

# 一、Python简介

## 1、为什么要学习Python

### ① 技术趋势

Python自带明星属性，热度稳居编程语言界前三

第1位

PYPL编程语言流行指数榜  
2020年3月

第1位

IEEE编程语言交互排行榜  
2020年1月

第3位

TIOBE编程语言排行榜  
2020年3月

<https://www.tiobe.com/tiobe-index/>

### ② 简单易学

开发代码少，精确表达需求逻辑；33个关键字，7种基本数据类型；语法规则简单，接近自然语言。

Python Hello World

```
print("hello World!");
```

PK

Java Hello World

```
public class Main{  
    public static void main(String[] args) {  
        System.out.println("HelloWorld!");  
    }  
}
```

### ③ 应用广泛

Python语言涉及IT行业70%以上的技术领域

自动化运维

自动化测试

数据分析

Web应用开发

桌面应用开发

操作系统管理

服务器软件

人工智能

## 2、Python语言的诞生

1989年，为了打发圣诞节假期，龟叔(吉多·范·罗苏姆)开始写Python语言的编译器；  
1991年，第一个Python编译器诞生  
Python这个名字，来自龟叔所挚爱的电视剧Monty Python's Flying Circus (蒙蒂·蟒蛇的飞行马戏团)



### 3、Python语言的优缺点

#### 优点

**简单**：Python是一种代表简单主义思想的语言。阅读一个良好的Python程序就感觉像是在读英语一样，Python的这种代码本质是它最大的优点之一。它使你能够专注于解决问题而不是去搞明白语言本身。

**易学**：就如同你即将看到的一样，Python极其容易上手。前面已经提到了，Python有极其简单的语法。

**免费、开源**：Python开源的。简单地说，你可以自由地阅读它的源代码、对它做改动、这是为什么Python如此优秀的原因之一，它是由一群希望看到一个更加优秀的Python的人创造并经常改进着的。

**可移植性**：由于它的开源本质，Python已经被移植在许多平台上（经过改动使它能够工作在不同平台上）。如果你小心地避免使用依赖于系统的特性，那么你的所有Python程序无需修改就可以在下述任何平台上面运行。

**丰富的库（避免造轮子）**：Python标准库确实很庞大。它可以帮助你处理各种工作，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV文件、密码系统、GUI（图形用户界面）、Tk和其他与系统有关的操作。记住，只要安装了Python，所有这些功能都是可用的。这被称作Python的“功能齐全”理念。pandas、sklearn、pytorch。。。

#### 缺点

Python语言非常完善，没有明显的短板和缺点，唯一的缺点就是执行效率慢，这个是解释型语言所通有的，同时这个缺点也将被计算机越来越强大的性能所弥补。

## 4、Python版本的选择

Python3.x

Python2.7、Python3.6、Python3.7、Python3.8、Python3.9...

在生产环境中，我们一般不会选择最新版本的Python，因为可能会存在未知Bug，所以一般强烈建议大家在选择软件版本时，向前推1~2个版本。

## 5、Python解析器

```
print('Hello World')
```

由于Python属于高级语言，其并不能直接在计算机中运行，因为缺少Python语言的运行环境：Python解析器



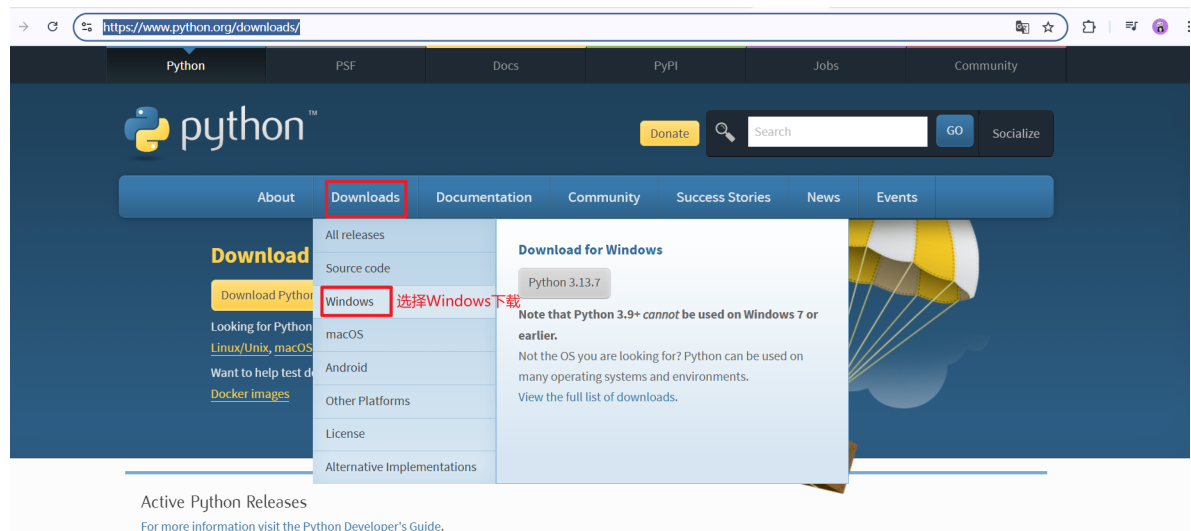
Python解析器的作用：就是把Python代码转换为计算机底层可以识别的机器语言，如0101...

# 二、Python环境安装

## 1、下载Python

Python官网：<https://www.python.org/downloads/>

根据电脑系统选择对应的系统下载



选择对应的版本进行下载(版本建议3.10.X以上即可)

## Stable Releases

- Python 3.13.7 - Aug. 14, 2025

Note that Python 3.13.7 cannot be used on Windows 7 or earlier.

- Download Windows installer (64-bit) 选择对应同版本64位下载
- Download Windows installer (32-bit)
- Download Windows installer (ARM64)
- Download Windows embeddable package (64-bit)
- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (ARM64)
- Download Windows release manifest

- Python 3.13.6 - Aug. 6, 2025

Note that Python 3.13.6 cannot be used on Windows 7 or earlier.

- Download Windows installer (64-bit)
- Download Windows installer (32-bit)
- Download Windows installer (ARM64)
- Download Windows embeddable package (64-bit)
- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (ARM64)
- Download Windows release manifest

- Python 3.13.5 - June 11, 2025

## Pre-releases

- Python install manager 25.0 beta 14 - Aug. 27, 2025

- Download Installer (MSIX)
- Download MSI package

- Python install manager 25.0 beta 13 - Aug. 14, 2025

- Download Installer (MSIX)
- Download MSI package

- Python 3.14.0rc2 - Aug. 14, 2025

- Download Windows installer (64-bit)
- Download Windows installer (32-bit)
- Download Windows installer (ARM64)
- Download Windows embeddable package (64-bit)
- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (ARM64)
- Download Windows release manifest

- Python install manager 25.0 beta 12 - July 22, 2025

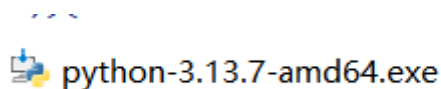
- Download Installer (MSIX)
- Download MSI package

- Python 3.14.0rc1 - July 22, 2025

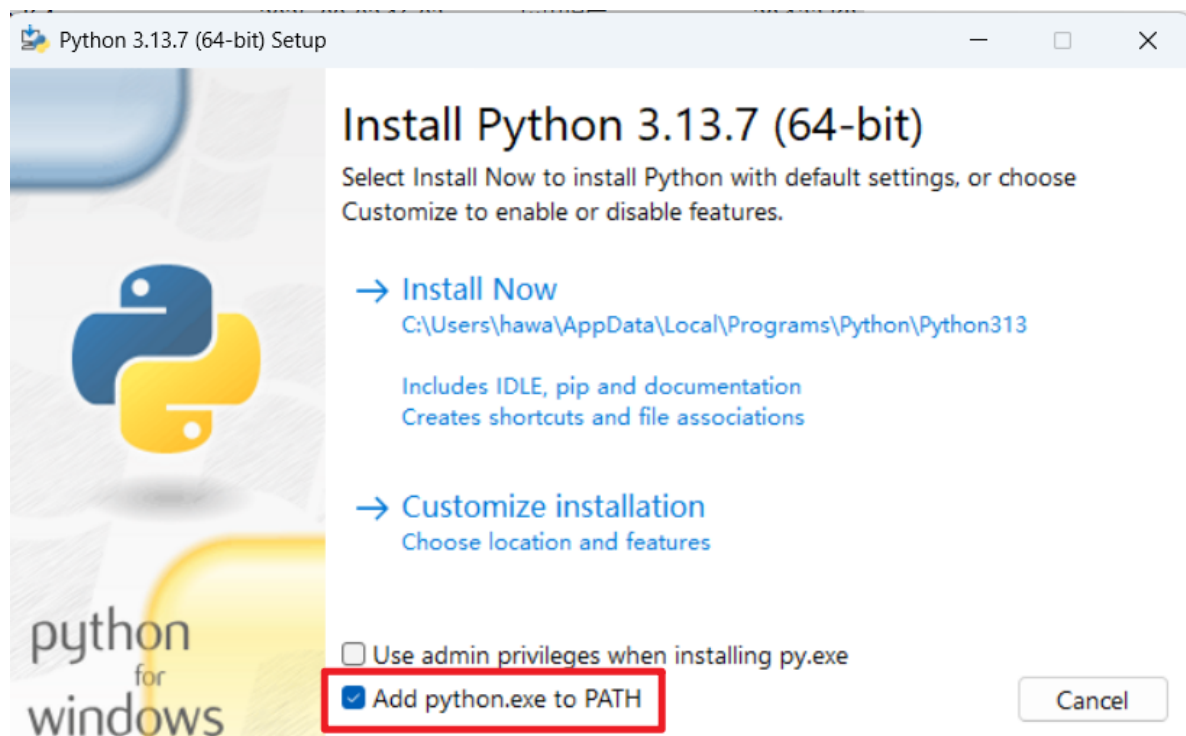
- Download Windows installer (64-bit)
- Download Windows installer (32-bit)

## 2、安装Python

双击下载的安装包



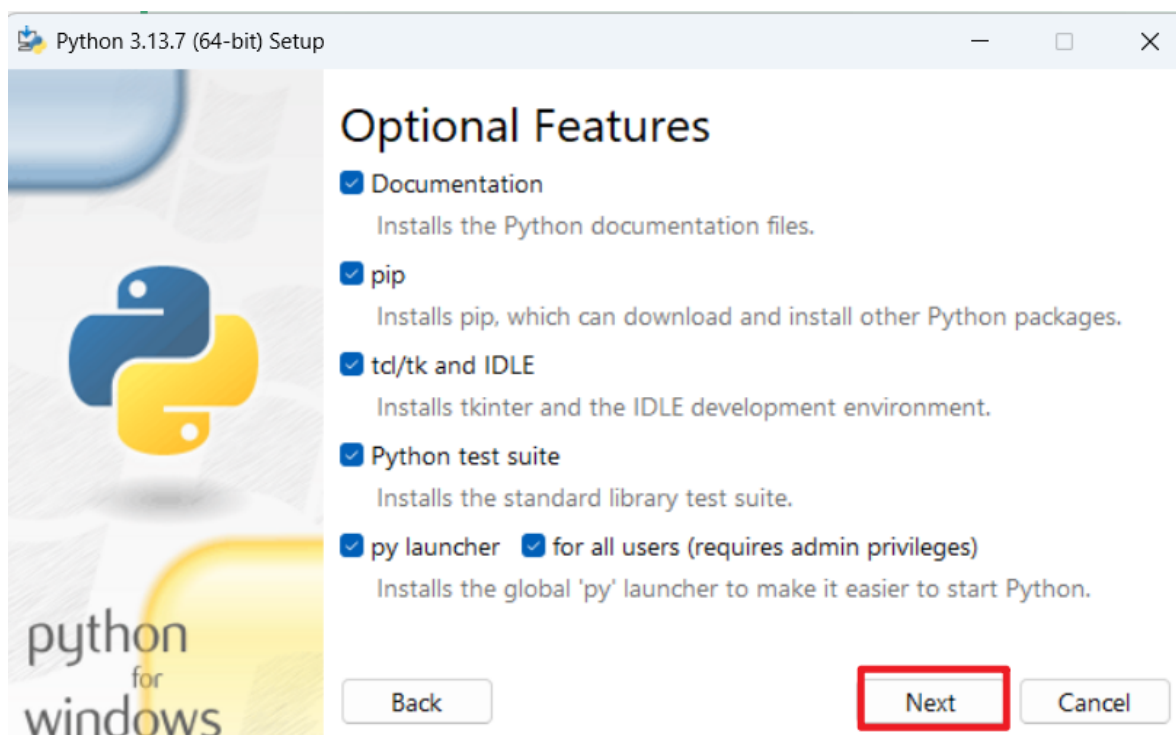
添加python环境变量



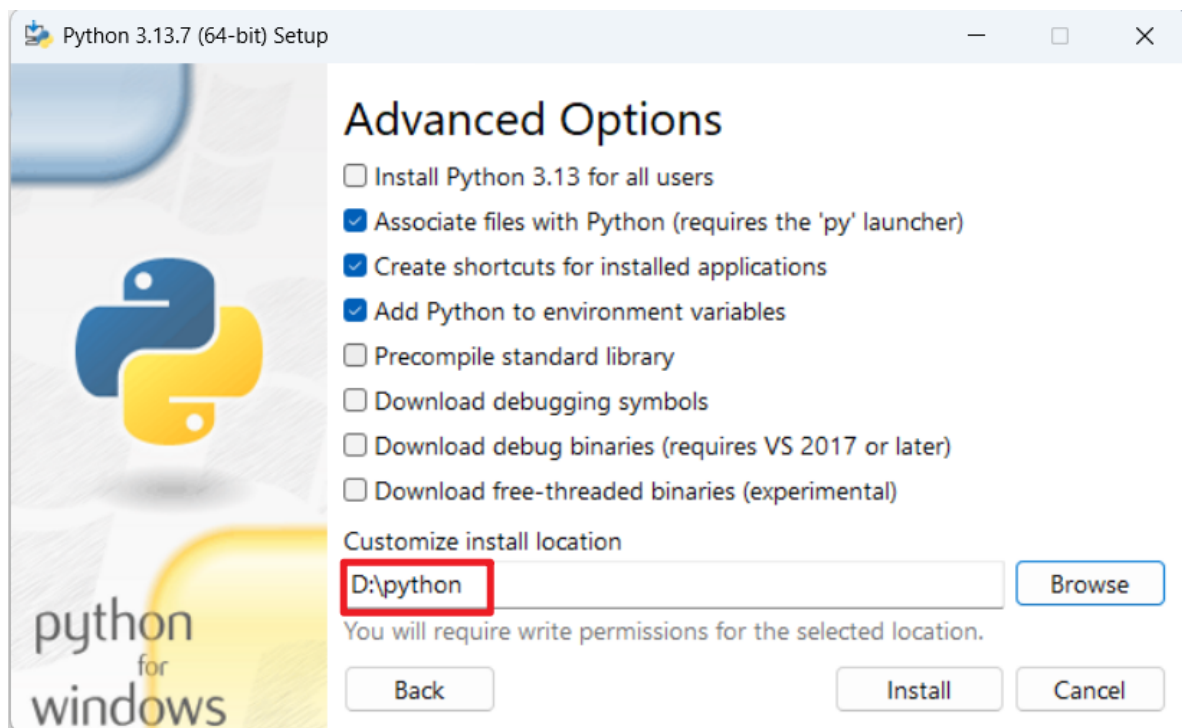
选择自定义安装



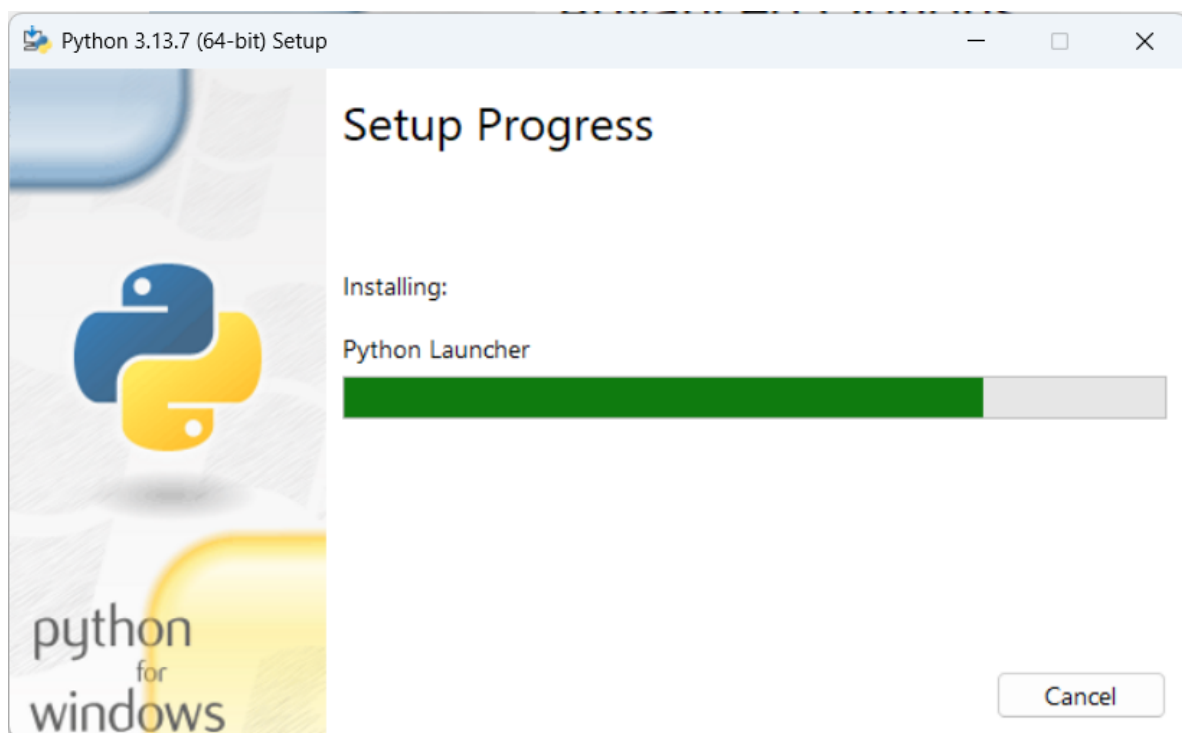
默认选项，下一步

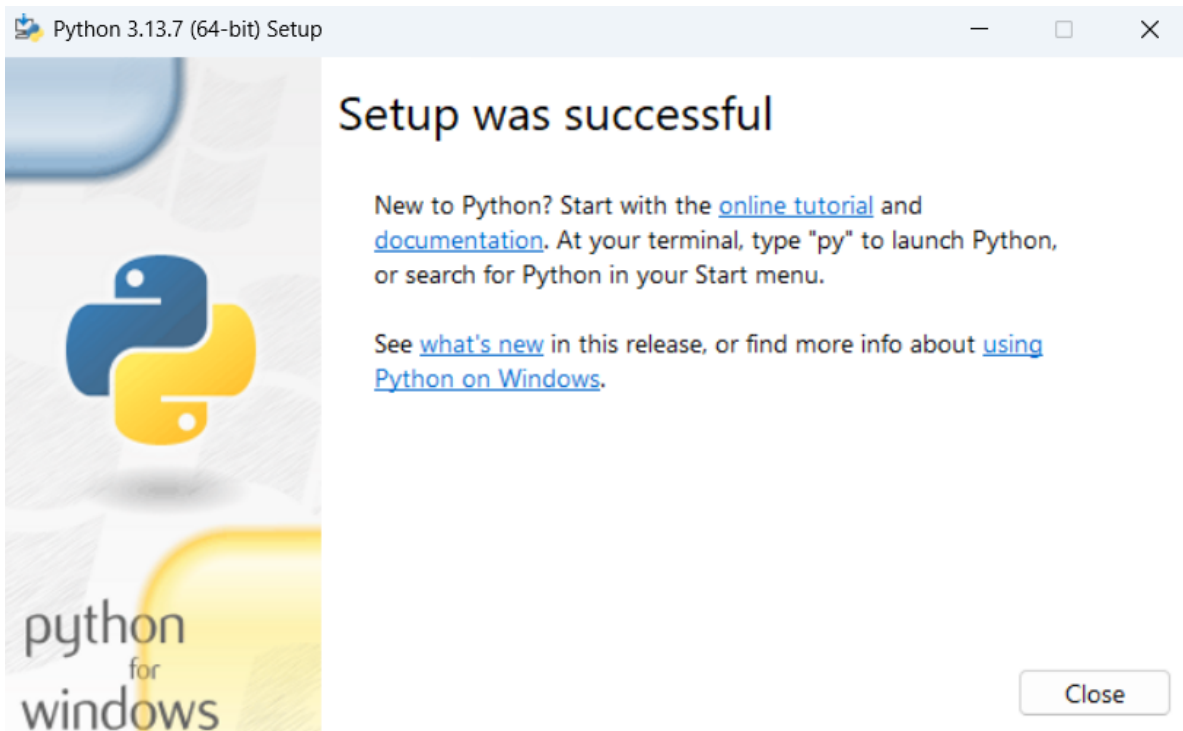


自行选择安装目录(目录中不要出现中文等特殊字符)，选择 **Install**



出现权限选择框，选择 是 ，等待安装。待安装完成后选择 Close

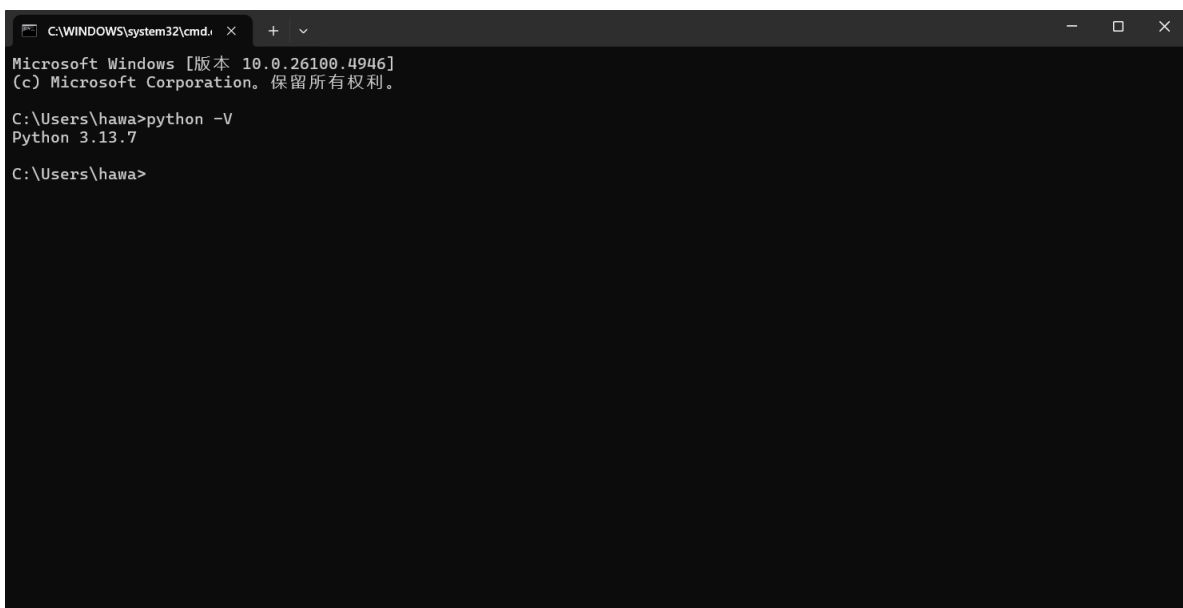




至此python解释器安装完成

### 3、测试环境

打开Windows终端( Win+R ), 输入命令 `python -V`



正确显示版本即表示安装成功

## 三、PyCharm下载安装

### 1、PyCharm下载



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

下载地址：<https://www.jetbrains.com.cn/pycharm/download/?section=windows>

选择版本进行下载

 JETBRAINS

AI 开发者工具 团队工具 教育 解决方案 支持 在线商店

PyCharm JetBrains for Data

用例 早期试用计划 最新变化 功能 学习 定价 下载

Windows macOS Linux

 **PyCharm** 统一产品

您需要的唯一 Python IDE

下载 .exe (Windows)

永久免费，另含一个月的 Pro



版本: 2025.2.1  
内部版本号: 252.25557.130  
2025年8月28日

系统要求  
安装说明

其他版本  
第三方软件

点击 [社区版](#) 下载

## 其他版本

版本 2025.2 2025.2.1

PyCharm

PyCharm Community Edition

[2025.2.1 - Linux \(tar.gz\)](#)

[2025.2.1 - Linux \(tar.gz\)](#)

[2025.2.1 - Linux ARM64 \(tar.gz\)](#)

[2025.2.1 - Linux ARM64 \(tar.gz\)](#)

[2025.2.1 - Windows \(exe\)](#)

[2025.2.1 - Windows \(exe\)](#)

[2025.2.1 - Windows ARM64 \(exe\)](#)

[2025.2.1 - Windows ARM64 \(exe\)](#)

[2025.2.1 - ZIP archive \(win.zip\)](#)

[2025.2.1 - ZIP archive \(win.zip\)](#)

[2025.2.1 - ZIP archive for Windows ARM64 \(win.zip\)](#)

[2025.2.1 - ZIP archive for Windows ARM64 \(win.zip\)](#)

[2025.2.1 - macOS \(dmg\)](#)

[2025.2.1 - macOS \(dmg\)](#)

版本: 2025.2.1 (版本说明)

构建: 252.25557.130

发布日期: 2025年8月28日

主要版本: 2025.2

发布日期: 2025年8月4日

PyCharm 第三方软件

PyCharm Community Edition 第三方软件


系统要求



- 64 位 Windows 10 1809 及更高版本，或 Windows Server 2019 及更高版本
- 最低 2 GB 可用 RAM 和 8 GB 系统总 RAM
- 3.5 GB 硬盘空间，推荐 SSD
- 最低屏幕分辨率 1024x768

## 2、PyCharm安装

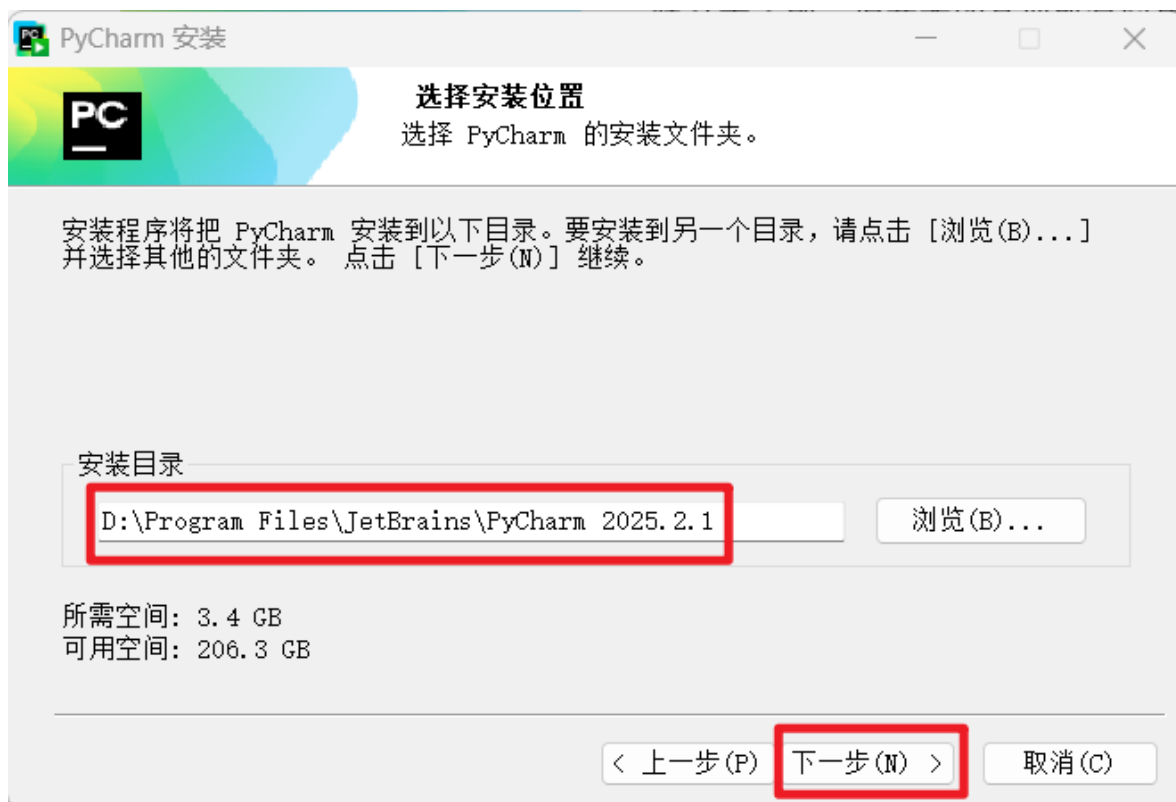
双击安装包

 pycharm-2025.2.1.exe

选择 下一步



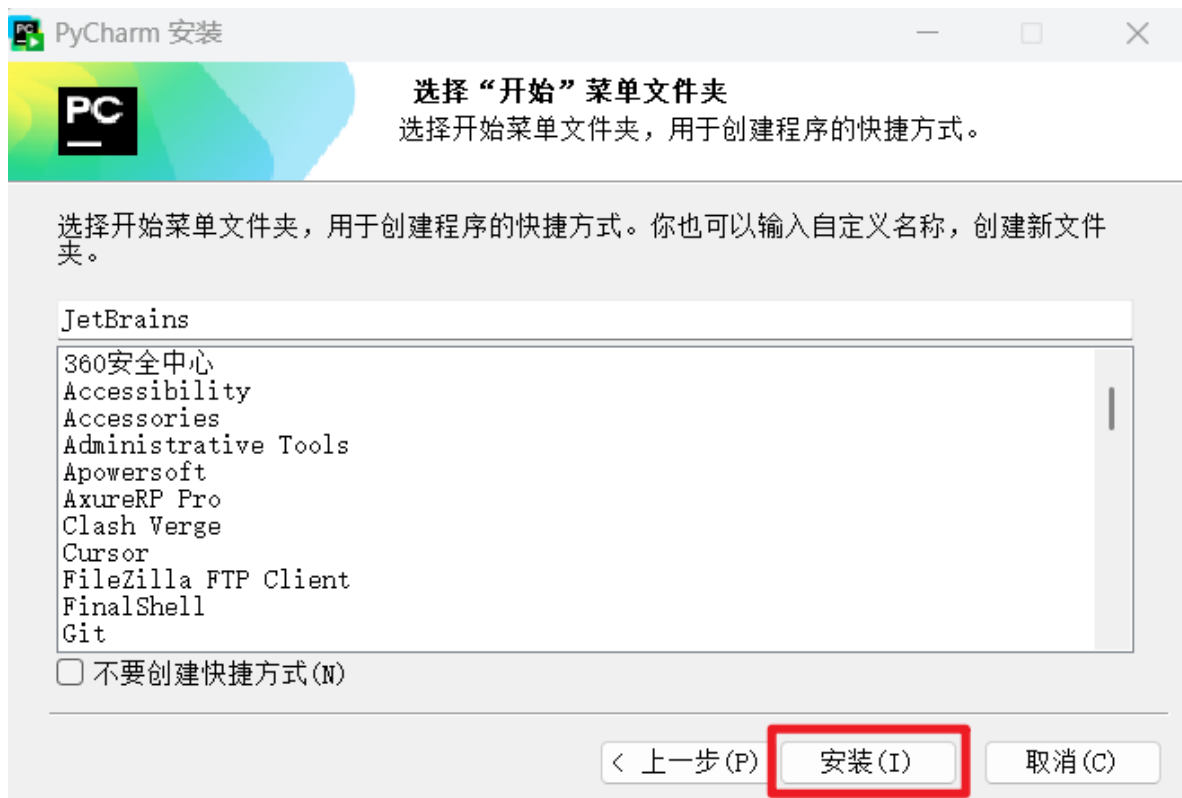
根据选择软件安装目录(不要出现中文等特殊字符)，选择下一步



创建桌面快捷方式，选择关联.py文件



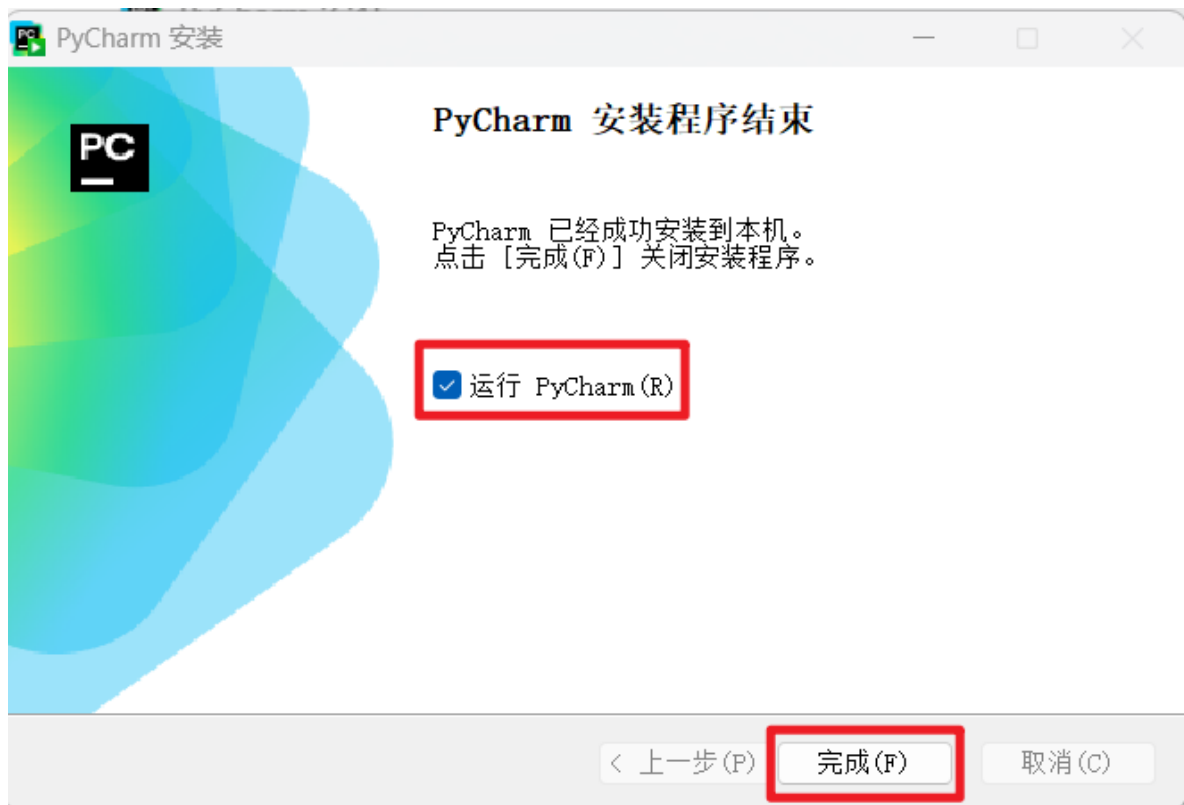
选择 安装



等待安装...

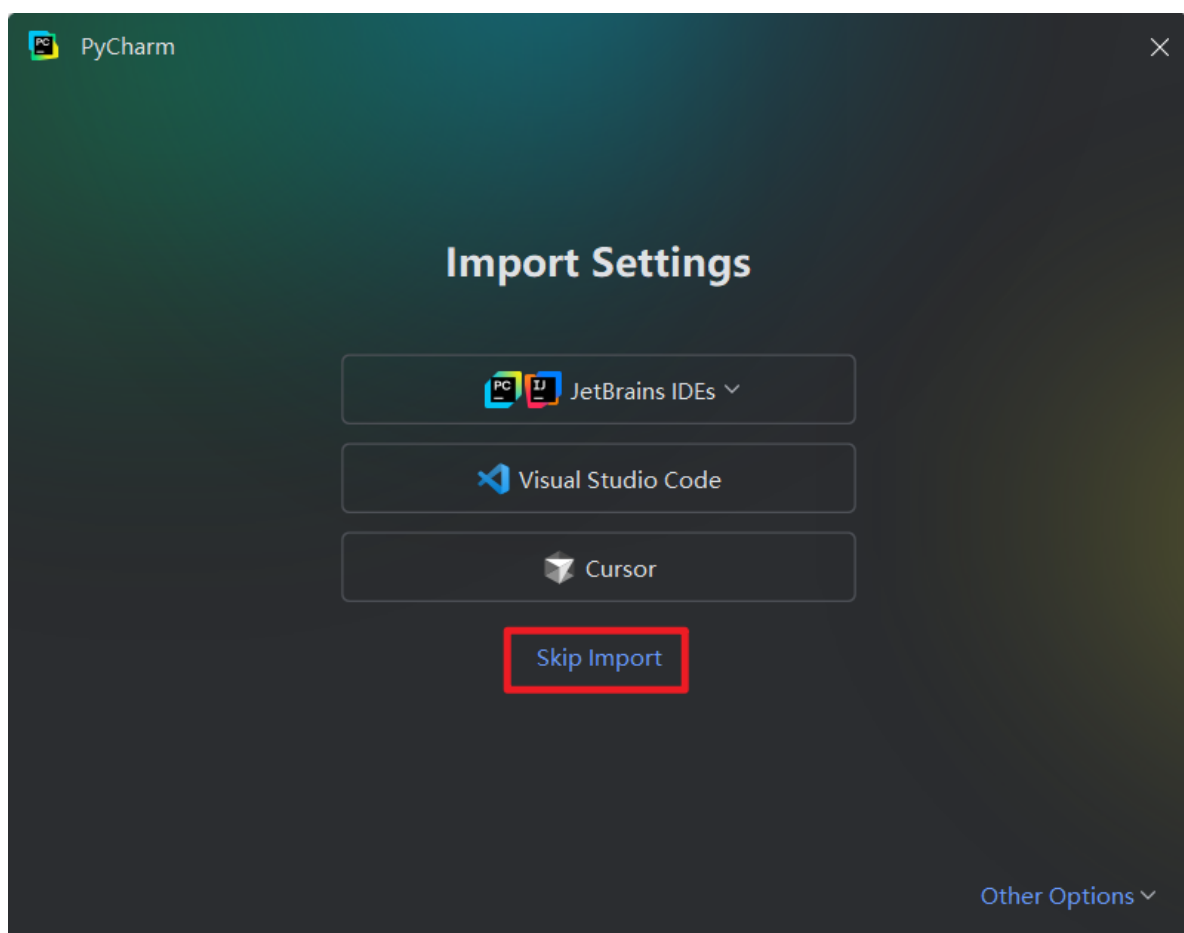


安装完成启动PyCharm

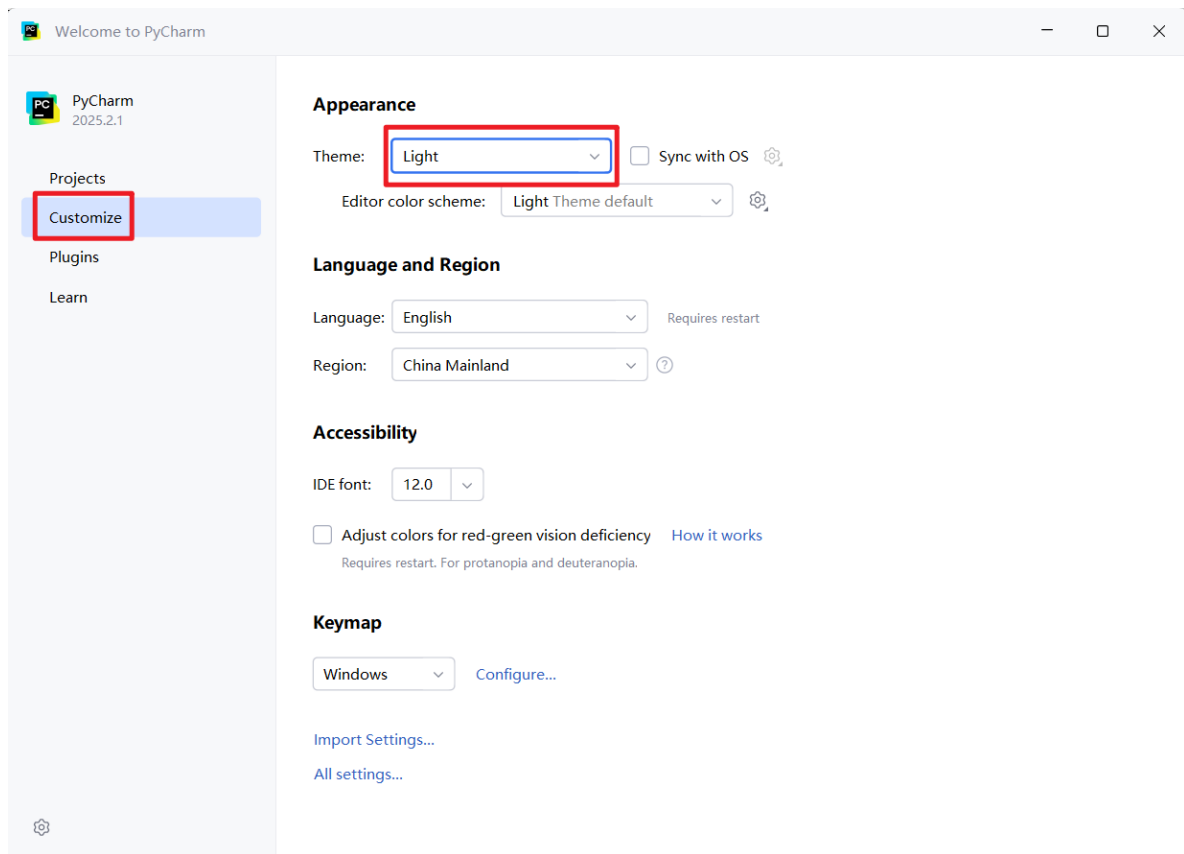


### 3、常用配置

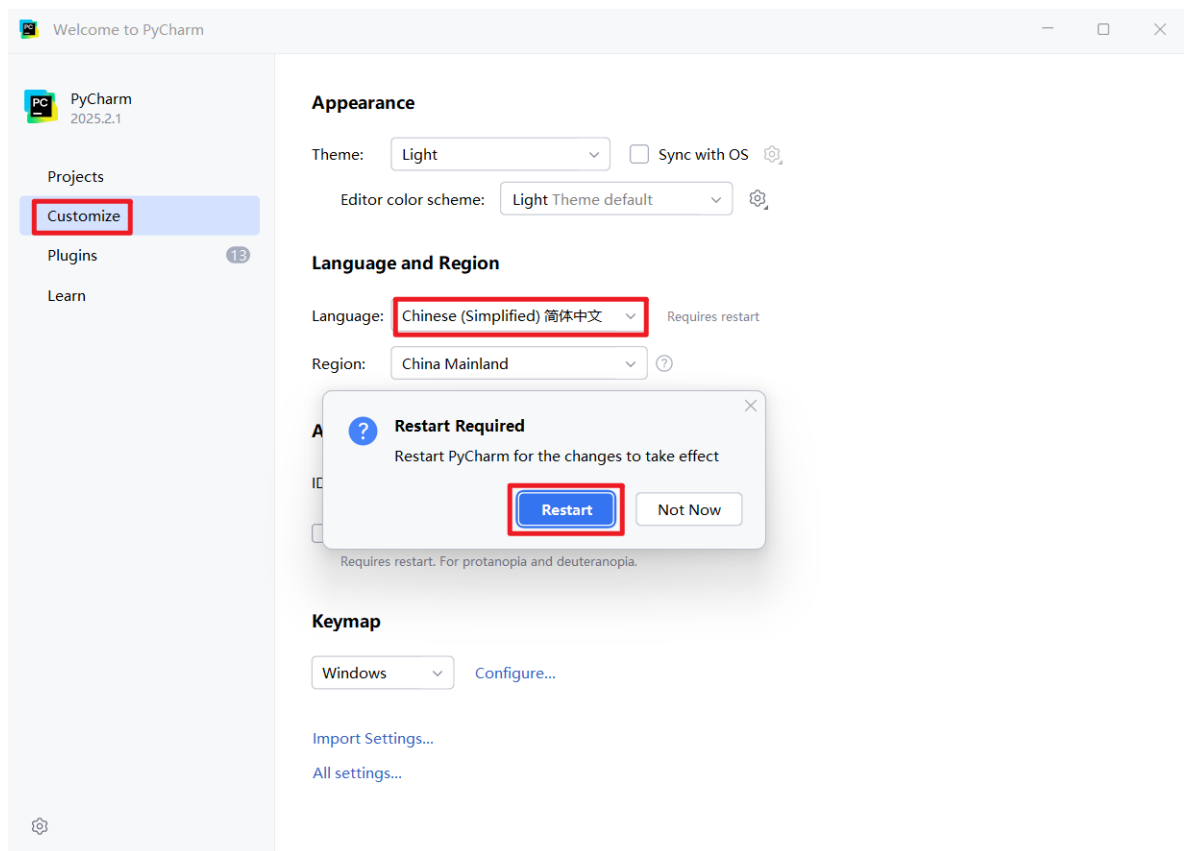
是否选择导入配置(如果没有选择跳过)



选择喜欢的主题



选择语言为中文，选择 **Restart**



## 4、创建工程

选择 **新建项目**



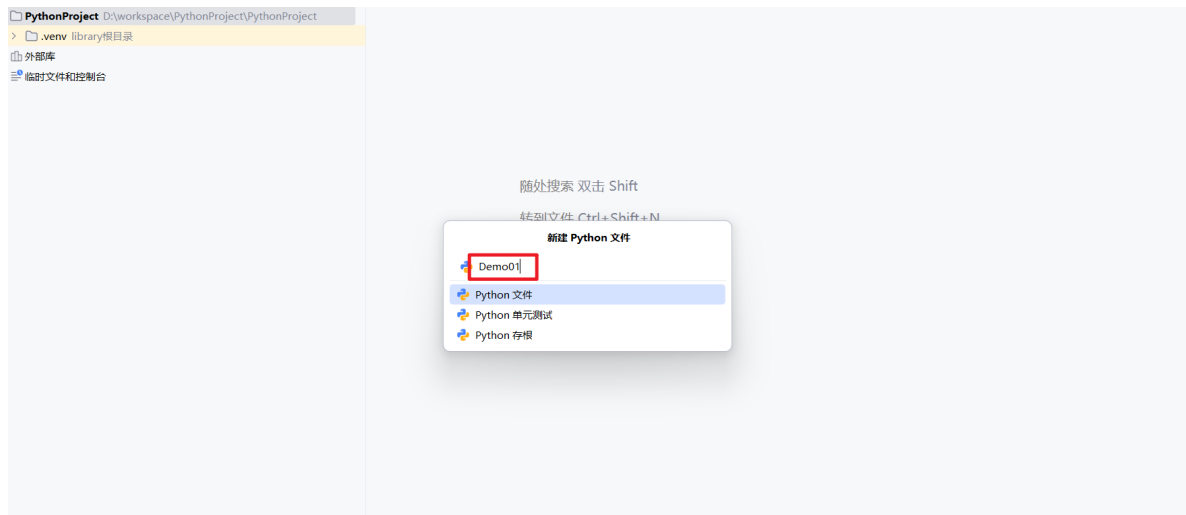
选择项目的保存路径以及项目名称(不要出现中文等特殊字符)，以及python的编辑器（一般都会自动识别），选择 **创建**



## 5、编辑代码

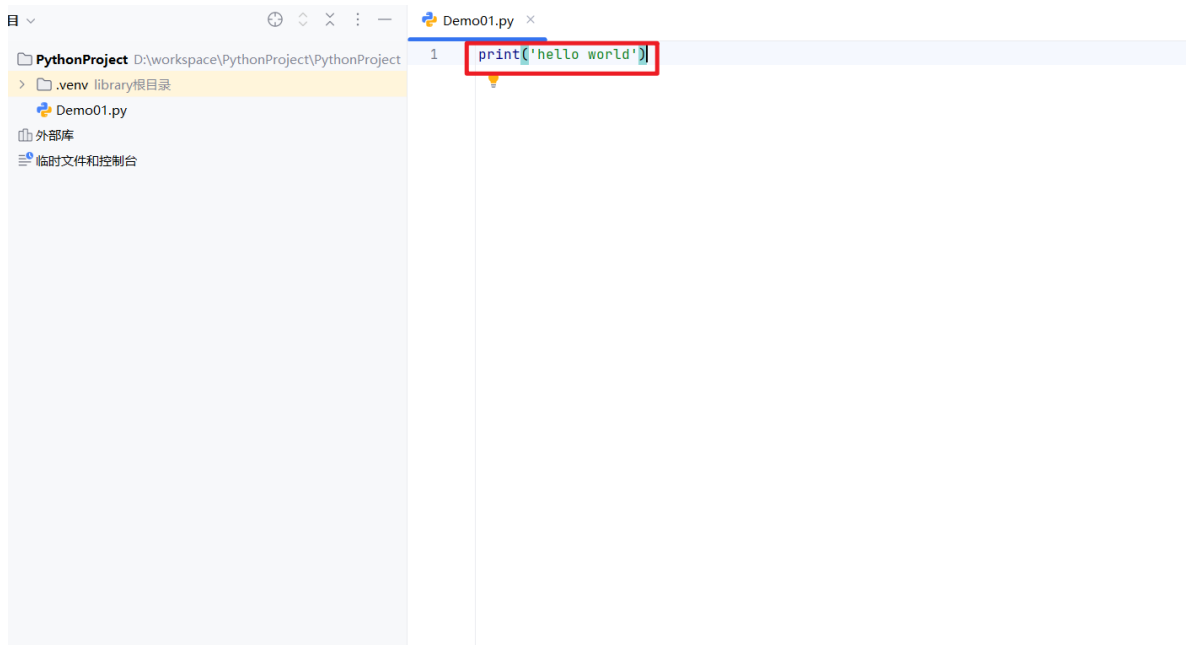
右击选择新建，选择Python文件

文件名称(不要出现中文等特殊字符)，并回车



编写Python代码

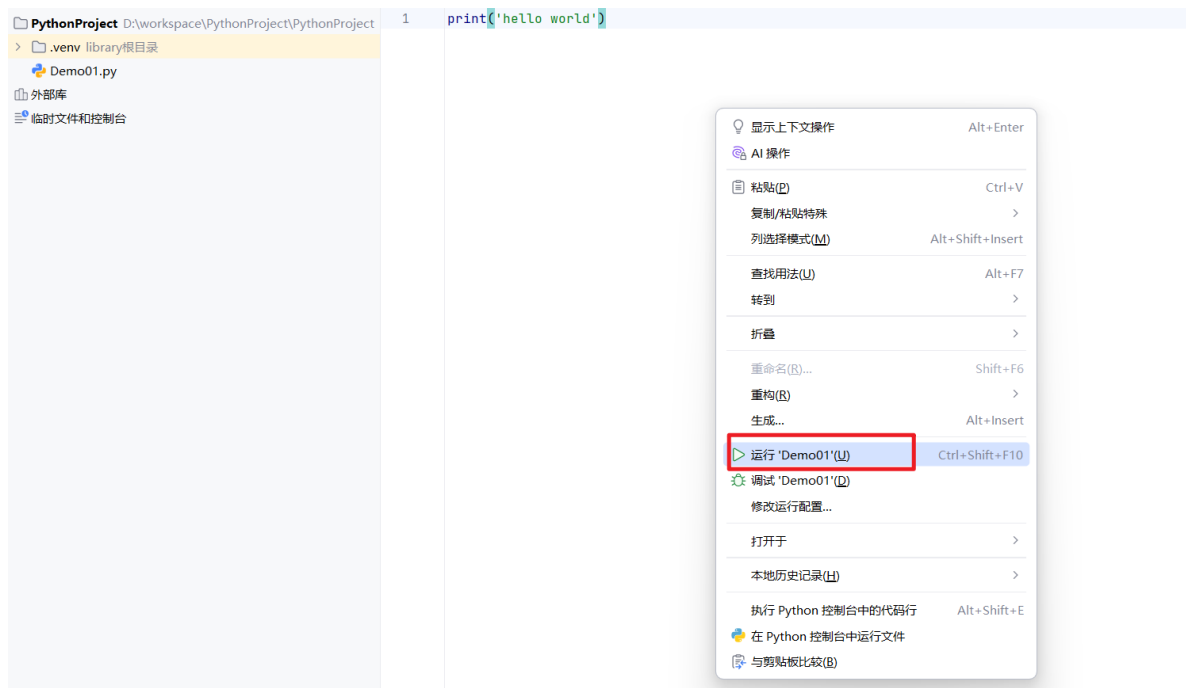
```
print('hello world')
```



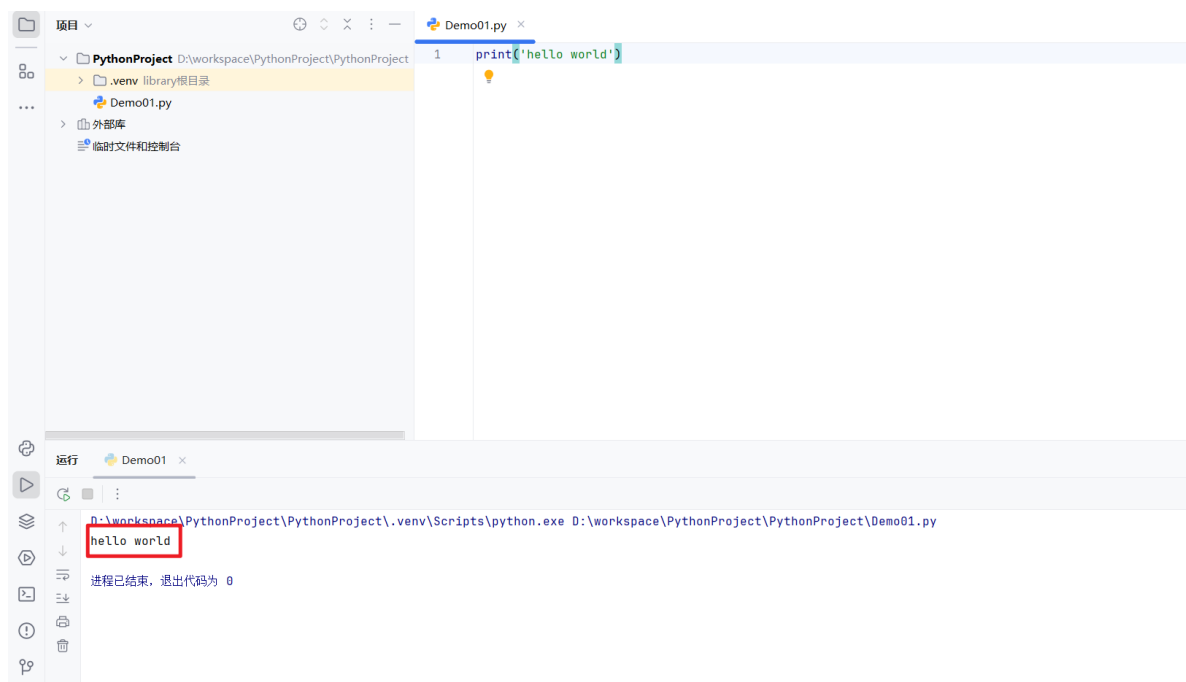
## 6、运行代码

右击代码空白处，选择 运行

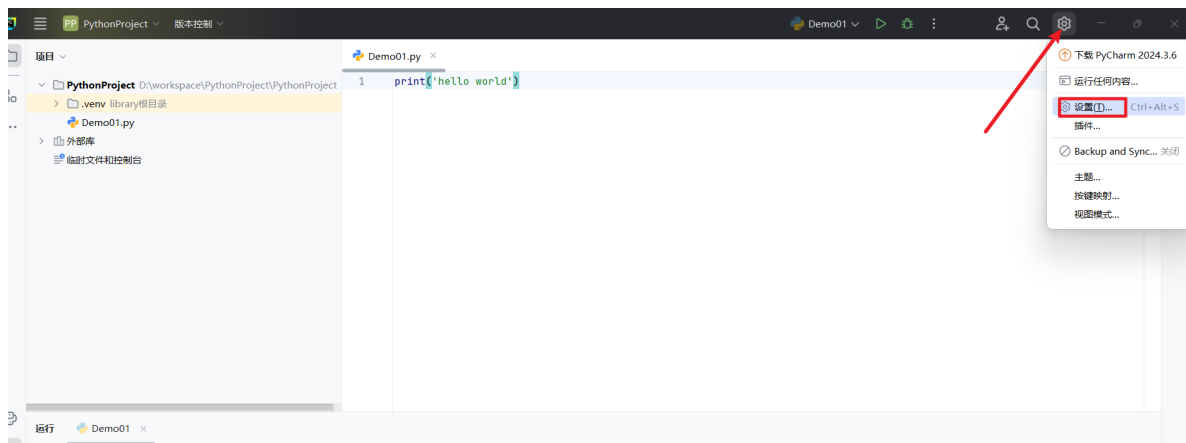




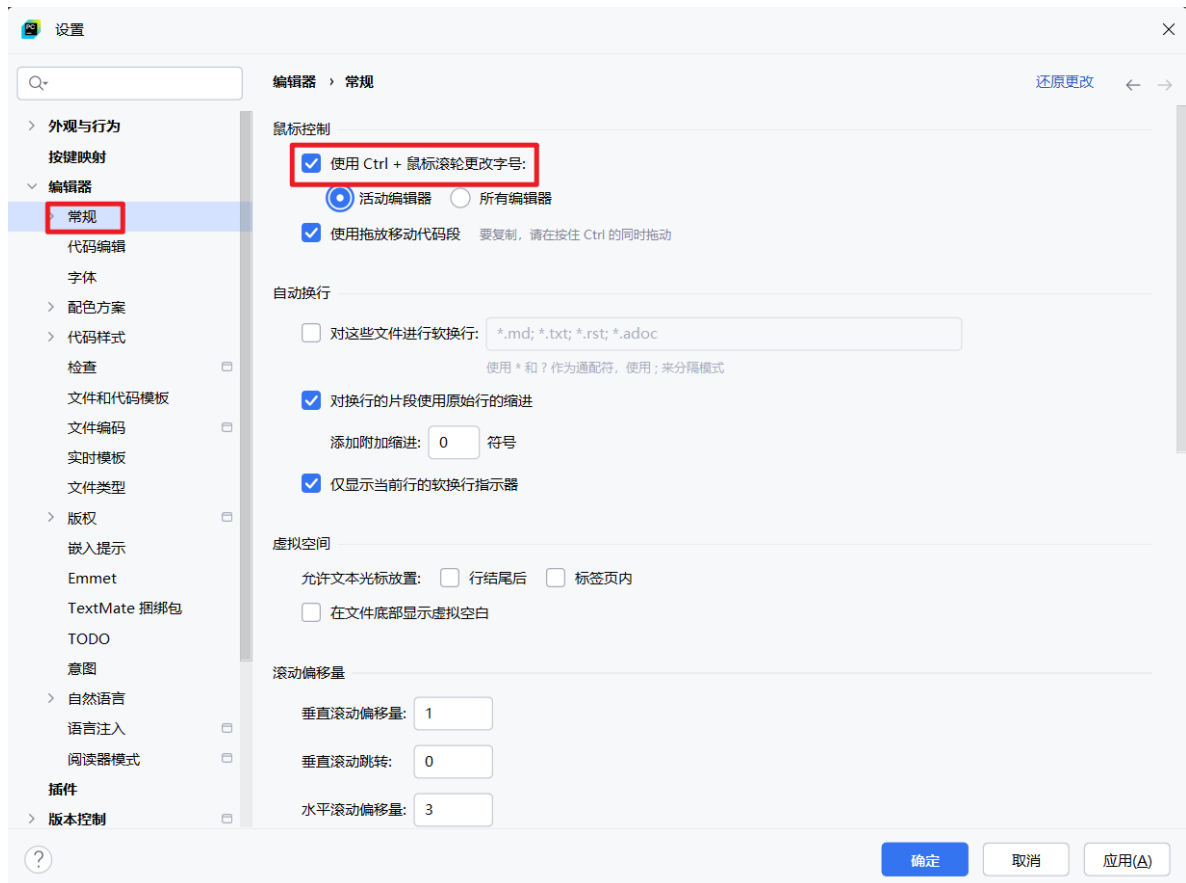
## 代码运行结果



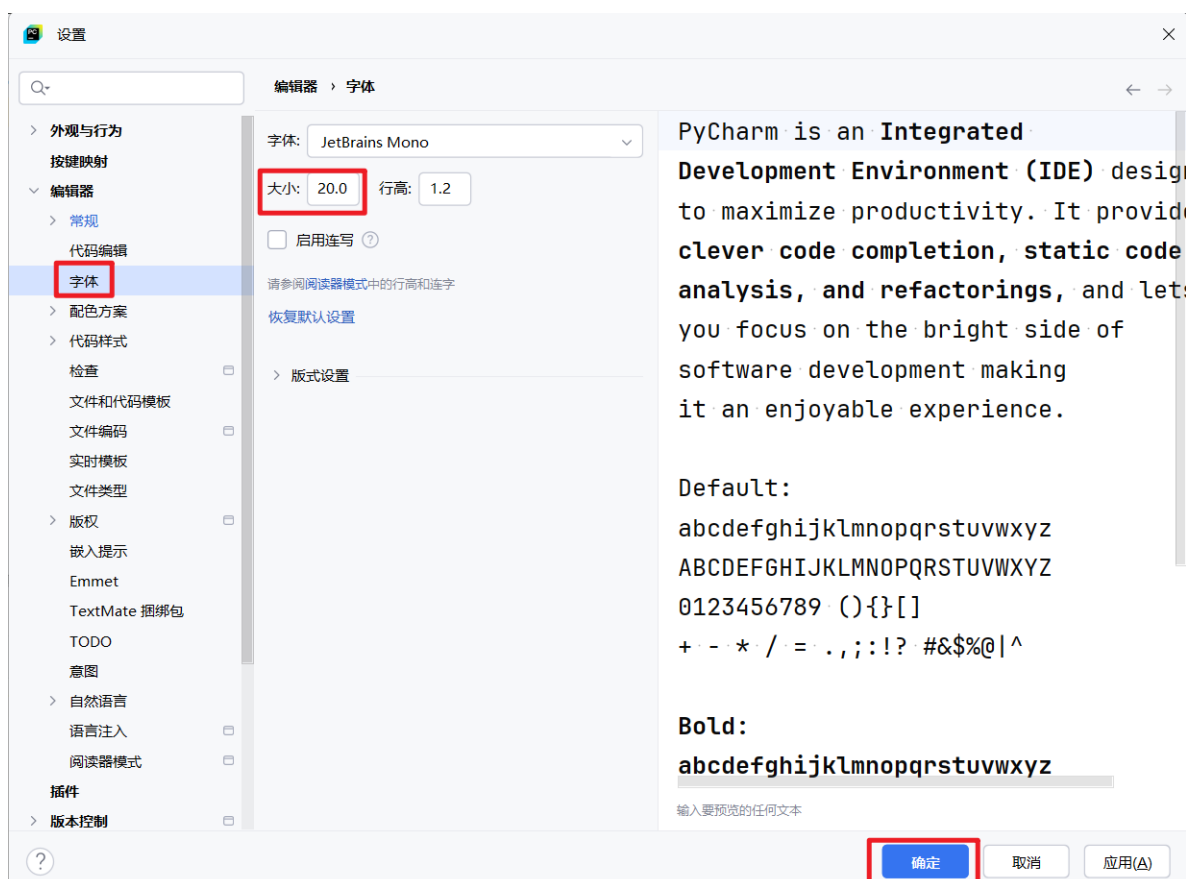
## 7、设置字体大小



## 选择使用滚轮+ctrl控制字体大小



## 设置默认字体大小



## 四、Python的注释

### 1、注释的作用

首先强调一件事：Python代码 => Python解析器 => 机器语言，但是注释经过了Python的解释器并不会解析与执行。因为其主要就是进行代码的注释。

注释作用： 提高代码的阅读性

### 2、注释的基本语法

#### 单行注释

单行注释，以"#" (Shift + 3)号开头，只能注释一行内容

```
# 注释内容
```

第一种：代码行的上面

```
# 输出Hello World字符串
print('Hello World')
```

第二种：放在代码的后面(代码后面保留2个空格)

```
print('Hello World') # 输出Hello World字符串
```

#### 多行注释

多行注释：3对引号（单引号或者双引号都行。实际工作推荐使用双引号）。可以同时注释多行代码或程序，常用于代码块的注释

```
"""
注释内容
第一行
第二行
第三行
"""
```

或

```
'''
注释内容
第一行
第二行
第三行
'''
```

示例代码：

```
"""
```

Python（意为大蟒蛇）是一款受欢迎的高级编程语言，以其简洁易读的语法和强大的功能著称。无论是初学者还是经验丰富的开发者，Python 都能满足您的需求。帮助您快速实现从简单脚本到复杂应用程序的开发。

```
"""
```

```
print('Hi, 大家好')  
print('让我们一起学习Python这门语言吧')
```

在PyCharm中，我们可以使用 **Ctrl + /斜杠** 来对代码或程序进行快速注释。

## 五、变量的使用

### 1、变量的概念

那什么是变量呢？

- ① 变量是存储数据的容器
- ② 变量在程序运行过程中是可以发生改变的量
- ③ 变量存储的数据是临时的

### 2、变量的定义

基本语法：

```
变量名称 = 变量的值  
name = "jack"
```

说明：在Python程序中，这个等号和日常生活中的等号不太一样，其有一个专业名词：赋值运算符，其读法：要从右向左读，把变量的值通过 = 赋值给左边的变量。

### 3、变量的命名规则

标识符命名规则是Python中定义变量名称时一种命名规范，具体如下：

- ① 由数字、字母、下划线(\_)组成
- ② 不能数字开头
- ③ 严格区分大小写
- ④ 不能使用内置关键字作为变量名称

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

注意：记不住Python关键字怎么办？答：借助于help()方法

```
help('keywords')
```

推荐变量的命名规则

- ① （强烈推荐）变量命名一定要做到见名知义。优先使用英文单词。
- ② 大驼峰：即每个单词首字母都大写，例如： `MyName` 。
- ③ 小驼峰：第二个（含）以后的单词首字母大写，例如： `myName` 。
- ④ 蛇形命名法（下划线）：例如： `my_name` 。

## 4、变量的定义与调用

在Python中，记住：变量一定要先定义，后使用，否则会报错。

定义：

```
name = 'jackson'
address = '湖北省武汉市洪山区'
```

调用：

```
print(name)
print(address)
或
print(name, address)

# 由于Python是一个解释性语言，因此变量的使用要在定义之前。所以下面的代码会报错
print(age)
age = 20
```

## 六、数据类型

### 1、数据类型分类

类型分类	数据类型	描述	示例
基本类型	整数 (int)	无小数部分的数字，支持任意大小	10、-5、0、999999999999
	浮点数 (float)	带小数部分的数字，支持科学计数法	3.14、-0.5、1e3（即1000）
	布尔值 (bool)	表示真/假，仅 True 和 False 两个值	True、False
	字符串 (str)	文本序列，用单引号、双引号或三引号包裹	'hello'、"Python"、'''多行文本'''
容器类型	列表 (list)	有序、可变的元素集合，元素类型可混合	[1, 'a', 3.14]
	元组 (tuple)	有序、不可变的元素集合，元素类型可混合	(1, 2, 3)、('a', 10.5)
	字典 (dict)	键值对集合（Python 3.7+ 有序），通过键快速访问值	{'name': 'Alice', 'age': 20}
	集合 (set)	无序、无重复元素的集合，适合去重和集合运算	{1, 2, 3}、set([2, 2, 3])
特殊类型	NoneType	仅包含 None，表示空值或无返回值	None

## 2、常用数据类型

### 数值类型

数值类型就是我们日常生活中的数字，数字又分为两种形式：整数 与 小数（带小数点）

整数类型：int类型

小数类型：float类型

案例1：定义一个人的年龄18岁

```
age = 18
print(type(age))
```

案例2：定义一个大白菜价格：3.5

```
price = 3.5
print(type(price))
```

## 布尔类型

布尔类型是与逻辑相关一种数据类型，只有两个值：True（真）与False（假）

案例：定义一个flag变量，其值为True

```
flag = True
print(flag)
print(type(flag))
```

## 字符串类型

在Python变量定义中，如果其赋值的内容是通过单引号或双引号引起来的内容就是字符串str类型。

```
msg = '这家伙很懒，什么都没有留下... '
print(type(msg))
```

# 七、数据类型转换

## 1、数据类型的转换方法

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

案例1：定义一个的数字，转换为整型



```
num = '10'
print(type(num)) #<class 'str'>

num = int(num)
print(type(num)) #<class 'int'>
```

### 案例2：其他类型的转换

```
# 1、整型转浮点类型 int => float
num1 = 10
print(float(num1))
print(type(float(num1)))

print('-' * 20)

# 2、浮点类型转换为整型 float => int, 浮点转整型, 其小数点后的数据会丢失!!!
num2 = 18.88
print(int(num2))

print('-' * 20)

# 3、把字符串类型转换为整型或浮点类型
str1 = '20'
str2 = '10.88'
print(type(int(str1)))
print(type(float(str2)))
```

### 案例3：eval()方法的使用，把字符串中的数字转换为原数据类型

```
price = '1.5'
print(eval(price))
print(type(eval(price)))

price = '2'
print(eval(price))
print(type(eval(price)))
```

注意事项： `int()`和`float()`在进行数据类型转换前，需要确保数据的内容是匹配。因此下面代码会报错

```
# 下面代码会报错
str = "abc"
result = int(str)
print(result, type(result))
```

## 八、Python的输入与输出

### 1、格式化输出

#### 1.1 百分号格式化

## 基本语法：

```
...
print(变量名称)
print('字符串%格式' % (变量名称))
print('字符串%格式 %格式 %格式' % (变量名称1, 变量名称2, 变量名称3))
```

## 格式常见形式如下：

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

案例1：定义两个变量name='jack', age=18，按照如下格式进行输出：我的名字是jack，今年18岁了。

```
name = 'jack'
age = 20
print('我的姓名为%s,年龄是%d' %(name, age))
```

案例2：定义两个变量title='大白菜', price=1.5，按照如下格式进行输出：今天大白菜只要1.50元/斤。

```
title = '大白菜'
price = 1.5
print('今天%s只要%.2f元/斤' % (title, price))
```

## 1.2 format方法格式化

### 基础语法

```
print('字符串{}'.format(变量名称1))
print('{}字符串{}'.format(变量名称1, 变量名称2))
```

案例1: 定义两个变量, name='jack', mobile='13845621547', 按照以下格式进行输出"姓名: jack, 联系方式: 13845621547"

```
name='jack'
mobile='13845621547'
print('姓名:{},联系方式:{}'.format(name,mobile))
```

### 1.3 format方法简写(推荐)

在Python3.6以后版本, 为了简化format输出操作, 引入了一个简写形式:

```
name='jack'
mobile='13845621547'
print(f'姓名:{name},联系方式:{mobile}')
```

扩展: 小数保留一定的小数位数, 保留小数位数的时候内部会进行四舍五入

格式 变量名称:.小数位数f

```
name = 'jack'
age = 20
score = 83.2203
print(f"我的名字是{name}, 今年{age}岁了, 考了{score:.2f}分")
```

### 1.4 输出换行

默认情况下, `print` 语句输出后会自动换行,可以通过 `end` 参数进行设置

```
print('hello',end=' ' )
print('world')
```

## 2、Python内容输入

在Python中, 如果想让Python程序接受用户的输入信息, 可以使用input()方法

基本语法:

```
变量名称 = input('提示信息..')
```

案例: 输入个人信息

```
name = input('请输入姓名')
age = input('请输入年龄')
sex = input('请输入性别')
print(f"姓名:{name}, 年龄:{age}, 性别:{sex}")
```

input()方法重要事项: 所有由input()方法获取的数据都是“字符串”类型

```
name = input('请输入您的姓名: ')
age = input('请输入您的年龄: ')

print(type(name)) # <class 'str'>
print(type(age)) # <class 'str'>
```

总结:

- ① input()可以用于接收由外部设备输入的信息，但是如果用户没有输入任何内容，则input()函数会中止当前代码的继续执行，处于等待状态，直到用户输入结束。
- ② 所有由input()方法获取的数据都是“字符串”类型

## 九、运算符

### 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	9 // 4 输出结果为 2
%	取余（取模）	9 % 4 输出结果为 1
**	幂指数	2 ** 4 输出结果为 16，即2的4次方，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

案例1：基本的加减乘除运算

```
num1 = 2
num2 = 10

# 四则运算 + - * /
print(f'加: {num1 + num2}')
print(f'减: {num1 - num2}')
print(f'乘: {num1 * num2}')
print(f'除: {num1 / num2}')
```

案例2：其他算术运算

```

num1 = 10
num2 = 3
num3 = 5

# 1、整除
print(f'整除: {num1 // num2}')
# 2、求余数
print(f'余数: {num1 % num2}')
# 3、幂指数
print(f'幂指数: {num2 ** 3}')
# 4、圆括号
print(f'优先级: {(num1 + num2) * num3}')
```

### 案例3：求三角形面积

```

a = print('请输入三角形的底')
b = print('请输入三角形的高')
s = a * b / 2

print(f'三角形的面积: {s}')
```

## 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
//=	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

### 案例1：= 赋值运算符使用方式

```

# 把某个值赋值给某个变量
num = 10
print(num)

# 多个变量同时进行赋值操作
n, f, s = 5, 10.88, 'hello world'
print(n, f, s)

# 多个变量赋予相同的值
a = b = 10
print(a, b)
```

## 案例2：其他赋值使用方式

```
a = 10
b = 12
# 使用a保存ab的总和
a += b # 相当于 a = a + b
print(a)
```

### 3、关系运算符

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

## 案例：两个数大小的比较

```
num1 = 10
num2 = 20

print(num1 > num2) # False
print(num1 < num2) # True
print(num1 >= num2) # False
print(num1 <= num2) # True
print(num1 == num2) # False
print(num1 != num2) # True
```

### 4、逻辑运算符

运算符	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。
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

案例1：输入变量a，判断a是否是一个偶数且大于10

```
a = int(input('请输入整数a'))  
print(a % 2 == 0 and a > 10)
```

案例2：a是否是一个奇数数或小于15

```
a = int(input('请输入整数a'))  
print(a % 2 != 0 or a < 15)
```

案例3：判断姓名不是'jack'

```
name = input('请输入姓名')  
print(not name == 'jack')
```