

# 一. python 开发准备

## 1. python 简介

Python是时下最流行、最火爆的编程语言之一，具体原因如下：

### 1. 简单、易学，适应人群广泛



#### 刚毕业，未来迷茫

大学/高中刚毕业，迷茫群体，  
看不到未来的方向，期待学一门  
有前景的技术



#### 跨专业转行

非计算机专业迫切要转行群体，  
期待学一门靠谱、有前景、  
易学的技术



#### 无基础 逻辑思维能力强

逻辑思维能力很强，  
想通过学一门技术来获得  
工作能力



#### 数学/统计学/物理专业

学过数学、大数据收集或分析、  
统计学、物理学等，  
是学这门课的合适人选



#### 传统运维转开发

如果你之前从事的是运维工作  
遇到瓶颈想转开发岗位，  
那Python将帮助你成功转型



#### 转型做Web全栈开发

如果你未来职业生涯致力于  
做Web全栈开发人才，  
Python会带你成功转型

### 2. 免费，开源

### 3. 应用领域广泛

备注：以下知名框架均是 Python 语言开发

Google 开源机器学习框架：TensorFlow

开源社区主推学习框架：Scikit-learn

百度开源深度学习框架：Paddle

Python 发展历史：<https://baike.baidu.com/item/Python/407313?fr=aladdin>

- Python 2.X

- Python 3.X

企业开发要求3.5版本及以上

- Python 3.5

- Python 3.6

- Python 3.7

注意：课程讲解3.7。

## 2. 解释器的作用

Python 解释器作用：运行文件

Python 解释器种类

CPython，C 语言开发的解释器[官方]，应用广泛的解释器。

IPython，基于 CPython 的一种交互式解释器。

其他解释器：

PyPy，基于 Python 语言开发的解释器。

Jython，运行在 Java 平台的解释器，直接把 Python 代码编译成 Java 字节码执行。

IronPython，运行在微软 .Net 平台上的 Python 解释器，直接把 Python 代码编译成 .Net 的字节码。

### 2.1 下载 Python 解释器

下载地址：<https://www.python.org/downloads/release/python-372/>

[单击上述链接] -- 查找目标文件：Windows x86-64 executable installer -- 单击即可下载

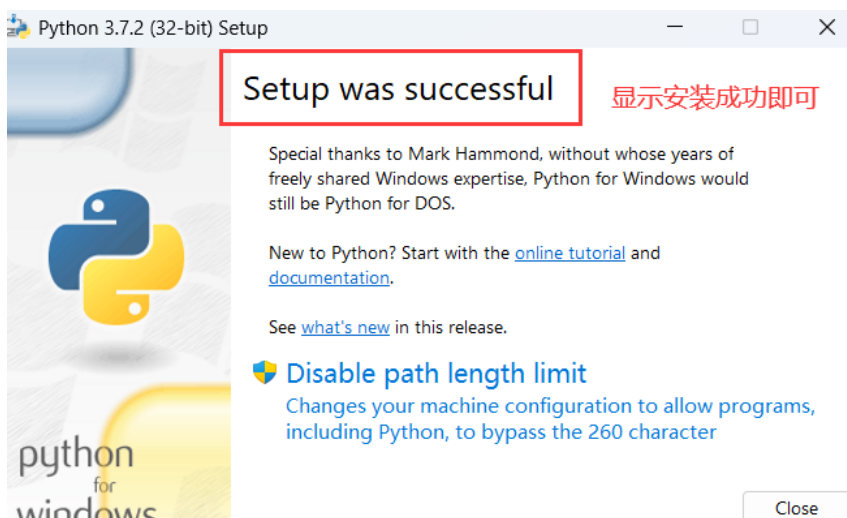
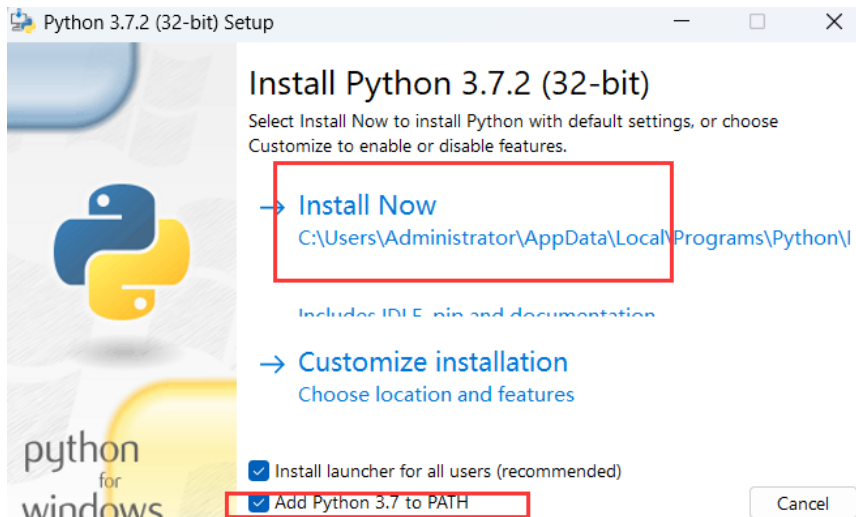
Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
<a href="#">Gzipped source tarball</a>	Source release		02a75015f7cd845e27b85192bb0ca4cb	21.8 MB	<a href="#">SIG</a>
<a href="#">XZ compressed source tarball</a>	Source release		df6ec36011808205beda239c72f947cb	16.3 MB	<a href="#">SIG</a>
<a href="#">macOS 64-bit installer</a>	macOS	for OS X 10.9 and later	0fc95e9f6d6b4881f3b499da338a9a80	26.5 MB	<a href="#">SIG</a>
<a href="#">macOS 64-bit/32-bit installer</a>	macOS	for Mac OS X 10.6 and later	d8ff07973bc9c009de80c269fd7efcca	32.8 MB	<a href="#">SIG</a>
<a href="#">Windows help file</a>	Windows		941b7d6279c0d4060a927a65dcab88c4	7.7 MB	<a href="#">SIG</a>
<a href="#">Windows x86 embeddable zip file</a>	Windows		26881045297dc1883a1d61baffeecaf0	6.2 MB	<a href="#">SIG</a>
<a href="#">Windows x86 executable installer</a>	Windows		38156b62c0cbcb03bfddeb86e66c3a0f	24.2 MB	<a href="#">SIG</a>
<a href="#">Windows x86 web-based installer</a>	Windows		1e6c626514b72e21008f8cd53f945f10	1.3 MB	<a href="#">SIG</a>
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64	f81568590bef56e5997e63b434664d58	6.7 MB	<a href="#">SIG</a>
<a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64	ff258093f0b3953c886192dec9f52763	24.9 MB	<a href="#">SIG</a>
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64	8de2335249d84fe1eeb61ec25858bd82	1.3 MB	<a href="#">SIG</a>

2.2 安装 Python 解释器

双击可执行文件 — 勾选[pip] -- [Next] -- [勾选添加环境变量] -- [Install], 按提示操作

 pycharm-professional-2023.3.4.exe	2024/3/10 13:24
 python-3.7.2.exe	2024/3/10 13:15



## 2.3 检验安装

Win+r 打开 cmd 命令窗口，输入 python，进入到解释器页面即为安装成功

```
C:\Users\Administrator>python
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

注：如有多版本，进入的是 2 开头的版本，可输入 `exit()` 命令退出解释权窗口

```
Python 2.7.13 (v2.7.13:a06454b1afaf, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
>>> exit()
```

然后输入 `python3`（为文件的名字）重新进入

```
C:\>python3
Python 3.7.2 (tags/v3.7.2:9a3ffe0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

python3.exe

### 3. 编辑器 PyCharm

#### 3.1 pycharm 的作用

PyCharm 是一种 Python IDE(集成开发环境),带有一整套可以帮助用户在使用 Python 语言开发时

提高其效率的工具, 内部集成的功能如下:


Project (项目) 管理    智能提示    语法高亮    代码跳转    调试代码    解释代码(解释器)

框架和库 (社区版没有)

#### 3.2 下载 pycharm

PythonCharm 分为专业版 (professional) 和社区版 (community)

下载地址: <http://www.jetbrains.com/pycharm/download/#section=windows>



## PyCharm Professional

The Python IDE for data science and web development

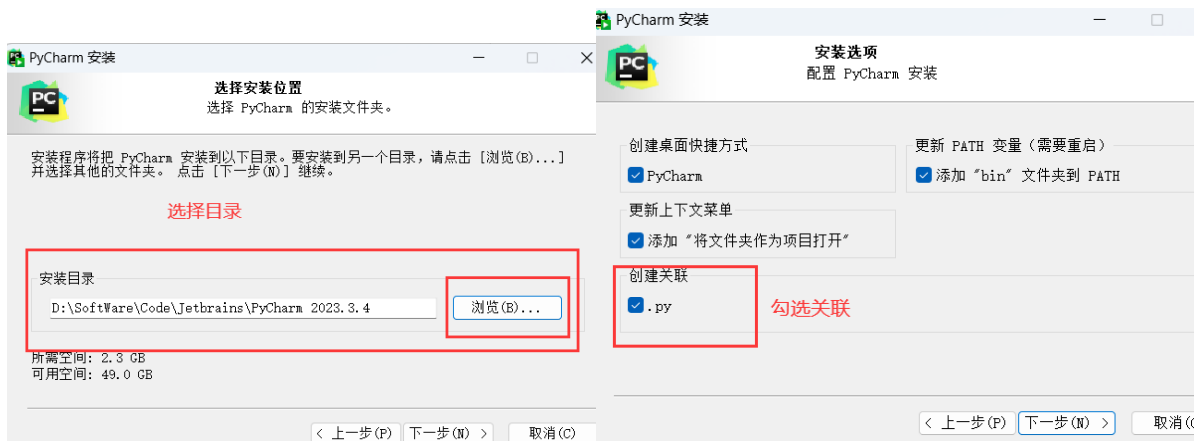
Download

.exe ▾

Free 30-day trial

pycharm-professional-2023.3.4.exe

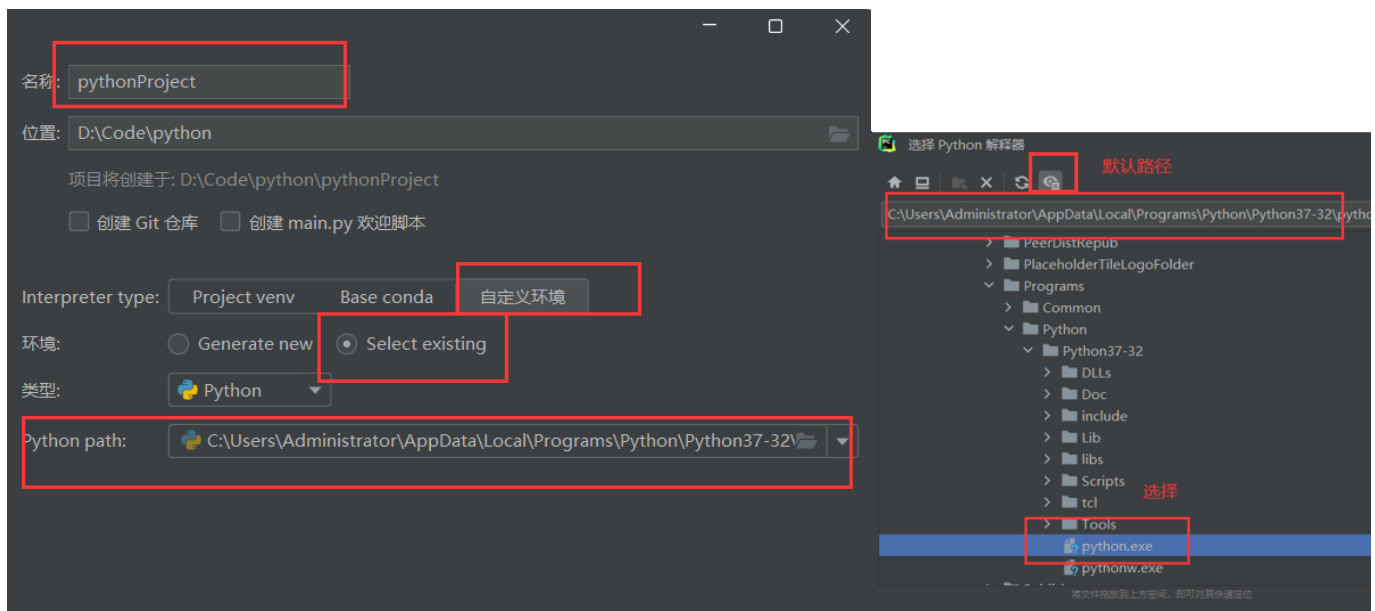
python-3.7.2.exe



### 3.3 pycharm 的使用

#### 3.3.1 新建项目

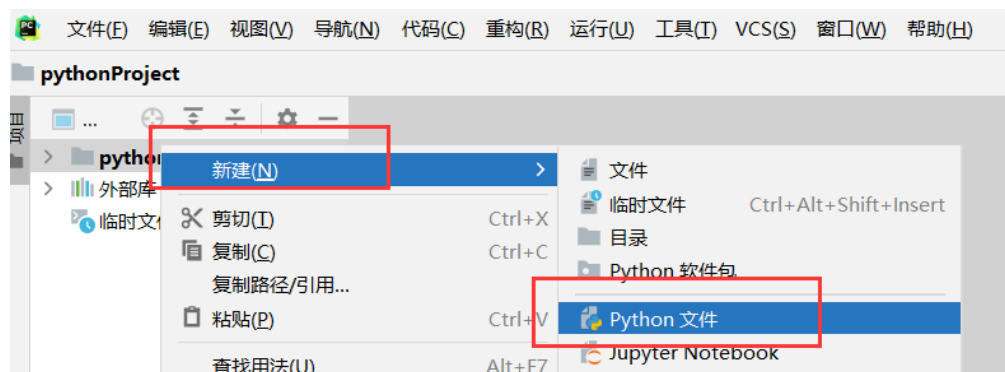
打开 PyCharm -- [Create New Project] -- 选择项目根目录和解释器版本 -- [Create]，即可完成新建一个项目



### 3.3.2 新建文件并书写代码

项目根目录或根目录内部任意位置 — 右键 -- [New] -- [Python File] -- 输入文件名 -- [OK]

注：如果是将来要上传到服务器的文件，那么文件名切记不能用中文

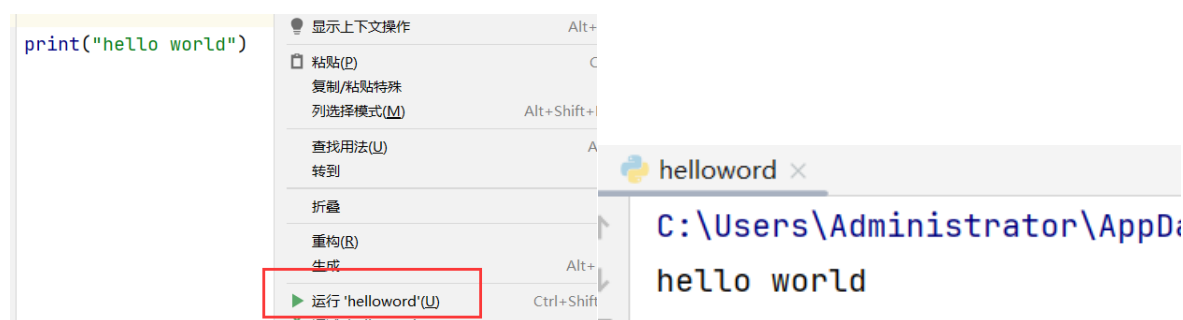


双击打开文件，并书写一个最简单的 Python 代码：

```
2 print("hello world")
```

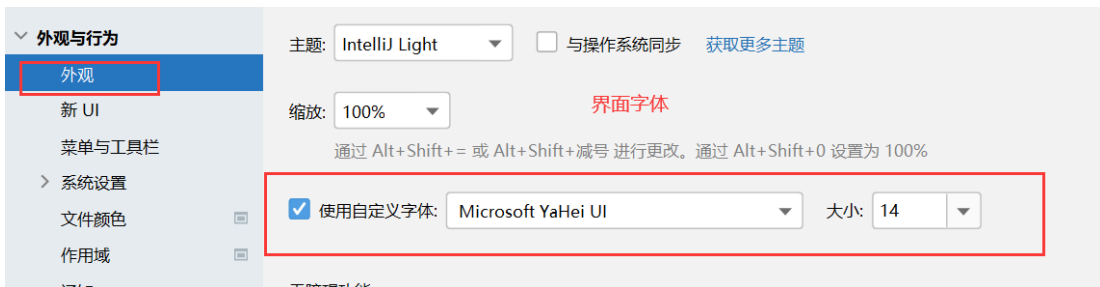
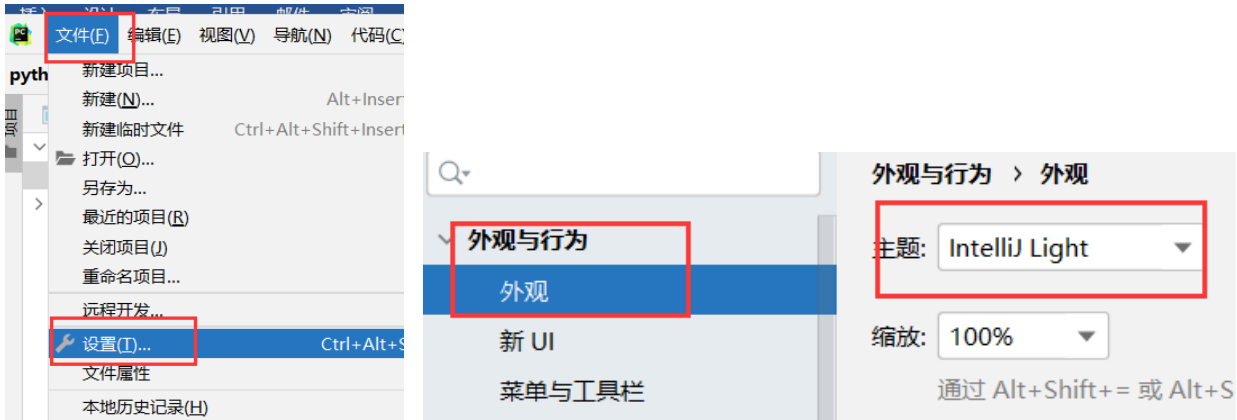
### 3.3.3 运行文件

文件打开状态 -- 空白位置 -- 右键 -- Run -- 即可调出 Pycharm 的控制台输出程序结果。

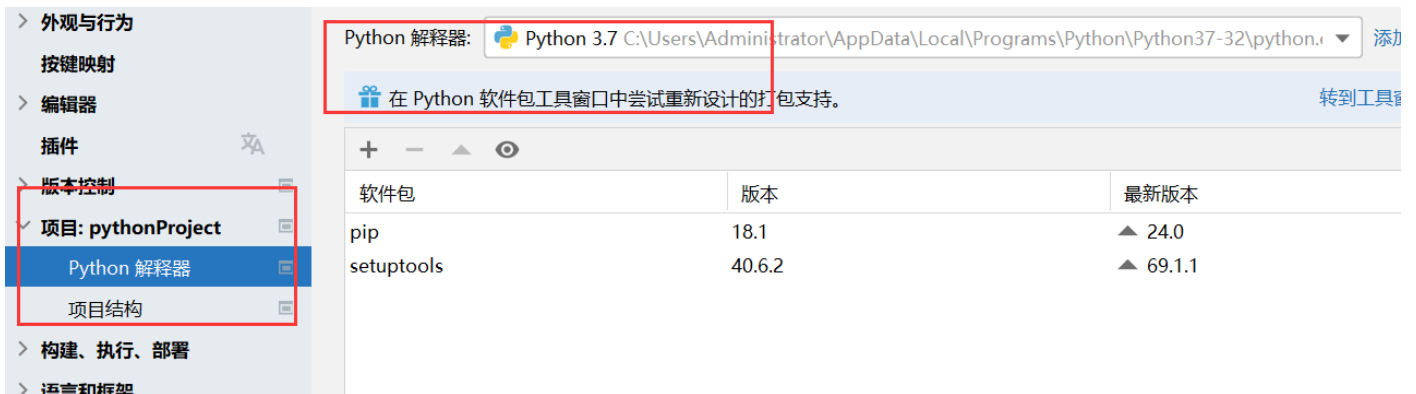


### 3.3.4 pycharm 的基本设置

[file] -- [Settings]/[Default Settings]



[Project: 项目名称] -- [Project Interpreter] -- [设置图标] -- [Add] -- 浏览到目标解释器



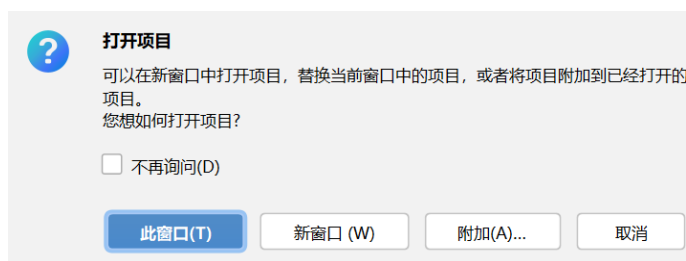
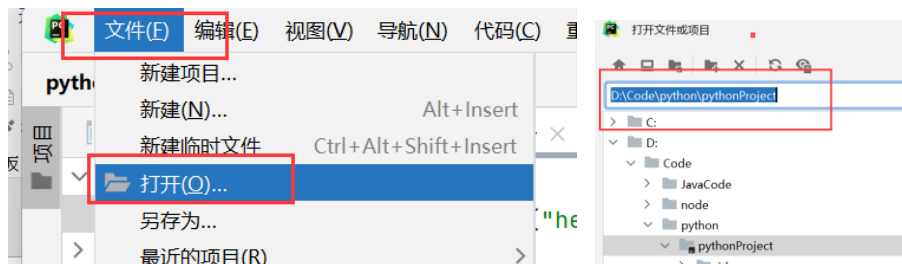


### 3.3.5 项目管理

#### 打开项目

[File] -- [Open] -- 浏览选择目标项目根目录 -- [OK] -- 选择打开项目方式。

打开项目的方式共三种，分别如下：

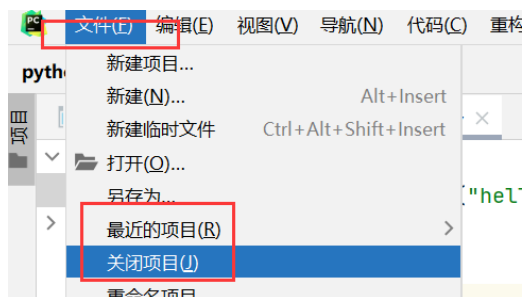


This Window（此窗口）覆盖当前项目，从而打开目标项目

New Window（新窗口）在新窗口打开，则打开两次 PyCharm，每个 PyCharm 负责一个项目

Attach（附加）将项目添加到已打开的项目当中

#### 关闭项目



## 4. 注释

通过用自己熟悉的语言，在程序中对某些代码进行标注说明，这就是注释的作用，能够大大增强程序的可读性

注释分为两类：**单行注释** 和 **多行注释**

**快捷键为：CTRL+ /**

**CTRL+SHIFT+ / （多行）**

**单行注释：**只能注释一行内容，语法如下：

**# 注释内容**

**多行注释：**可以注释多行内容，一般用在注释一段代码的情况

"""	'''
第一行注释	注释 1
第二行注释	注释 2
第三行注释	注释 3
"""	'''

## 二. Python 基础

### 1. 变量

程序中，数据都是临时存储在内存中，为了更快速的查找或使用这个数据，通常我们把

这个数据在内存中存储之后定义一个名称，这个名称就是变量

## 1.1 定义变量

变量名 = 值

变量名自定义，要满足标识符命名规则

标识符：

标识符命名规则是 Python 中定义各种名字的时候的统一规范，具体如下：

- 由数字、字母、下划线组成
- 不能数字开头
- 不能使用内置关键字
- 严格区分大小写

内置关键字：

False	None	True	and	as	assert	break	class
continue	def	del	elif	else	except	finally	for
from	global	if	import	in	is	lambda	nonlocal
not	or	pass	raise	return	try	while	with
yield							

命名习惯：

- 见名知义。
- 大驼峰：即每个单词首字母都大写，例如： **MyName**

➤ 小驼峰：第二个（含）以后的单词首字母大写，例如： `myName`

➤ 下划线：例如： `my_name`

## 1.2 使用变量

```
5 my_name = 'TOM'
6 print(my_name)
7 schoolName = 'itmy'
8 print(schoolName)
```

TOM  
itmy

## 2. bug

所谓 bug，就是程序中的错误。如果程序有错误，需要程序员排查问题，纠正错误

### 2.1 Debug 工具

Debug 工具是 PyCharm IDE 中集成的用来调试程序的工具，在这里程序员可以查看程序的执行细节和流程或者调解 bug。

Debug 工具使用步骤： 1. 打断点 2. Debug 调试

### 2.2 打断点

**断点位置**：目标要调试的代码块的第一行代码即可，即一个断点即可

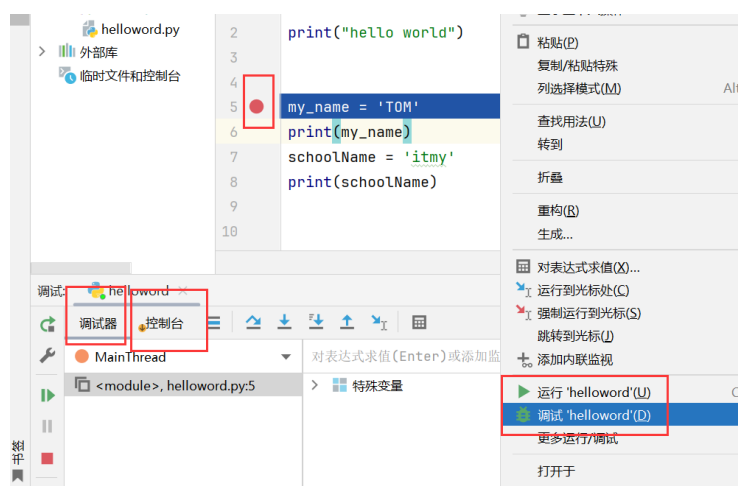
**打断点的方法**：单击目标代码的行号右侧空白位置

```

4
5 my_name = 'TOM'
6 print(my_name)
7 schoolName = 'itmy'
8 print(schoolName)
9

```

打成功断点后，在文件内部任意位置 -- 右键 -- Debug' 文件名' -- 即可调出 Debug 工具面板 -- 单击 StepOver/F8，即可按步执行代码。



面板输出解释：

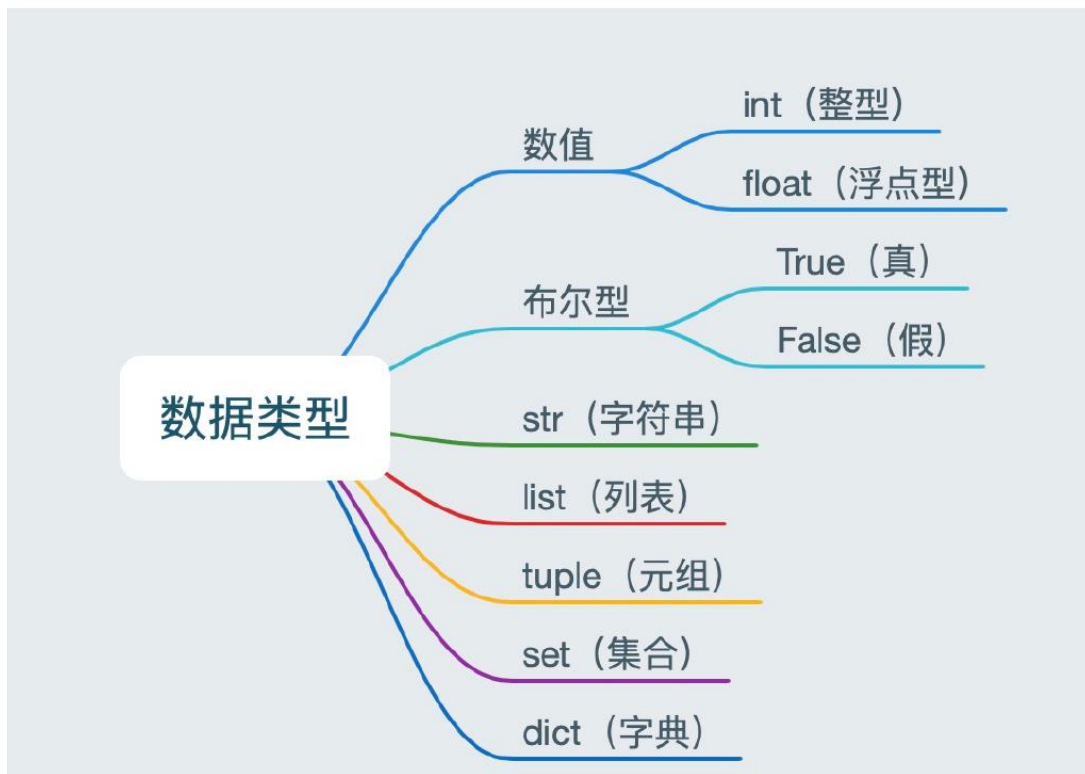
Debugger（调试器）显示程序执行过程的变量和变量的细节

Console（控制台）输出内容

### 3. 数据类型

整型：int      浮点型：float      字符串：str      布尔型：bool

元组：tuple      集合：set      字典：dict



### 3.1 检测数据的方法

检测数据类型的方法：`type()`

```
a = 1
print(type(a)) # <class 'int'> -- 整型
b = 1.1
print(type(b)) # <class 'float'> -- 浮点型
c = True
print(type(c)) # <class 'bool'> -- 布尔型
d = '12345'
print(type(d)) # <class 'str'> -- 字符串
e = [10, 20, 30]
print(type(e)) # <class 'list'> -- 列表
f = (10, 20, 30)
print(type(f)) # <class 'tuple'> -- 元组
h = {10, 20, 30}
print(type(h)) # <class 'set'> -- 集合
g = {'name': 'TOM', 'age': 20}
print(type(g)) # <class 'dict'> -- 字典
```

```
10 a = 1
11 print(type(a)) # <class 'int'> -- 整型
12 b = 1.1
13 print(type(b)) # <class 'float'> -- 浮点型
14 c = True
15 print(type(c)) # <class 'bool'> -- 布尔型
16 d = '12345'
17 print(type(d)) # <class 'str'> -- 字符串
18 e = [10, 20, 30]
19 print(type(e)) # <class 'list'> -- 列表
20 f = (10, 20, 30)
21 print(type(f)) # <class 'tuple'> -- 元组
22 h = {10, 20, 30}
23 print(type(h)) # <class 'set'> -- 集合
24 g = {'name': 'TOM', 'age': 20}
25 print(type(g)) # <class 'dict'> -- 字典
26
```

<class 'int'>  
<class 'float'>  
<class 'bool'>  
<class 'str'>  
<class 'list'>  
<class 'tuple'>  
<class 'set'>  
<class 'dict'>

4. 输出 print

作用：程序输出内容给用户

```
1 age = 20
2 print(age)
```

4.1 格式化输出

格式符号转换	格式符号转换
%s	%s 字符串
%d	%d 有符号的十进制整数
%f	%f 浮点数
%c	%c 字符
%u	%u 无符号十进制整数
%o	%o 八进制整数
%x	%x 十六进制整数（小写 ox）
%X	%X 十六进制整数（大写 OX）
%e	%e 科学计数法（小写'e'）
%E	%E 科学计数法（大写'E'）
%g	%g %f 和 %e 的简写

%G	%G %f 和%E 的简写
----	---------------

技巧:

%06d, 表示输出的整数显示位数, 不足以0补全, 超出当前位数则原样输出

%.2f, 表示小数点后显示的小数位数。

```
1 age = 20
2 name = 'itmy'
3 weight = 75.5
4 student_id = 22140339
```

↓  
📄  
📄  
🖨️  
🗑️

我的名字是itmy  
我的学号是22140339  
我的学号是022140339  
我的体重是75.50公斤  
我的体重是75.500000公斤  
我的名字是itmy, 今年20岁了  
我的名字是itmy, 明年21岁了

```
6 # 我的名字是itmy, 字符串%s
7 print('我的名字是%s' % name)
8 # 我的学号是22140339, 整数%d
9 print('我的学号是%d' % student_id)
10 # %09d输出的整数显示位数, 不足以0补全
11 print('我的学号是%09d' % student_id)
12 # 我的体重是75.50公斤, %f浮点数
13 print('我的体重是%.2f公斤' % weight)
14 print('我的体重是%.5f公斤' % weight)
15 # 我的名字是itmy, 今年20岁了
16 print('我的名字是%s, 今年%d岁了' % (name, age))
17 # 我的名字是itmy, 明年21岁了
18 print('我的名字是%s, 明年%d岁了' % (name, age + 1))
```

格式化字符串除了%s, 还可以写为 f'{表达式}'

```
19 # 我的名字是itmy, 后年22岁了
20 print(f'我的名字是{name}, 后年{age + 2}岁了')
```

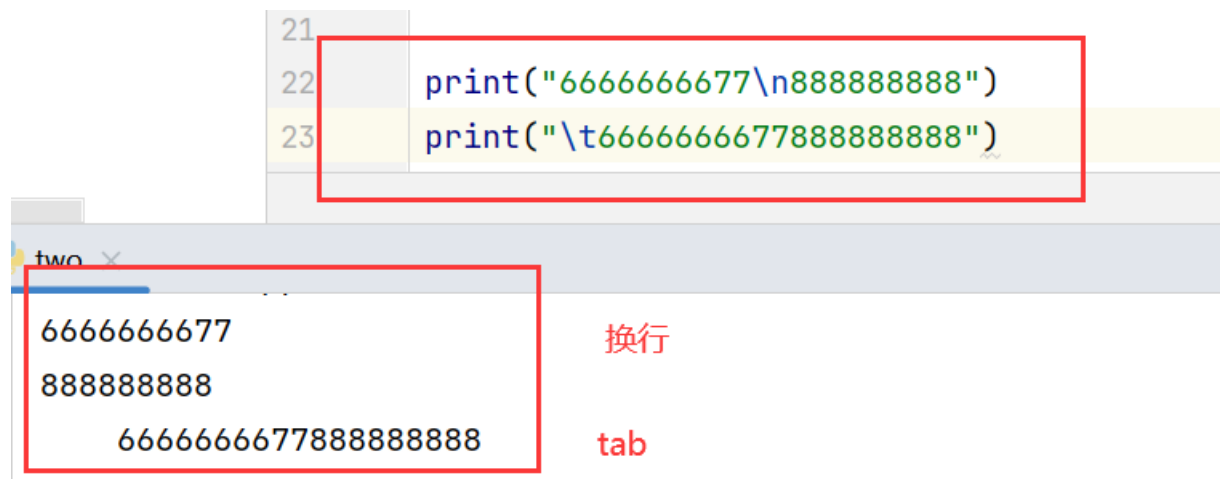
我的名字是itmy, 后年22岁了



## 4.2 转义字符

`\n` : 换行

`\t` : 制表符, 一个 tab 键 (4 个空格) 的距离



## 4.3 结束符

在 Python 中, `print()`, 默认自带 `end="\n"` 这个换行结束符, 所以导致每两个 `print` 直接会换行

展示, 用户可以按需求更改结束符。

```
print('输出的内容', end="\n")
```

## 5. 输入 input

在 Python 中, 程序接收用户输入的数据的功能即是输入

语法: `input("提示信息")`

特点:

当程序执行到 input ，等待用户输入，输入完成之后才继续向下执行。

在 python 中， input 接收用户输入后，一般存储到变量，方便使用。

在 Python 中， input 会把接收到的任意用户输入的数据都当做字符串处理。

```
2 pwd = input("请输入您的密码")
3 print(f"您输入的密码是: {pwd}")
```

请输入您的密码123456

您输入的密码是: 123456

输入后才可执行下一步

```
5 print(type(pwd))
<class 'str'>
```

6. 转换数据类型

函数	说明
int(x [,base ])	将 x 转换为一个整数
float(x )	将 x 转换为一个浮点数
complex(real [,imag ])	创建一个复数，real 为实部，imag 为虚部
str(x )	将对象 x 转换为字符串
repr(x )	repr(x ) 将对象 x 转换为表达式字符串
eval(str )	用来计算在字符串中的有效 Python 表达式,并返回一个对象
tuple(s )	将序列 s 转换为一个元组
list(s )	将序列 s 转换为一个列表
chr(x )	将一个整数转换为一个 Unicode 字符
ord(x )	将一个字符转换为它的 ASCII 整数值
hex(x )	将一个整数转换为一个十六进制字符串
oct(x )	将一个整数转换为一个八进制字符串
bin(x )	将一个整数转换为一个二进制字符串

```

8 num = input('请输入您的幸运数字: ')
9 # 打印结果
10 print(f"您的幸运数字是{num}")
11 # 检测接收到的用户输入的数据类型 -- str类型
12 print(type(num))
13 # 转换数据类型为整型 -- int类型
14 print(type(int(num)))

```

请输入您的幸运数字: 1

您的幸运数字是1

<class 'str'>

<class 'int'>

```

16 # 1. float() -- 转换成浮点型
17 num1 = 1
18 num11 = float(num1)
19 print(type(num11))
20 # 2. str() -- 转换成字符串类型
21 num2 = 10
22 print(type(str(num2)))
23 # 3. tuple() -- 将一个序列转换成元组
24 list1 = [10, 20, 30]
25 print(tuple(list1))
26 print(type(tuple(list1)))
27 # 4. list() -- 将一个序列转换成列表
28 t1 = (100, 200, 300)
29 print(list(t1))
30 print(type(list(t1)))

```

```

31 # 5. eval() -- 将字符串中的数据转换成Python表达式原本类型
32 str1 = '10'
33 str2 = '[1, 2, 3]'
34 str3 = '(1000, 2000, 3000)'
35 print(type(eval(str1)))
36 print(type(eval(str2)))
37 print(type(eval(str3)))

```

```

<class 'float'>
<class 'str'>
(10, 20, 30)
<class 'tuple'>
[100, 200, 300]
<class 'list'>
<class 'int'>
<class 'list'>
<class 'tuple'>

```

# 1. float() -- 转换成浮点型

```
num1 = 1
```

```
num11 = float(num1)
```

```
print(type(num11))
```

# 2. str() -- 转换成字符串类型

```
num2 = 10
```

```
print(type(str(num2)))
```

# 3. tuple() -- 将一个序列转换成元组

```
list1 = [10, 20, 30]
print(tuple(list1))
print(type(tuple(list1)))
# 4. list() — 将一个序列转换成列表
t1 = (100, 200, 300)
print(list(t1))
print(type(list(t1)))
# 5. eval() — 将字符串中的数据转换成 Python 表达式原本类型
str1 = '10'
str2 = '[1, 2, 3]'
str3 = '(1000, 2000, 3000)'
print(type(eval(str1)))
print(type(eval(str2)))
print(type(eval(str3)))
```

7. 运算符

7.1 算术运算符

运算符	描述	实例
+	加	1 + 1 输出结果为 2
-	减	1-1 输出结果为 0
*	乘	2 * 2 输出结果为 4
/	除	10 / 2 输出结果为 5
//	整除	整除 9 // 4 输出结果为 2
%	取余	9 % 4 输出结果为 1
**	指数	2 ** 4 输出结果为 16，即 2 * 2 * 2 * 2
()	小括号	小括号用来提高运算优先级，即 (1 + 2) * 3 输出结果为 9

注：混合运算优先级顺序： **() 高于 \*\* 高于 \* / // % 高于 + -**

7.2 赋值运算符

运算符	描述	实例
=	赋值	将=右侧的结果赋值给等号左侧的变量
+=	加法赋值运算符	c += a 等价于 c = c + a
-=	减法赋值运算符	c -= a 等价于 c = c - a
*=	乘法赋值运算符	c *= a 等价于 c = c * a

/=	除法赋值运算符	c /= a 等价于 c = c / a，除法得到的结果一定是小数
//=	整除赋值运算符	c //= a 等价于 c = c // a
%=	取余赋值运算符	c %= a 等价于 c = c % a
**=	幂赋值运算符	c **= a 等价于 c = c ** a

39404142434445464748495051

#单个变量赋值  
a =1  
print(a)  
  
#多变量赋相同值  
b = c =10  
print(c+b)  
  
#多个变量赋值  
d,e,f = 5,6,7  
print(d)  
print(e)  
print(f)

1

20

5

6

7

5455565758596061626364656667

aa = 100  
#aa = aa + 1  
aa += 1  
print(aa)  
  
bb = 2  
#bb = bb\*3  
bb \*= 3  
print(bb)  
  
cc = 10  
#cc += 3, cc = cc + 3, 优先运算右侧  
cc += 1 + 2  
print(cc)

101

6

13

7.3 比较运算符

比较运算符也叫关系运算符， 通常用来判断。

运算符	描述	实例
==	判断相等。如果两个操作数的结果相等	如 a=3, b=3

	则条件结果为真(True)，否则条件结果为假(False)	则 (a == b) 为 True
!=	不等于 。如果两个操作数的结果不相等 则条件为真(True)，否则条件结果为假(False)	如 a=3,b=3 则 (a == b) 为 True 如 a=1,b=3 则 (a != b) 为 True
>	运算符左侧操作数结果是否大于右侧操作数结果， 如果大于，则条件为真，否则为假	如 a=7,b=3 则 (a > b) 为 True
<	运算符左侧操作数结果是否小于右侧操作数结果， 如果小于，则条件为真，否则为假	如 a=7,b=3 则 (a < b) 为 False
>=	运算符左侧操作数结果是否大于等于右侧操作数结 果，如果大于，则条件为真，否则为假	如 a=7,b=3 则 (a < b) 为 False 如 a=3,b=3, 则 (a >= b) 为 True
<=	运算符左侧操作数结果是否小于等于右侧操作数结 果，如果小于，则条件为真，否则为假	如 a=3,b=3 则 (a <= b) 为 True

40a = 7  
41b = 5  
42print(a == b) # False  
43print(a != b) # True  
44print(a < b) # False  
45print(a > b) # True  
46print(a <= b) # False  
47print(a >= b) # True

↓

⇨

⇩

>>

False  
True  
False  
True  
False  
True

7.4 逻辑运算符

运算符	逻辑表达式	描述	实例
And	X and y	布尔"与": 如果 x 为 False, x and y 返回 False, 否则它返回 y 的值	True and False 返回 False
Or	X or y	布尔"或": 如果 x 是 True, 它返回 True, 否则它返回 y 的值	False or True 返回 True

Python 知识点总结documentUser : itmy22setTime:2024 年 3 月 10 日 13:14:11

Not	Not x	布尔"非": 如果 x 为 True, 返回 False 如果 x 为 False, 它返回 True	not True 返回 False not False 返回 True
-----	-------	--------------------------------------------------------	----------------------------------------

```
49 a = 1
50 b = 2
51 c = 3
52 print((a < b) and (b < c)) # True
53 print((a > b) and (b < c)) # False
54 print((a > b) or (b < c)) # True
```

7.5 数字之间的逻辑运算（拓展）

and 运算符，只要有一个值为 0，则结果为 0，否则结果为最后一个非 0 数字

```
57 a = 0
58 b = 1
59 c = 2
60 d = 0
61 # and运算符，只要有一个值为0，则结果为0，否则结果为最后一个非0数字
62 print(a and b) # 0
63 print(b and a) # 0
64 print(a and c) # 0
65 print(c and a) # 0
66 print(b and c) # 2
67 print(c and b) # 1
```

or 运算符，只有所有值为 0 结果才为 0，否则结果为第一个非 0 数字

```
57 a = 0
58 b = 1
59 c = 2
60 d = 0
61
62 # or运算符，只有所有值为0结果才为0，否则结果为第一个非0数字
63 print(a or b) # 1
64 print(a or d) # 0
65 print(a or c) # 2
66 print(b or c) # 1
```

## 8. 条件语句

条件成立执行某些代码，条件不成立则不执行这些代码

### 8.1 if 语句

语法：if 条件：

    条件成立执行的代码 1

    条件成立执行的代码 2

    .....（注：只会执行 if 语句后带缩进的代码）

注：if 语句后无缩进的代码不会执行，也代表 if 语句结束

<pre>1 if True: 2     print("条件成立，输出") 3 4 5 print("我不在if语句的缩进中，无论条件是什么都执行")</pre>	<pre>条件成立，输出 我不在if语句的缩进中，无论条件是什么都执行</pre>
------------------------------------------------------------------------------------	-------------------------------------------

### 8.2 if...else...语句

条件成立执行 if 下方的代码；条件不成立执行 else 下方的代码

语法：if 条件：

    条件成立执行的代码 1

    条件成立执行的代码 2

    .....

else:

    条件不成立执行的代码 1

    条件不成立执行的代码 2



.....

案例：输入年龄，判断是否可以上网

```
5 #input接受用用户输入的数据是字符串类型，条件是age和整型18做判断，所以这里要使用int转换数据类型
6 age = int(input("请输入您的年龄"))
7 if age >= 18:
8     print(f"您的年龄是{age}您已经成年，可以上网")
9 else:
10    print(f"您的年龄是{age}您未成年，不可以上网")
```

↓

请输入您的年龄20  
您的年龄是20您已经成年，可以上网

↓  
=

请输入您的年龄17  
您的年龄是17您未成年，不可以上网

8.3 多重判断

多重判断可以和 else 配合使用。

一般 else 放到整个 if 语句的最后，表示以上条件都不成立的时候执行的代码

语法：if 条件 1:

条件 1 成立执行的代码 1

条件 1 成立执行的代码 2

.....

elif 条件 2:

条件 2 成立执行的代码 1

条件 2 成立执行的代码 2

.....

.....

else:

以上条件都不成立执行的代码

## 案例：工龄判断

拓展：如：age>18, age<20, 可以写成 18<age<20

```
age = int(input("请输入您的年龄"))
if age < 18:
    print(f"您的年龄为{age},属于童工")
elif 18<= age <=60:
    print(f"您的年龄为{age},合法工龄")
elif age>60:
    print(f"您的年龄为{age},退休工龄")
#最后可以写成
# else:
#     print(f"您的年龄为{age},退休工龄")
```

请输入您的年龄70

您的年龄为70,退休工龄

请输入您的年龄20

您的年龄为20,合法工龄

请输入您的年龄16

您的年龄为16,属于童工

## 8.4 if 嵌套循环

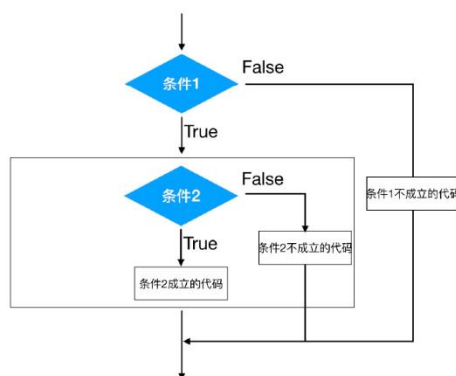
语法：if 条件1:

条件1 成立执行的代码

条件1 成立执行的代码

Else:

if 条件2:



条件 2 成立执行的代码

条件 2 成立执行的代码

Else:

...

Else:

案例：1. 如果有钱，则可以上车，如果没钱，不能上车

2. 上车后，如果有空座，可以坐下，上车后，如果没有空座，则站着等空座位

假设用 `money = 1` 表示有钱，`money = 0` 表示没有钱；`seat = 1` 表有空座，`seat = 0` 表没空座

```
30 money = 0
31 seat = 1
32 if money == 1:
33     print('您已投票，请上车')
34     if seat == 1:
35         print("本车有空座")
36     else:
37         print("本车没有空座")
38 else:
39     print("您未投票，请下车")
```



您已投票，请上车  
本车有空座

您未投票，请下车

您已投票，请上车  
本车没有空座

## 9. 三目运算符

三目运算符也叫三元运算符或三元表达式

条件成立执行的表达式    if 条件    else 条件    不成立执行的表达式

```

a = 1
b = 2
#当a大于b时, c的值和a相等, 当a大于b不成立时, 则c的值为b
c = a if a>b else b
print(c) #2
c = a if a<b else b
print(c) #1

```

先看中间条件

中间条件成立则看左边

中间条件不成立则看右边

## 10. 循环语句

循环的作用：让代码更高效的重复执行

循环分为 while 和 for 两种，最终实现效果相同

### 10.1 while 语句

语法：while 条件：

条件成立重复执行的代码 1

条件成立重复执行的代码 2

.....

```

#循环执行10次Hello World
i = 0
while i < 10:
    print("Hello World")
    i += 1

```

```

↓
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World

```

案例：计算 1 到 100 的和

```
7 i = 1
8 result = 0
9 while i <= 100:
10     result += i
11     i += 1
12 print(result)
```

5050

进程已结束，退出代码为 0

案例：计算 1 到 100 的偶数和

2500

```
7 i = 1
8 result = 0
9 while i <= 100:
10     result += i
11     i += 2
12 print(result)
```

计数器增量控制为2

```
# 方法二：条件判断和2取余数为0则累加计算
i = 1
result = 0
while i <= 100:
    if i % 2 == 0:
        result += i
    i += 1
print(result)
```

## 10.2 while 的 break 和 continue

break 和 continue 是循环中满足一定条件退出循环的两种不同方式

break 是终止此循环

```
i = 1
while i <= 5:
    if i == 4:
        print(f'吃饱了了不吃了了')
        break
    print(f'吃了了第{i}个苹果')
    i += 1
```

吃了了第1个苹果  
吃了了第2个苹果  
吃了了第3个苹果  
吃饱了了不吃了了

continue 退出当前一次循环继而执行下一次循环代码

```

25 i = 1
26 while i <= 5:
27     if i == 3:          跳过i=3
28         print(f'有虫子, 第{i}个不吃了')
29     # 在continue之前一定要修改计数器, 否则会陷入死循环
30     i += 1
31     continue
32     print(f'吃了了第{i}个苹果')
33     i += 1

```

吃了了第1个苹果  
吃了了第2个苹果  
有虫子, 第3个不吃了  
吃了了第4个苹果  
吃了了第5个苹果

拓展：建议在实际开发中，使用循环时，i 以 0 开始，i 小于多少，就执行多少次

### 10.3 while 循环嵌套

while 循环嵌套，就是一个 while 里面嵌套一个 while 的写法，每个 while 的基础语法是相同的

语法：while 条件 1:

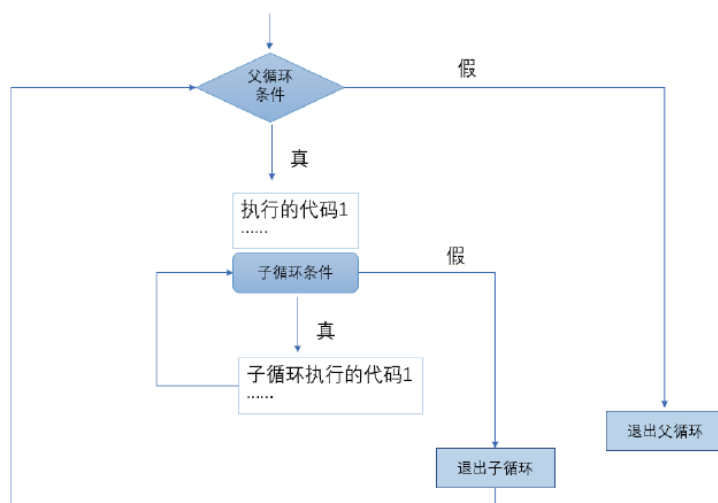
条件 1 成立执行的代码

.....

while 条件 2:

条件 2 成立执行的代码

.....



### 案例：连续 3 次输出——我错了 3 次

```
29
30 j = 0
31 while j < 3:
32     i = 0
33     while i < 3:
34         print("我错了")
35         i += 1
36     print("下次还敢")
37     j += 1
```

我错了  
我错了  
我错了  
下次还敢  
我错了  
我错了  
我错了  
下次还敢  
我错了  
我错了  
我错了  
下次还敢

### 案例：打印星星（正方形）

```
39 # 重复打印5行行星星
40 j = 0
41 while j <= 4:
42     # 一行星星的打印
43     i = 0
44     while i <= 4:
45         # 一行内的星星不能换行，取消print默认结束符\n
46         print('*', end='')
47         i += 1
48     # 每行结束要换行，这里借助一个空的print，利用print默认结束符换行
49     print()
50     j += 1
```

\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*

案例：打印金字塔

white x

53

# 重复打印5行星

↑

C:\User

54

# j表示行号

↓

\*

55

j = 0

↶

\*\*

56

while j <= 4:

↷

\*\*\*

57

# 一行星星的打印

↵

\*\*\*\*

58

i = 0

↵

\*\*\*\*\*

59

# i表示每行里面星星的个数，这个数字要和行号相等所以i要和j联动

↵

60

while i <= j:

↵

61

print('\*', end='')

↵

62

i += 1

↵

63

print()

↵

64

j += 1

进程已结

每次重新赋值清零

案例：九九乘法表

1\*1=1

1\*2=2

2\*2=4

1\*3=3

2\*3=6

3\*3=9

1\*4=4

2\*4=8

3\*4=12

4\*4=16

1\*5=5

2\*5=10

3\*5=15

4\*5=20

5\*5=25

1\*6=6

2\*6=12

3\*6=18

4\*6=24

5\*6=30

6\*6=36

1\*7=7

2\*7=14

3\*7=21

4\*7=28

5\*7=35

6\*7=42

7\*7=49

1\*8=8

2\*8=16

3\*8=24

4\*8=32

5\*8=40

6\*8=48

7\*8=56

8\*8=64

1\*9=9

2\*9=18

3\*9=27

4\*9=36

5\*9=45

6\*9=54

7\*9=63

8\*9=72

9\*9=81

67

# 重复打印9行表达式

68

j = 1

69

while j <= 9:

70

# 打印一行里面的表达式 a \* b = a\*b

71

i = 1

72

while i <= j:

73

print(f'{i}\*{j}={j\*i}', end='\t')

74

i += 1

75

print()

76

j += 1

77

10.4 for 循环语句

语法：for 临时变量 in 序列:

重复执行的代码 1

重复执行的代码 2

.....



```
i
t
m
y

str1 = "itmy"
for i in str1:
    print(i)
```

## 10.5 for 的 break 和 continue

```
i
t
遇到m结束运行

进程已结束，退出

str1 = "itmy"
for i in str1:
    if i == "m":
        print("遇到m结束运行")
        break
    print(i)
```

```
i
t
遇到m跳过运行

y

进程已结束，退出

str1 = "itmy"
for i in str1:
    if i == "m":
        print("遇到m跳过运行")
        continue
    print(i)
```

## 10.6 while...else

循环可以和 `else` 配合使用, `else` 下方缩进的代码指的是当循环正常结束之后要执行的代码

语法:     **while** 条件:

          条件成立重复执行的代码

**else:**

## 循环正常结束之后要执行的代码

```
hhh
hhh
5 hhh
hhh
hhh
下次还敢
i
i = 1
while i <= 5:
    print("hhh")
    i += 1
else:
    print("下次还敢")
```

配合 **break** 使用: **break** 终止循环的情况, **else** 下方缩进的代码将不执行

```
hhh
hhh
进程已结束, 退出代码为 0
i = 1
while i <= 5:
    if i == 3:
        break
    print("hhh")
    i += 1
else:
    print("下次还敢")
```

配合 **continue** 使用: 因为 **continue** 是退出当前一次循环, 继续下一次循环, 所以该循环在 **continue** 控制下是可以正常结束的, 当循环结束后, 则执行了 **else** 缩进的代码

```
hhh
hhh
hhh
下次还敢
进程已结束, 退出代码为 0
9 i = 1
10 while i <= 5:
11     if i == 3:
12         i += 1 # 无这段代码将陷入死循环
13         continue
14     print("hhh")
15     i += 1
16 else:
17     print("下次还敢")
```

## 10.7 for...else

语法: **for** 临时变量 **in** 序列:

重复执行的代码

...

**else:**

## 循环正常结束之后要执行的代码

```
C:\Users\A 18
i 19 str1 = "itmy"
t 20 for i in str1:
m 21     print(i)
y 22 else:
输出结束 23     print("输出结束")
```

## 配合 break 使用:

```
C:\Users\Admi 19 str1 = "itmy"
i 20 for i in str1:
t 21     if i == "m":
遇到m跳过运行 22         print("遇到m跳过运行")
23         break
24     print(i)
25 else:
26     print("输出完毕")
进程已结束，退出
```

## 配合 continue 使用:

```
C:\Users\Admini 19 str1 = "itmy"
i 20 for i in str1:
t 21     if i == "m":
遇到m跳过运行 22         print("遇到m跳过运行")
23         continue
24     print(i)
y 25 else:
输出完毕 26     print("输出完毕")
进程已结束，退出代
```

## 11. 字符串

### 11.1 一对引号字符串



The screenshot shows a Python IDE with a code editor and a console. The code in the editor defines two strings, 'Tom' and 'Rose', and a string with an escaped single quote. It then prints each string and its type. The console output shows the strings and their types as <class 'str'>.

```
name1 = 'Tom'
name2 = "Rose"
#想进行符号输出需要转义字符\
b = "i\'am Rose"
print(name1)
print(name2)
print(b)
print(type(name1))
print(type(name2))
print(type(b))
```

运行:

```
Tom
Rose
i'am Rose
<class 'str'>
<class 'str'>
<class 'str'>
```

### 11.2 三引号字符串

三引号形式的字符串支持换行

```

14 name3 = 'Tom'
15 name4 = "Rose"
16 a = 'i am Tom,
17 nice to meet you!'
18 print(name3)
19 print(name4)
20 print(a)
21 #控制台显示结果为<class 'str'> , 即数据类型为str(字符串)。
22 print(type(name3))
23 print(type(name4))
24 print(type(a))

```

```

Tom
Rose
i am Tom,
nice to meet you!
<class 'str'>
<class 'str'>
<class 'str'>

```

## 11.3 字符串输出

### 4. 输出 print

```

#引号直接输出
print('hello world')

#符号格式化输出
name = 'Tom'
print('我的名字是%s' % name)
print(f'我的名字是{name}')

```

```

hello world
我的名字是Tom
我的名字是Tom

```

## 11.4 字符串输入

### 5. 输入 input

```
33 name = input('请输入您的名字: ')
34 print(f'您输入的名字是{name}')
35 print(type(name))
36 password = input('请输入您的密码: ')
37 print(f'您输入的密码是{password}')
38 print(type(password))
```

请输入您的名字: xmy  
您输入的名字是xmy  
<class 'str'>  
请输入您的密码: 123456  
您输入的密码是123456  
<class 'str'>

### 11.5. 下标（索引）

“下标” 又叫“索引” ，就是编号。

注意：下标从 0 开始

示例：字符串 name = "abcdef" ，取到不同下标对应的数据

```
40 name = "abcdef"
41 print(name[1])
42 print(name[0])
43 print(name[2])
```

b  
a  
c  
.



## 11.6. 切片

切片是指对操作的对象截取其中一部分的操作。字符串、表、元组都支持切片操作。

**语法：** 序列 [ 开始位置下标 : 结束位置下标 : 步长 ]

注：

1. 不包含结束位置下标对应的数据， 正负整数均可；
2. 步长是选取间隔， 正负整数均可， 默认步长为 1。

```
45 name = "abcdefg"
46 print(name[2:5:1]) # cde
47 print(name[2:5]) # cde
48 print(name[:5]) # abcde
49 print(name[1:]) # bcdefg
50 print(name[:]) # abcdefg
51 print(name[:2]) # aceg
52 print(name[:-1]) # abcdef, 负1表示倒数第一个数据,
53 print(name[-4:-1]) # def
54 print(name[::-1]) # gfedcba, 步长负数即为倒数列出
55
```

```
cde
cde
abcde
bcdefg
abcdefg
aceg
abcdef
def
gfedcba
```

## 11.7 查找

所谓字符查找方法即是查找子串在字符串中的位置或出现的次数

**find()** (某个子串是否包含在这个字符串) 检测某个子串是否包含在这个字符串中，如果在则返回这个子串开始的位置下标，否则则返回-1。

**rfind()**： 和 find() 功能相同，但查找方向为右侧开始。

**语法：** 字符串序列.find(子串, 开始位置下标, 结束位置下标)

**注意：** 开始和结束位置下标可以省略，表示在整个字符序列中查找。

```

56 mystr = "hello world and itcast and itheima and Python"
57 print(mystr.find('and')) # 12
58 print(mystr.find(_sub: 'and', _start: 15, _end: 30)) # 23
59 print(mystr.find('ands')) # -1

```

↓	12
↺	23
↻	-1

**index()** (子串是否包含在这个字符串) 检测某个子串是否包含在这个字符串中，如果在返回这个子串开始的位置下标，否则则报异常。

**rindex()**: 和 index() 功能相同，但查找方向为右侧开始。

**语法:** 字符串序列.index(子串, 开始位置下标, 结束位置下标)

**注意:** 开始和结束位置下标可以省略，表示在整个字符串序列中查找。

```

63 mystr = "hello world and itcast and itheima and Python"
64 print(mystr.index('and')) # 12
65 print(mystr.index(_sub: 'and', _start: 15, _end: 30)) # 23
66 print(mystr.index('ands')) # 报错

```

```

Traceback (most recent call last):
  File "D:\Code\python\pythonProject\str.py", line 66, in <module>
    print(mystr.index('ands')) # 报错
ValueError: substring not found

```

12  
23

**count()** (子串在字符串中出现的次数) 返回某子串在字符串中出现的次数

**语法:** 字符串序列.count(子串, 开始位置下标, 结束位置下标)

**注意:** 开始和结束位置下标可以省略，表示在整个字符串序列中查找。

```

68 mystr = "hello world and itcast and itheima and Python"
69 print(mystr.count('and')) # 3
70 print(mystr.count('ands')) # 0
71 print(mystr.count(x: 'and', _start: 0, _end: 20)) # 1

```

↓	3
↺	0
↻	1

## 11.8 修改

所谓修改字符串，指的就是通过函数的形式修改字符串中的数据



## replace(): 替换

语法: 字符串序列.replace(旧子串, 新子串, 替换次数)

注意: 替换次数如果查出子串出现次数, 则替换次数为该子串出现次数

```
73 mystr = "hello world and itcast and itheima and Python"
74 print(mystr.replace(_old: 'and', _new: 'he'))
75 # 结果: hello world he itcast he itheima he Python
76 print(mystr.replace(_old: 'and', _new: 'he', _count: 10))
77 # 结果: hello world he itcast he itheima he Python
78 print(mystr)
79 # 结果: hello world and itcast and itheima and Python
80 hello world he itcast he itheima he Python
81 hello world he itcast he itheima he Python
82 hello world and itcast and itheima and Python
```

注意: 数据按照是否能直接修改分为可变类型和不可变类型两种。字符串类型的数据修改的时候不能改变原有字符串, 属于不能直接修改数据的类型即是不可变类型。

## split(): 按照指定字符分割字符串

语法: 字符串序列.split(分割字符, num)

注意: num 表示的是分割字符出现的次数, 即将来返回数据个数为 num+1 个。

```
82 mystr = "hello world and itcast and itheima and Python"
83 print(mystr.split('and'))
84 # 结果: ['hello world ', ' itcast ', ' itheima ', ' Python']
85 print(mystr.split(sep: 'and', maxsplit: 2))
86 # 结果: ['hello world ', ' itcast ', ' itheima and Python']
87 print(mystr.split(' '))
88 # 结果: ['hello', 'world', 'and', ' itcast', 'and', ' itheima', 'and', 'Python']
89 print(mystr.split(sep: ' ', maxsplit: 2))
90 # 结果: ['hello', 'world', 'and itcast and itheima and Python']
91 ['hello world ', ' itcast ', ' itheima ', ' Python']
92 ['hello world ', ' itcast ', ' itheima and Python']
93 ['hello', 'world', 'and', ' itcast', 'and', ' itheima', 'and', 'Python']
94 ['hello', 'world', 'and itcast and itheima and Python']
```

**注意：**如果分割字符是原有字符串中的子串，分割后则丢失该子串。

**join()：**用一个字符或子串合并字符串，即是将多个字符串合并为一个新的字符串。

**语法：**字符或子串.join(多字符串组成的序列)

```
92
93 list1 = ['chuan', 'zhi', 'bo', 'ke']
94 t1 = ('aa', 'b', 'cc', 'ddd')
95 print('_'.join(list1))
96 # 结果: chuan_zhi_bo_ke
97 print('...'.join(t1))
98 # 结果: aa...b...cc...ddd
99
```

chuan\_zhi\_bo\_ke  
aa...b...cc...ddd

**capitalize()：**将字符串第一个字符转换成大写

**注意：**capitalize()函数转换后，只字符串第一个字大写，其他的字符全都小小写。

```
mystr = "hello world and itcast and itheima and Python"
print(mystr.capitalize())
# 结果: Hello world and itcast and itheima and python
```

Hello world and itcast and itheima and python

**title()：**将字符每个单词首字母转换成大写。

```
mystr = "hello world and itcast and itheima and Python"
print(mystr.title())
```

Hello World And Itcast And Itheima And Python

**lower()**：将字符串中大写转小写。

```
mystr = "hello world and itcast and itheima and Python"  
print(mystr.lower())
```

```
hello world and itcast and itheima and python
```

**upper()**：将字符串中小写转大写

```
mystr = "hello world and itcast and itheima and Python"  
print(mystr.upper())
```

```
HELLO WORLD AND ITCAST AND ITHEIMA AND PYTHON
```

**lstrip()**：删除字符串左侧空白字符

```
mystr = "  hello world and itcast and itheima and Python  "  
print(mystr.lstrip())
```

```
hello world and itcast and itheima and Python  '
```

**rstrip()**：删除字符串右侧空白字符

```
12  
13 mystr = "  hello world and itcast and itheima and Python  "  
14 print(mystr.rstrip())
```

```
15  
16 hello world and itcast and itheima and Python  
17
```

**strip()**: 删除字符串两侧空白字符

```
mystr = "    hello world and itcast and itheima and Python    "  
print(mystr.strip())  
hello world and itcast and itheima and Python
```

**ljust()**: 返回一个原字符串左对齐, 并使用指定字符(默认空格)填充至对应长度的新字符串

语法: 字符串序列.ljust(长度, 填充字符)

```
116  
117 mystr = 'hello'  
118 print(mystr.ljust(_width: 10, _fillchar: '-'))  
119 hello-----  
120
```

**rjust()**: 返回一个原字符串右对齐, 并使用指定字符(默认空格)填充至对应长度的新字符串, 语法和 ljust() 相同。

语法: 字符串序列.rjust(长度, 填充字符)

```
7 mystr = 'hello'  
3 print(mystr.rjust(_width: 10, _fillchar: '-'))  
7 -----hello  
)
```

**center()**: 返回一个原字符串居中对齐, 并使用指定字符(默认空格)填充至对应长度的新字符串, 语法和 ljust() 相同。

语法: 字符串序列.center(长度, 填充字符)

```

116
117     mystr = 'hello'
118     print(mystr.center( __width: 10, __fillchar: '-'))
119     --hello---
120

```

## 11.9 判断

所谓判断即是判断真假，返回的结果是布尔型数据类型：True 或 False

**startswith()** 检查字符串是否是以指定子串开头，是则返回 True，否则返回 False。

如果设置开始和结束位置下标，则在指定范围内检查。

语法：字符串序列.startswith(子串，开始位置下标，结束位置下标)

```

121
122     mystr = "hello world and itcast and itheima and Python "
123     print(mystr.startswith('hello'))# 结果: True
124     print(mystr.startswith( __prefix: 'hello', __start: 5, __end: 20))# 结果False
125

```

**endswith()** 检查字符串是否是以指定子串结尾，是则返回 True，否则返回 False。如果设置开

始和结束位置下标，则在指定范围内检查。

语法：字符串序列.endswith(子串，开始位置下标，结束位置下标)

```

129
130     mystr = "hello world and itcast and itheima and Python"
131     print(mystr.endswith('Python'))# 结果: True
132     print(mystr.endswith('python'))# 结果: False
133     print(mystr.endswith( __suffix: 'Python', __start: 2, __end: 20))# 结果: False
134

```

**isalpha()** 如果字符串至少有一个字符并且所有字符都是字母则返回 True, 否则返回 False

```
mystr1 = 'hello'
mystr2 = 'hello12345'
print(mystr1.isalpha())# 结果: True
print(mystr2.isalpha())# 结果: False
```

True  
False

**isdigit()** 如果字符串只包含数字则返回 True 否则返回 False。

```
142 mystr1 = 'aaa12345'
143 mystr2 = '12345'
144 print(mystr1.isdigit())# 结果: False
145 print(mystr2.isdigit())# 结果: true
146
```

False  
True

**isalnum()** 如果字符串至少有一个字符并且所有字符都是字母或数字则返回 True, 否则返回 False。

```
148 mystr1 = 'aaa12345'
149 mystr2 = '12345'
150 print(mystr1.isalnum())# 结果: True
151 print(mystr2.isalnum())# 结果: False
152
```

True  
False

**isspace()** 如果字符串中只包含空白, 则返回 True, 否则返回 False

```
mystr1 = '1 2 3 4 5'
mystr2 = ' '
print(mystr1.isspace())# 结果: False
print(mystr2.isspace())# 结果: True
```

False  
True



# 综合案例

## 案例 1：猜拳游戏

需求分析：

参与游戏的角色 玩家：手动出拳      电脑：随机出拳

玩家	电脑
石头	剪刀
剪刀	布
布	石头

平局 玩家出拳 和 电脑出拳相同

### 1. 需要导入 random 模块——使用 random 模块中的随机整数功能

```
import random
random.randint(开始,结束)

# 导入random模块
import random
# 计算电脑出拳的随机数字,随机获取0,1,2之间的整数
computerSize = random.randint(a: 0, b: 2)
# 获取玩家输入的数字
personSize = int(input("请输入您的数字, 0-石头, 1-剪刀, 2-布"))

if ((personSize == 0) and (computerSize == 1)
    or (personSize == 1) and (computerSize == 2)
    or (personSize == 2) and (computerSize == 0)):
    print("玩家获胜")
elif personSize == computerSize:
    print("游戏平局")
else:
    print("电脑获胜")

import random # 导入 random 模块

computerSize = random.randint(0, 2) # 计算电脑出拳的随机数字,随机获取 0,1,2 之间的整数
personSize = int(input("请输入您的数字, 0-石头, 1-剪刀, 2-布")) # 获取玩家输入的数字
if ((personSize == 0) and (computerSize == 1)
    or (personSize == 1) and (computerSize == 2)
    or (personSize == 2) and (computerSize == 0)):
    print("玩家获胜")

elif personSize == computerSize:
```



```
    print("游戏平局")  
else:  
    print("电脑获胜")
```