True  
this_Int = 5  
this_float = 2.5
```

Variables

Name	Type	Size	Value
this_Int	int		5
this_bool	bool		True
this_float	float		2.5
this_string	str	11	Hello World

```
[2] type(this_string)
```



str

```
[3] type(this_bool)
```



bool

```
[4] type(this_Int)
```



int

```
[5] type(this_float)
```



float

练习

- 请使用者输入确认是否要继续(Y/N)
 - Yes : 显示继续
 - No : 显示结束
 - 其他输入 : 显示请重新输入

练习

- 请使用者输入确认是否要继续(Y/N)
 - Yes : 显示继续
 - No : 显示结束
 - 其他输入 : 显示请重新输入

```
user_input = input("Would you like to continue?(Y/N)")

if user_input.lower()=="yes" or user_input.lower()=="y":
    print("Continuing ...")
    print("Complete!")
elif user_input.lower()=="no" or user_input.lower()=="n":
    print("Exiting")
else:
    print("Please try again and respond with yes or no.")
```

知识检查

1. 代码行 `if x == 3` 有什么问题？

- ☐ 它使用 `==` 而不是 `=` 来检查是否相等。
- ☐ 末尾缺少关键字 `then`。
- ☐ 布尔表达式两边缺少括号。
- ☐ 末尾缺少 `:`。

2. 当 `x = 6` 时，以下布尔表达式的结果是什么： `x > 3 or x < 5`？

- ☐ `True`
- ☐ `False`
- ☐ 布尔表达式的语法不正确。

3. 当 `x = 6` 时，以下布尔表达式的结果是什么： `not x`？

- ☐ `True`
- ☐ `False`
- ☐ 布尔表达式的语法不正确。

字符串练习

```
first_string = '用单引号的字符串'  
second_string = "用双引号的字符串"  
third_string = "有'单引号'的字符串"
```

Variables

Name	Type	Size	Value
first_string	str	7	用單引號的字串
second_string	str	7	用雙引號的字串
third_string	str	9	有'單引號'的字串

字符串练习

```
first_string = '用单引号的字符串'  
second_string = "用双引号的字符串"  
third_string = "有'单引号'的字符串"
```

```
[5] first_string[0]
```



'用'

```
[6] second_string[0]
```



'用'

```
[7] first_string[0] == second_string[0]
```



True

字符串练习

```
first_string = '用单引号的字符串'  
second_string = "用双引号的字符串"  
third_string = "有'单引号'的字符串"  
#也可以这样写  
fourth_string = '有\'单引号\'的字符串'
```

Variables

	Name	Type	Size	Value
	first_string	str	7	用單引號的字符串
	fourth_string	str	9	有'單引號'的字符串
	second_string	str	7	用雙引號的字符串
	third_string	str	9	有'單引號'的字符串

字符串练习

```
first_string = '用单引号的字符串'
```

```
second_string = "用双引号的字符串"
```

```
third_string = "有'单引号'的字符串"
```

```
#也可以这样写
```

```
fourth_string = '有\'单引号\'的字符串'
```

```
#很多行的这样写
```

```
fifth_string = '''可以
```

```
撰写
```

```
多行
```

```
'''
```

输出型式

```
first = "张三"  
second = "李四"  
third = "王五"  
print(first, second, third)
```

```
[1] ▶ first = "張三"...
```



張三 李四 王五

输出型式

```
first = "张三"  
second = "李四"  
third = "王五"  
print(first, second, third, sep="、")
```

```
[2] ▶ first = "張三"...
```



張三、李四、王五

输出型式

```
first = "张三"  
second = "李四"  
third = "王五"  
print(first, second, third, sep="、", end="。")
```

```
[3] ▶ first = "張三"...
```



張三、李四、王五。

大写、小写、首字大写

```
this_string = "heLLo wOrlD!"  
print(this_string)  
print(this_string.lower())  
print(this_string.upper())  
print(this_string.capitalize())
```

```
[1] ▶ this_string = "heLLo wOrlD!"...
```



```
heLLo wOrlD!  
hello world!  
HELLO WORLD!  
Hello world!
```

计算次数

```
repeat_string = "Yo Yo Yo"  
print(repeat_string.count("Yo"))
```

```
[2] print(repeat_string.count("Yo"))
```



3

计算长度

```
long_word = "interesting"  
print(len(long_word))
```

```
[3] ▶ long_word = "interesting"...
```



11

找位置

```
this_sentence = "Where is Ryan?"  
print(this_sentence.find("Ryan"))  
print(this_sentence.find("John"))
```

```
[5] print(this_sentence.find("Ryan"))
```



9

```
[6] print(this_sentence.find("John"))
```



-1

去除空格

```
have_spaces = "    There are some spaces    "  
print(have_spaces.lstrip()+".")  
print(have_spaces.rstrip()+".")  
print(have_spaces.strip()+".")
```

```
[13] print(have_spaces.lstrip()+".")
```



```
There are some spaces .
```

```
[14] print(have_spaces.rstrip()+".")
```



```
There are some spaces.
```

```
[15] print(have_spaces.strip()+".")
```



```
There are some spaces.
```

补满位置

```
want_stars = "Shining Baby"  
print(want_stars.rjust(20, "*"))  
print(want_stars.ljust(20, "*"))  
print(want_stars.center(20, "*"))
```

```
[20] print(want_stars.rjust(20, "*"))  
✖ *****Shining Baby  
[21] print(want_stars.ljust(20, "*"))  
✖ Shining Baby*****  
[22] print(want_stars.center(20, "*"))  
✖ ****Shining Baby****
```

格式化输出

```
medicine = '感冒药'  
count = 5  
duration = 4
```

```
instructions = '{} - 每{}小时吃{}颗'.format(medicine, duration, count)
```

```
instructions2 = '{0} - 每{2}小时吃{1}颗'.format(medicine, duration, count)
```

```
instructions3 = '{m} - 每{d}小时吃{c}颗'.format(m=medicine, d=duration, c=count)
```

[3] instructions



'感冒藥 - 每4小時吃5顆'

[6] instructions2



'感冒藥 - 每5小時吃4顆'

[16] instructions3



'感冒藥 - 每4小時吃5顆'

格式化输出 - 更直观的写法

```
medicine = '感冒药'  
count = 5  
duration = 4
```

```
instructions = '{} - 每{}小时吃{}颗'.format(medicine, duration, count)
```

```
instructions2 = '{0} - 每{2}小时吃{1}颗'.format(medicine, duration, count)
```

```
instructions3 = '{m} - 每{d}小时吃{c}颗'.format(m=medicine, d=duration, c=count)
```

```
instructions4 = f"{medicine} - 每{duration}小时吃{count}颗"
```

```
[18] instructions4
```



```
'感冒藥 - 每4小時吃5顆'
```

知识检查

1. 需要向文本字符串添加单引号。 以下哪个选项不起作用？

- ☐ 创建逐字字符串并在需要时添加单引号。
- ☐ 创建典型双引号字符串并在需要时添加单引号。
- ☐ 创建典型字符串，并在需要时使用转义序列（反斜杠和单引号）。
- ☐ 创建典型单引号字符串，并添加三个单引号以创建转义序列。

2. 如何确定包含字符串的变量 `my_value` 中的字符数？

- ☐ `len(my_value)`
- ☐ `my_value.len()`
- ☐ `count(my_value)`
- ☐ `my_value.count()`

检查资料形态

`user_name = "王小明"`

`user_age = 25`

`user_weight = 65.3`

Variables

	Name	Type	Size	Value
	user_age	int		25
	user_name	str	3	王小明
	user_weight	float		65.3

```
[2] type(user_name)
```



str

```
[3] type(user_age)
```



int

```
[4] type(user_weight)
```



float

判断资料型態

`user_name = "王小明"`

`user_age = 25`

`user_weight = 65.3`

```
[5] isinstance(user_name, str)
```



True

```
[6] isinstance(user_age, float)
```



False

```
[7] isinstance(user_weight, int)
```



False

判断该字符串是否可以转换成数字

```
input_age = input("请输入你的年纪:")
```

```
if input_age.isnumeric():  
    print(str(2020-int(input_age))+"年生")  
else:  
    print("无法转换")
```

25

請輸入你的年紀: (按 'Enter' 鍵確認或按 'Esc' 鍵取消)

```
[1] ▶ input_age = input("請輸入你的年紀:")...
```



× 1995年生

ABC

請輸入你的年紀: (按 'Enter' 鍵確認或按 'Esc' 鍵取消)

```
[2] ▶ input_age = input("請輸入你的年紀:")...
```



× 無法轉換

练习：计算机

- 请提示使用者输入：
 - 第一个数字
 - 运算符(+ - * / % // **)
 - 第二个数字
- 如果使用者输入的两个均为数字，则进行运算并输出
- 若有非数字或非运算符的输入，则请使用者重新输入

练习：计算机

```
import operator
```

```
first_number = input("请输入第一个数字:")
input_operator = input("请输入运算(+-*//**")
second_number = input("请输入第二个数字")
ops = {
    "+":operator.add,
    "-":operator.sub,
    "*":operator.mul,
    "/":operator.truediv,
    "%":operator.mod,
    "//":operator.floordiv,
    "**":operator.pow
}
```

```
if first_number.isnumeric() and second_number.isnumeric():
    if input_operator in ops.keys():
        print(f"{first_number}{input_operator}{second_number}={ops[input_operator](int(first_number),int(second_number))}")
    else:
        print("运算符号有误")
else:
    print("输入的数字有误!")
```

operator.add(a, b)

Return a + b, for a and b numbers.

知识检查

1. 如果要评估值是否为字符串类型，应使用哪个函数？

- ☐ `isstring()` 函数
- ☐ `type()` 函数
- ☐ `isinstance()` 函数
- ☐ `isnumeric()` 函数

2. 你想确定一个数字能否被另一个数字整除。应使用哪种运算符？

- ☐ `/` 运算符
- ☐ `^` 运算符
- ☐ `&` 运算符
- ☐ `%` 运算符

3. 假设名为 `interest_rate` 的变量中有一个值 `1.468`，你想将其舍入到 `1.5`。该如何调用 `round()` 方法？

- ☐ `round(interest_rate)`
- ☐ `round(interest_rate, 0)`
- ☐ `round(interest_rate, 1)`
- ☐ `round(interest_rate, 2)`

汇入模組

```
import random  
print("一到十随机来一个:"+str(random.randint(1,10)))
```

```
[2] ▶ import random...
```



× 一到十隨機來一個:9

```
[3] ▶ import random...
```



× 一到十隨機來一個:2

```
[4] ▶ import random...
```



× 一到十隨機來一個:5

```
[5] ▶ import random...
```



× 一到十隨機來一個:3

```
[6] ▶ import random...
```



× 一到十隨機來一個:2

匯入模組 – 使用別名

```
import random as rd  
print("一到十隨機來一個:" + str(rd.randint(1,10)))
```

```
[2] ▶ import random...
```



一到十隨機來一個:9

```
[3] ▶ import random...
```



一到十隨機來一個:2

```
[4] ▶ import random...
```



一到十隨機來一個:5

```
[5] ▶ import random...
```



一到十隨機來一個:3

```
[6] ▶ import random...
```



一到十隨機來一個:2

安装套件：pip install

- 开启终端机

```
pip install emoji
```

- 安装完成后，回到程式码，测试安装的套件

```
import emoji  
print(emoji.emojize('Howdy :sun_with_face:'))
```

```
[2] print(emoji.emojize('Howdy :sun_with_face:'))
```



Howdy 🌞

知识检查

1. 什么是模块？

- ☐ 在本地计算机上安装的代码文件包。
- ☐ 代码使用的依赖项的列表。
- ☐ 只能由 Python 开发团队创建的代码文件。
- ☐ 包含可从自己的代码引用和调用的函数的代码文件。

2. `pip` 实用工具的作用是什么？

- ☐ `pip` 实用工具将从 GitHub 安装包。
- ☐ `pip` 实用工具安装在 Python 包索引 (PyPI) 中列出的包及其依赖项。
- ☐ `pip` 实用工具将帮助你查找用于解决编程问题的包。
- ☐ `pip` 实用工具将从 Python 标准库安装包。

while 一直做

- 从1 ~ 1000 抽号码，直至抽到666为止

```
import random
```

```
roll = 0
```

```
count = 0
```

```
while roll!=666:
```

```
    count = count + 1
```

```
    roll = random.randint(1,1000)
```

```
print(f'从1~1000里面抽到666，我抽了{count}次')
```

```
[1] ▶ import random...
```



從1~1000裡面抽到666，我抽了157次

```
[2] ▶ import random...
```



從1~1000裡面抽到666，我抽了11次

```
[3] ▶ import random...
```



從1~1000裡面抽到666，我抽了2406次

练习：谁去倒垃圾

- 使用者可输入一个个姓名，输入完毕按下q
- 输入空白会被忽略、输入q结束输入
- 结束后从使用者输入的名子中乱数选择一个

張三

請輸入姓名，停止請按q: (按 'Enter' 鍵確認或按 'Esc' 鍵取消)

李四

請輸入姓名，停止請按q: (按 'Enter' 鍵確認或按 'Esc' 鍵取消)

王五

請輸入姓名，停止請按q: (按 'Enter' 鍵確認或按 'Esc' 鍵取消)

```
[2] ▶ from random import randint...
```



今天王五去倒垃圾

练习：谁去倒垃圾

```
from random import randint
```

```
people = []  
name = ""
```

```
while name != 'q':  
    name = input("请输入姓名，停止请按q:")  
    if name.strip() == '':  
        continue  
    if name.strip() == 'q':  
        break  
    people.append(name)
```

```
print(f'今天{people[randint(0, len(people)-1)]}去倒垃圾')
```

练习：猜数字

- 系统随机产生1~5之间的一个数字
- 使用者输入猜测，直至猜中
- 系统回应猜测次数与正确答案

练习：猜数字

```
from random import randint
```

```
answer = randint(1,5)
```

```
user_input = ""
```

```
count = 0
```

```
while user_input != str(answer):
```

```
    user_input = input("请输入一个1~5的数字:")
```

```
    count+=1
```

```
print(f'答对了，是{answer}，你猜了{count}次')
```

练习：猜数字进阶版

- 系统随机产生1~5之间的一个数字
- 使用者输入猜测，直至猜中
- 使用者若输入非数字，提示使用者不要乱打(不计次)
- 使用者输入值比正确答案大或小，也会给予提示
- 答对时，系统回应猜测次数与正确答案

练习：猜数字进阶版

```
from random import randint
```

```
answer = randint(1,5)
```

```
user_input = ""
```

```
count = 0
```

```
while user_input!=str(answer):
```

```
    user_input = input("请输入一个1~5的数字:")
```

```
    if user_input.isnumeric()!=True:
```

```
        print("不要乱打")
```

```
        continue
```

```
    elif int(user_input)>answer:
```

```
        print("太大了")
```

```
    elif int(user_input)<answer:
```

```
        print("太小了")
```

```
    else:
```

```
        pass
```

```
    count+=1
```

```
print(f'答对了，是{answer}，你猜了{count}次')
```

while-else 寻人好帮手

```
this_group = ["张三", "李四", "王五"]
i = 0
while i <= len(this_group) - 1:
    if this_group[i] == "张三":
        print(f'张三在第{i}个房间')
        break
    i += 1
else:
    print("张三没在这")
```

break之后，**else**这一段就没有执行了!

```
[7] ▶ this_group = ["张三", "李四", "王五"]...
```



张三在第0个房间

while-else 寻人好帮手

```
this_group = ["张三", "李四", "王五"]
i = 0
while i <= len(this_group) - 1:
    if this_group[i] == "柳六":
        print(f'柳六在第{i}个房间')
        break
    i += 1
else:
    print("柳六没在这")
```

while整个跑完，没有**break**，就进入了**else**

```
[8] ▶ this_group = ["张三", "李四", "王五"]...
```



柳六沒在這

知识检查

1. 何时使用 `while` 语句？

- ☐ 需要循环访问代码块，直到条件不再为 `True` 时。
- ☐ 需要循环访问代码块，直到条件变为 `True` 时。
- ☐ 需要对代码进行分支时。
- ☐ 需要调用子例程时。

2. 应使用哪个语句处理 `while` 布尔表达式不再为 `True` 的情况？

- ☐ `break` 语句。
- ☐ `else` 语句。
- ☐ `continue` 语句。
- ☐ `goto`

3. 以下哪个赋值运算符将执行乘法和赋值运算？

- ☐ `x=`
- ☐ `^=`
- ☐ `$=`
- ☐ `*=`

List 操作

	0	1	2	3	4	5	6
rainbow =	'红'	'橙'	'黄'	'绿'	'蓝'	'靛'	'紫'
	-7	-6	-5	-4	-3	-2	-1

```
[2] rainbow[1]
```



'橙'

```
[3] rainbow[-2]
```



'靛'

```
[4] rainbow[7]
```



IndexError: list index out of range

```
[5] rainbow[2:5]
```



['黄', '绿', '蓝']

```
[6] rainbow[-4:-2]
```



['绿', '蓝']

[开始位置, 结束位置-不包含]

List 操作

- pop

- 指定位置
- 不指定位置(最后面)

```
[8] rainbow.pop(0)
```



```
'紅'
```

```
[9] rainbow
```



```
['澄', '黃', '綠', '藍', '碇', '紫']
```

```
[10] rainbow.pop()
```



```
'紫'
```

```
[11] rainbow
```



```
['澄', '黃', '綠', '藍', '碇']
```

List 操作

- append
 - 加至最后面
- insert
 - 指定位置

```
[12] rainbow.append('紫')
```



```
[13] rainbow
```



```
['澄', '黃', '綠', '藍', '靛', '紫']
```

```
[14] rainbow.insert(0, '紅')
```



```
[15] rainbow
```



```
['紅', '澄', '黃', '綠', '藍', '靛', '紫']
```

List 操作

- remove
 - 指定元素刪除

```
[16] rainbow.append('青')
```



```
[17] rainbow
```



```
['紅', '澄', '黃', '綠', '藍', '碇', '紫', '青']
```

```
[18] rainbow.remove('青')
```



```
[19] rainbow
```



```
['紅', '澄', '黃', '綠', '藍', '碇', '紫']
```

List 操作

- extend
- 合并

```
rainbow1 = ['红', '橙', '黄', '绿']  
rainbow2 = ['蓝', '靛', '紫']
```

```
[20] rainbow1 = ['紅', '橙', '黃', '綠']
```



```
[21] rainbow2 = ['藍', '靛', '紫']
```



```
[22] rainbow1.extend(rainbow2)
```



```
[23] rainbow1
```



```
['紅', '橙', '黃', '綠', '藍', '靛', '紫']
```

List 操作

- clear

- 清空

```
rainbow1 = ['红', '橙', '黄', '绿']  
rainbow2 = ['蓝', '靛', '紫']
```

```
[24] rainbow2
```



```
['藍', '靛', '紫']
```

```
[25] rainbow2.clear()
```



```
[26] rainbow2
```



```
[]
```


Membership testing 是否在群里

dataType.py > ...

```
1 roomGuest = ["王明", "柳宇", "陳尚"]
2 roomKey = (1111, 2222, 3333)
3 breakfastChoice = {"中式", "西式", "法式"}
4 guestBreakfast = {
5     "王明": "中式",
6     "柳宇": "法式",
7     "陳尚": "中式"
8 }
9 print("王明有住房嗎? " + str("王明" in roomGuest))
10 print("陳尚應該沒來住房吧? " + str("陳尚" not in roomGuest))
```

Python Interactive X

X ↶ ↷ □ ↻ 📄 📄 📄

[11] ▶ roomGuest = ["王明", "柳宇", "陳尚"]...



X 王明有住房嗎? True
陳尚應該沒來住房吧? False

练习：扑克牌

- 用回圈将所有牌印出

紅心A
紅心2
紅心3
紅心4
紅心5
紅心6
紅心7
紅心8
紅心9
紅心10
紅心J
紅心Q
紅心K

黑桃A
黑桃2
黑桃3
黑桃4
黑桃5
黑桃6
黑桃7
黑桃8
黑桃9
黑桃10
黑桃J
黑桃Q
黑桃K

方塊A
方塊2
方塊3
方塊4
方塊5
方塊6
方塊7
方塊8
方塊9
方塊10
方塊J
方塊Q
方塊K

鑽石A
鑽石2
鑽石3
鑽石4
鑽石5
鑽石6
鑽石7
鑽石8
鑽石9
鑽石10
鑽石J
鑽石Q
鑽石K

练习：扑克牌

- 用回圈将所有牌印出

```
suits = ["红心", "黑桃", "方块", "钻石"]
ranks = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']

for this_suit in suits:
    for this_rank in ranks:
        print(f'{this_suit}{this_rank}')
```

练习：扑克牌

- 随机发一张牌

```
suits = ["红心", "黑桃", "方块", "钻石"]
```

```
ranks = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
```

方块K

紅心10

方块J

方块5

练习：扑克牌

- 随机发一张牌

```
suits = ["红心", "黑桃", "方块", "钻石"]  
ranks = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
```

方块K

紅心10

方块J

方块5

```
import random  
#发一张牌  
print(f'{random.choice(suits)}{random.choice(ranks)}')
```

练习：扑克牌发5张牌

- 先产生一副牌52张，然后随机发5张牌
- 显示目前剩余牌数
- 再执行发牌程序，可以再发5张牌(与前面不重复)
- 再显示目前剩余牌数

练习：扑克牌发5张牌

```
import random
```

```
#产生一副牌
```

```
suits = ["红心", "黑桃", "方块", "钻石"]
```

```
ranks = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
```

```
pokers = []
```

```
for this_suit in suits:
```

```
    for this_rank in ranks:
```

```
        pokers.append(f'{this_suit}{this_rank}')
```

```
print(f'现在有{len(pokers)}张牌')
```

```
#发牌程序
```

```
count = 0
```

```
pick_pokers = []
```

```
while count<5:
```

```
    count+=1
```

```
    this_pick = random.choice(pokers)
```

```
    pick_pokers.append(this_pick)
```

```
    pokers.remove(this_pick)
```

```
print(f'抽到这五张牌:{pick_pokers}')
```

```
print(f'现在还剩下{len(pokers)}张牌')
```

知识检查

1. 你有一个包含五个项的列表，要检索最后三个项。应该使用哪个切片？

- ☐ `my_list[1:4]`
- ☐ `my_list[:2]`
- ☐ `my_list[:-3]`
- ☐ `my_list[-3:]`

2. 可以使用以下哪个帮助程序函数使列表模拟队列？

- ☐ `remove()`
- ☐ `pop()`
- ☐ `del()`
- ☐ `clear()`

3. 了解值是否属于列表的最简单方法是什么？

- ☐ 使用 `find()` 帮助程序函数。
- ☐ 使用 `in` 运算符。
- ☐ 使用 `is` 运算符。
- ☐ 使用 `contains()` 帮助程序函数。

函数/函式 Function

- 关键词def
- 注意缩排、冒号

```
def functionName([parameters]):  
    statements  
    [return | return value]  
    [statements]
```

练习

- 修改前一个例子，定义出两个函数

- 产生一副牌

- 发5张牌

```
[2] poker_generator()
```



× 現在有52張牌

```
[3] get_five_cards()
```



× 抽到這五張牌: ['紅心10', '鑽石6', '紅心K', '鑽石Q', '方塊3']
現在還剩下47張牌

```
[4] get_five_cards()
```



× 抽到這五張牌: ['黑桃2', '鑽石5', '鑽石8', '紅心9', '方塊J']
現在還剩下42張牌

```
[5] get_five_cards()
```



× 抽到這五張牌: ['紅心5', '紅心2', '黑桃7', '鑽石7', '紅心A']
現在還剩下37張牌

练习

```
import random
```

```
pokers = []
```

两个方法都有使用到，拉至外面

#产生一副牌

```
def poker_generator():
```

```
    suits = ["红心", "黑桃", "方块", "钻石"]
```

```
    ranks = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
```

```
    for this_suit in suits:
```

```
        for this_rank in ranks:
```

```
            pokers.append(f'{this_suit}{this_rank}')
```

```
print(f'现在有{len(pokers)}张牌')
```

#发牌程序

```
def get_five_cards():
```

```
    count = 0
```

```
    pick_pokers = []
```

```
    while count<5:
```

```
        count+=1
```

```
        this_pick = random.choice(pokers)
```

```
        pick_pokers.append(this_pick)
```

```
        pokers.remove(this_pick)
```

```
print(f'抽到这五张牌:{pick_pokers}')
```

```
print(f'现在还剩下{len(pokers)}张牌')
```

有输入参数的函数

- 修改前例，让随机选5张牌变成可以指定随机选X张牌

```
[7] get_n_cards(3)
```



抽到這3張牌: ['鑽石4', '方塊5', '方塊10']
現在還剩下34張牌

有输入参数的函数

- 修改前例，让随机选5张牌变成可以指定随机选X张牌

#发牌程序v2

```
def get_n_cards(n):  
    count = 0  
    pick_pokers = []  
    while count < n:  
        count += 1  
        this_pick = random.choice(pokers)  
        pick_pokers.append(this_pick)  
        pokers.remove(this_pick)  
  
    print(f'抽到这{n}张牌:{pick_pokers}')  
    print(f'现在还剩下{len(pokers)}张牌')
```

有输入参数的函数

- 修改前例，随机选X张牌，若不指定就是5张

[9] `get_n_cards()`



× 抽到這5張牌: ['方塊7', '鑽石10', '方塊A', '方塊9', '紅心6']
現在還剩下29張牌

[10] `get_n_cards(2)`



× 抽到這2張牌: ['紅心8', '方塊K']
現在還剩下27張牌

有输入参数的函数

- 修改前例，随机选X张牌，若不指定就是5张

```
def get_n_cards(n=5):  
    count = 0  
    pick_pokers = []  
    while count < n:  
        count += 1  
        this_pick = random.choice(pokers)  
        pick_pokers.append(this_pick)  
        pokers.remove(this_pick)  
  
    print(f'抽到这{n}张牌:{pick_pokers}')  
    print(f'现在还剩下{len(pokers)}张牌')
```

有输入多个参数的函数

- 修改前例，随机选X张牌，可以输入玩家姓名或不输入

[12] `get_n_cards()`



× 抽到這5張牌: ['黑桃3', '黑桃J', '鑽石J', '黑桃K', '黑桃6']
現在還剩下22張牌

[13] `get_n_cards(2)`



× 抽到這2張牌: ['黑桃Q', '方塊6']
現在還剩下20張牌

[14] `get_n_cards(user_name='Ryan')`



× Ryan抽到這5張牌: ['紅心7', '鑽石K', '方塊4', '方塊8', '黑桃9']
現在還剩下15張牌

[15] `get_n_cards(4, user_name='John')`



× John抽到這4張牌: ['黑桃4', '紅心4', '黑桃8', '紅心Q']
現在還剩下11張牌

有输入多个参数的函数

- 修改前例，随机选X张牌，可以输入玩家姓名或不输入

```
def get_n_cards(n=5, user_name=None):  
    count = 0  
    pick_pokers = []  
    while count < n:  
        count += 1  
        this_pick = random.choice(pokers)  
        pick_pokers.append(this_pick)  
        pokers.remove(this_pick)  
    if user_name == None:  
        print(f'抽到这{n}张牌:{pick_pokers}')    else:  
        print(f'{user_name}抽到这{n}张牌:{pick_pokers}')  
    print(f'现在还剩下{len(pokers)}张牌')
```

知识检查

1. 以下哪项陈述_不是_创建函数的充分理由?

- ☐ 需要将经常用到的功能封装在一个位置时。
- ☐ 在将较大的问题分解为较小的、更易于管理的部分方面需要帮助时。
- ☐ 需要通过将函数添加到模块中来跨一个或多个程序共享功能时。
- ☐ 需要存储将在整个程序中使用的信息时。

2. 如何使输入参数成为可选参数?

- ☐ 通过添加**必需**的关键字。
- ☐ 通过提供在调用方未传递参数的情况下应使用的默认值。
- ☐ 通过使用关键字参数。
- ☐ 通过检查其值查看它是否等于 `None`。

获得奖杯

<https://docs.microsoft.com/zh-cn/users/XXXXXX/achievements>



Microsoft

Docs

奖杯

概述

活动

挑战

书签

集合

关注

成就

设置



奖杯

使用 Python 迈出第一步



Reactor



developer.microsoft.com/reactor/
[@MSFTReactor](https://twitter.com/MSFTReactor) on Twitter

议程结束 感谢聆听



请记得填写课程回馈问卷
<https://aka.ms/Reactor/Survey>

© 2019 Microsoft Corporation. All rights reserved. The text in this document is available under the Creative Commons Attribution 3.0 License, additional terms may apply. All other content contained in this document (including, without limitation, trademarks, logos, images, etc.) are not included within the Creative Commons license grant. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. Microsoft makes no warranties, express or implied, with respect to the information provided here.