

安装 `python` 解释器和 `pycharm` 是学习python的第一步。安装 `python` 会我们提供开发 `python` 程序的工具,诸如 `python.exe` 解释器、`pip.exe` 包管理器等等工具,而 `pycharm` 是开发 `python` 所使用的集成开发环境,为我们提供诸如代码补全、智能提示等功能,使用 `pycharm` 可以让我们在开发 `python` 程序的时候如虎添翼,事半功倍。

1. `python` 的安装和配置

下载 `python`

本教程使用的是从官网下载的 `python` 版本

下载页面 <https://www.python.org/downloads/> , 打开之后会看到大量的python发行版本, 目前尚在维护更新的是 `3.6` 、 `3.7` 、 `3.8` 、 `3.9`

Looking for a specific release?
Python releases by version number:

Release version	Release date		Click for more
Python 3.9.4	April 4, 2021	Download	Release Notes
Python 3.8.9	April 2, 2021	Download	Release Notes
Python 3.9.2	Feb. 19, 2021	Download	Release Notes
Python 3.8.8	Feb. 19, 2021	Download	Release Notes
Python 3.6.13	Feb. 15, 2021	Download	Release Notes
Python 3.7.10	Feb. 15, 2021	Download	Release Notes
Python 3.8.7	Dec. 21, 2020	Download	Release Notes

选择最新的 `python` 版本, 此处以 `3.9.4` 版本为例, 点击 `download` 按钮, 然后进入新页面后下拉到底, 选择适合自己的 `操作系统` 版本

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		cc8507b3799ed4d8baa7534cd8d5b35f	25411523	SIG
XZ compressed source tarball	Source release		2a3dba5fc75b695c45cf1806156e1a97	18900304	SIG
macOS 64-bit Intel installer	Mac OS X	for macOS 10.9 and later	2b974bfd787f941fb8f0b5b8084e569	29866341	SIG
macOS 64-bit universal2 installer	Mac OS X	for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental)	9aa68872b9582c6c71151d5dd4f5ebca	37648771	SIG
Windows embeddable package (32-bit)	Windows		b4bd8ec0891891158000c6844222014d	7580762	SIG
Windows embeddable package (64-bit)	Windows		5c34eb7e79cfe8a92bf56b5168a459f4	8419530	SIG
Windows help file	Windows		aaacfe224768b5e4aa7583c12af68fb0	8859759	SIG
Windows installer (32-bit)	Windows		b790fdaff648f757bf0f233e4d05c053	27222976	SIG
Windows installer (64-bit)	Windows	Recommended	ebc65aaa142b1d6de450ce241c50e61c	28323440	SIG


本教程以 `window` 平台 `64位` 版本为例, 下载 [Windows installer \(64-bit\)](#) `安装器` (installer) 至本地

- TIPS: 下载 `python` 有两种途径:
1. 在 `python` 的官网下载

2. 下载 `anaconda` 版本的 `python`

相较于官方版本，`anaconda` 版本的 `python` 自带了很多的数据分析方面的第三库，适合数据分析师，但是体积庞大，需要超过 `5GB` 的硬盘空间。而是用官方版本的 `python` 可以获得最小环境，大约只占用 `150MB` 硬盘空间

安装 `python`

 `python-3.9.4-amd64.exe`

下载后的文件名为 `python-3.9.4-amd64.exe`，双击运行 `安装器`，

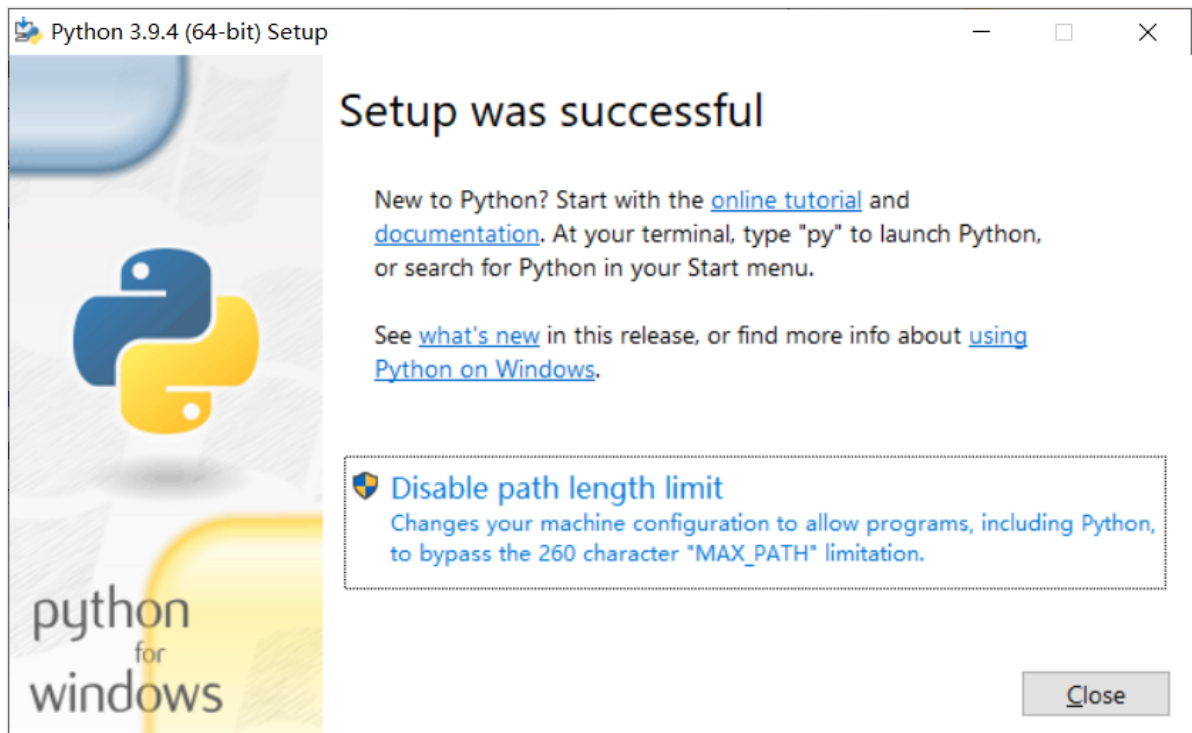


勾选下方的 `Add Python 3.9 to PATH`，此时就不需要我们单独配置环境变量了，并点击 `Install Now`，就会开始为我们安装python。

TIPS:

安装Python需要点击的是 `Install Now`，而不是 `Cancel`，`Cancel` 表示取消安装。

如果安装 python 的时候，没有勾选 `Add Python 3.9 to PATH` 就需要自己手动配置环境变量，配置方式下面会提到。



看到 **Setup was successful** 表示python已经被正确的安装到了我们的环境中，接下来打开 **powershell** 或者 **CMD**，输入 **python**，可以看到 **python3.9.4** 已经被终端所识别。并且输出的 **python** 版本信息和你下载的版本信息相同的话，你已经完美的完成了 **python** 的安装，可以跳到第二部分查看关于安装 **pycharm** 的事宜。

```
PS C:\Users\27> python
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

但是，如果你输入 **python** 后输出的版本信息和下载的版本不一致，这说明你在这之前安装了其他版本的 **python**。那么你需要好好看看下面关于配置环境变量的那些事了。

又或者在安装 **python** 的时候没有勾选 **Add Python 3.9 to PATH**，导致输出 **无法将“python”项识别为 cmdlet、函数、脚本文件或可运行程序的名称。请检查名称的拼写，如果包括路径，请确保路径正确，然后再试一次。** 或者跳转到了 **微软商店**，那也需要自己手动配置环境变量。（也可以卸载后重新启动安装器，勾选勾选 **Add Python 3.9 to PATH** 重装 **python**）

cmd 和 **powershell** 都是 **windows** 下的终端。

打开 **cmd** 的方式为同时按下 **win+r** 键，并输入 **cmd** 后回车，便会看到一个黑框框，这就是 **cmd**。
cmd 是 **command** 的缩写，是一个提供用户使用计算机的虚拟终端。

打开 **powershell** 的方式为，鼠标右键电脑左下角的win图标，在弹出的列表中选择 **windows powershell**，打开即可，在 **windows10** 上，**powershell** 的默认样式是蓝底黑字，颇具年代感，但它是现代化版本的 **cmd**，提供了更加强的功能和更丰富的命令

在终端输入 **python** 命令，便会进入 **python** 的 **交互模式**，在这个模式下，便可进行python代码的编写，要退出 **交互模式**，可以输入 **exit()** 退出交互模式

配置环境变量

不管是安装 `python` 还是安装 `java`，甚至是安装 `mysql`、`maven`、`git` 等等工具都离不开环境变量的配置，因此理解并学会如何配置环境变量是程序员的必修课。

什么是环境变量

环境变量 (environment variables) 一般是指在操作系统中用来指定操作系统运行环境的一些参数。

使用 `Get-ChildItem env:` 命令查看当前系统中的所有环境变量

```
PS C:\Users\27> Get-ChildItem env:

Name                           Value
----                           -
...
OneDrive                       C:\Users\27\OneDrive
OneDriveConsumer               C:\Users\27\OneDrive
OS                             Windows_NT
Path                           C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\wbem;C:\WINDOWS\System32\wind
owsPowerShell\v1.0\;C:\WINDOWS\System32\O...
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.CPL
PROCESSOR_ARCHITECTURE        AMD64
PROCESSOR_IDENTIFIER           AMD64 Family 23 Model 24 Stepping 1, AuthenticAMD
...
```

我们在安装 `python` 的时候需要关注的一个环境变量叫做 `PATH`，这个变量至关重要，`PATH` 变量的值是一个字符串，保存了很多的路径，用 ; 分号隔开,使用 `Get-ChildItem env:PATH | fl` 可以查看完整 `PATH` 变量的值。

```
PS C:\Users\27> Get-ChildItem env:PATH | fl

Name : Path
Value : C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WINDOWS\System32\OpenSSH\;C:\Program Files\I
ntel\WiFi\bin\;C:\Program Files\Common Files\Intel\WirelessCommon\;C:\Program Files\Git\cmd\;C:\Program Files\MySQL\mysql-8.0.22\bin\;C:\HashiCorp\Va
grant\bin\;C:\Program Files\Java\jdk-11.0.10\bin\;C:\Program Files\Apache\apache-maven-3.6.3\bin\;C:\Users\27\AppData\Local\Programs\Python\Python39\Sc
ripts\;C:\Users\27\AppData\Local\Programs\Python\Python39\;C:\Users\27\cargo\bin\;C:\Users\27\AppData\Local\Programs\Python\Python38\Scripts\;C:\U
sers\27\AppData\Local\Programs\Python\Python38\;C:\Users\27\AppData\Local\Microsoft\WindowsApps\;C:\Program Files\JetBrains\PyCharm 2020.3.3\bin\;C:\U
sers\27\AppData\Local\Programs\Microsoft VS Code\bin\;C:\Program Files\JetBrains\CLion 2020.3.3\bin\;C:\Program Files\JetBrains\IntelliJ IDEA 2020.3
.3\bin
```

我们在终端输入 `python` 的时候，会在系统变量 `PATH` 下的文件夹中搜索 `python.exe` 文件。如上图所示,首先会在 `C:\WINDOWS\system32` 文件夹下搜索是否 `python.exe`，如果有则调用 `C:\WINDOWS\system32` 文件夹下的 `python.exe` 文件，如果没有，则去下一项搜索，重复上述步骤。所有 `PATH` 值下的所有文件夹都没有 `python.exe` 就会报错。

例如输入 `run`，当搜索所有 `PATH` 变量下的文件夹都没有 `run.exe`，就会报错 无法将“run”项识别为 `cmdlet`、函数、脚本文件或可运行程序的名称。

```
PS C:\Users\27> run
run : 无法将“run”项识别为 cmdlet、函数、脚本文件或可运行程序的名称。请检查名称的拼写，如果包括路径，请确保路径正确，然后再试一次。
所在位置 行:1 字符: 1
+ run
+ ~~~
+ CategoryInfo          : ObjectNotFound: (run:String) [],
CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

什么情况下需要配置环境变量

pycharm、vscode 这些编写python程序的软件就会通过 PATH 变量感知计算机中是否安装了python，如果在PATH变量的所有文件夹中都找不到python.exe，便会认为系统没有安装python解释器。

因此为了可以让我们的pycharm可以识别到我们安装在计算机中的python程序，就需要将python.exe所在的文件夹添加到PATH中，不仅仅要添加到PATH变量中，位置要尽量靠前，如果你的电脑中，安装了多个python，最前位置的python将被优先识别。

当我们在只安装了一个python版本的时候，并且安装时勾选了Add Python 3.9 to PATH，你的环境变量也不见得会正确。

正如安装python时候预留的两个问题。

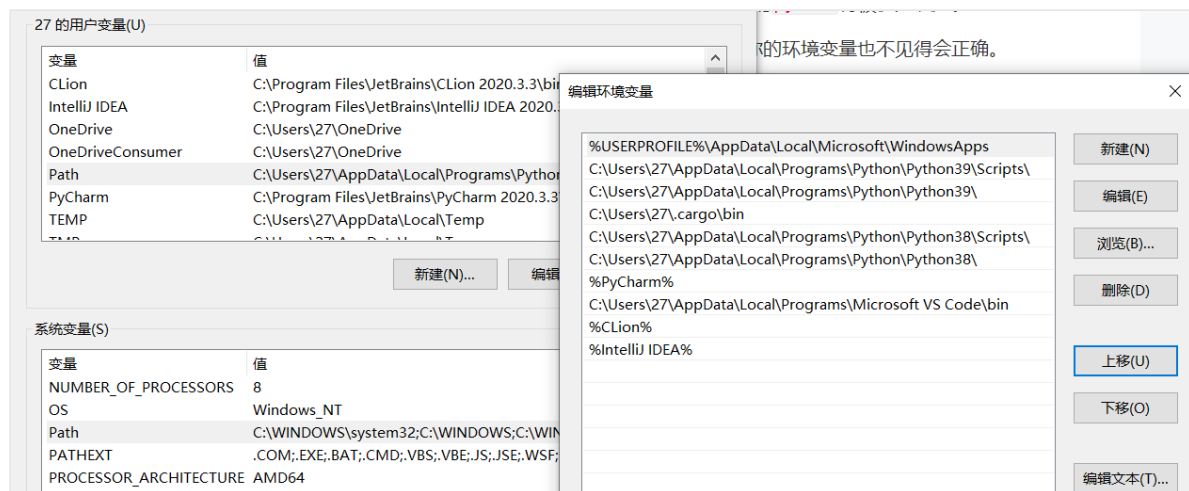
- 输入python后跳到了微软商店
- 输出的版本信息和下载的版本不一致

TIPS：环境变量分为用户变量和系统变量，系统变量的优先级高于用户变量

跳转到微软商店的原因

万恶的微软为了让我们多使用微软商店，在用户变量的PATH中留了一个%USERPROFILE%\AppData\Local\Microsoft\WindowsApps路径，并在该路径下放置了一个python.exe文件，如果你手动添加地python环境变量至用户变量，并且记录在%USERPROFILE%\AppData\Local\Microsoft\WindowsApps下面，那么就会出现优先识别到假python.exe而跳转到微软商店。

解决办法：将%USERPROFILE%\AppData\Local\Microsoft\WindowsApps下移到最低下。此做法无任何副作用。



安装版本和输出信息版本不一致的原因

因为电脑中存在多个python版本（在你依据本教程安装python之前的，已经自行安装过python，例如在pycharm中被pycharm好友关照了，或者去微软商店安装了商店python，又或者你安装了anaconda版本的python），而这些“早已安装”的python在PATH变量中的位置比我们刚刚安装的靠前，根据PATH搜索文件的“短路原则”，其他的python.exe被调用。

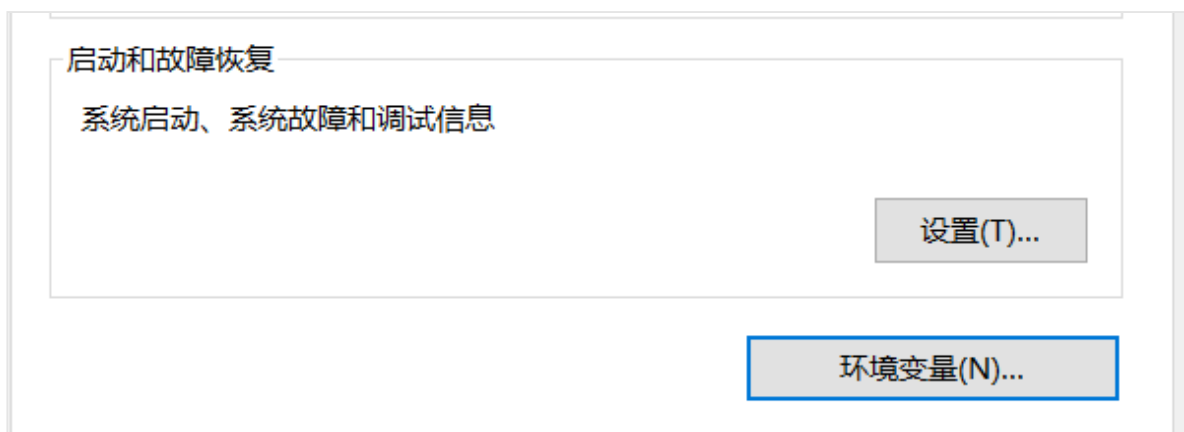
解决办法：依据下文介绍的配置环境变量方法，调整（上移）我们新python路径在PATH当中的位置，盖过那些“早已安装”的python

如何配置环境变量

步骤一：按下 **win+s** 键，在搜索框中输入 **环境变量** 四个字（必须完整输入这四个字），点击 **编辑系统环境变量**。



步骤二：点击 **环境变量**



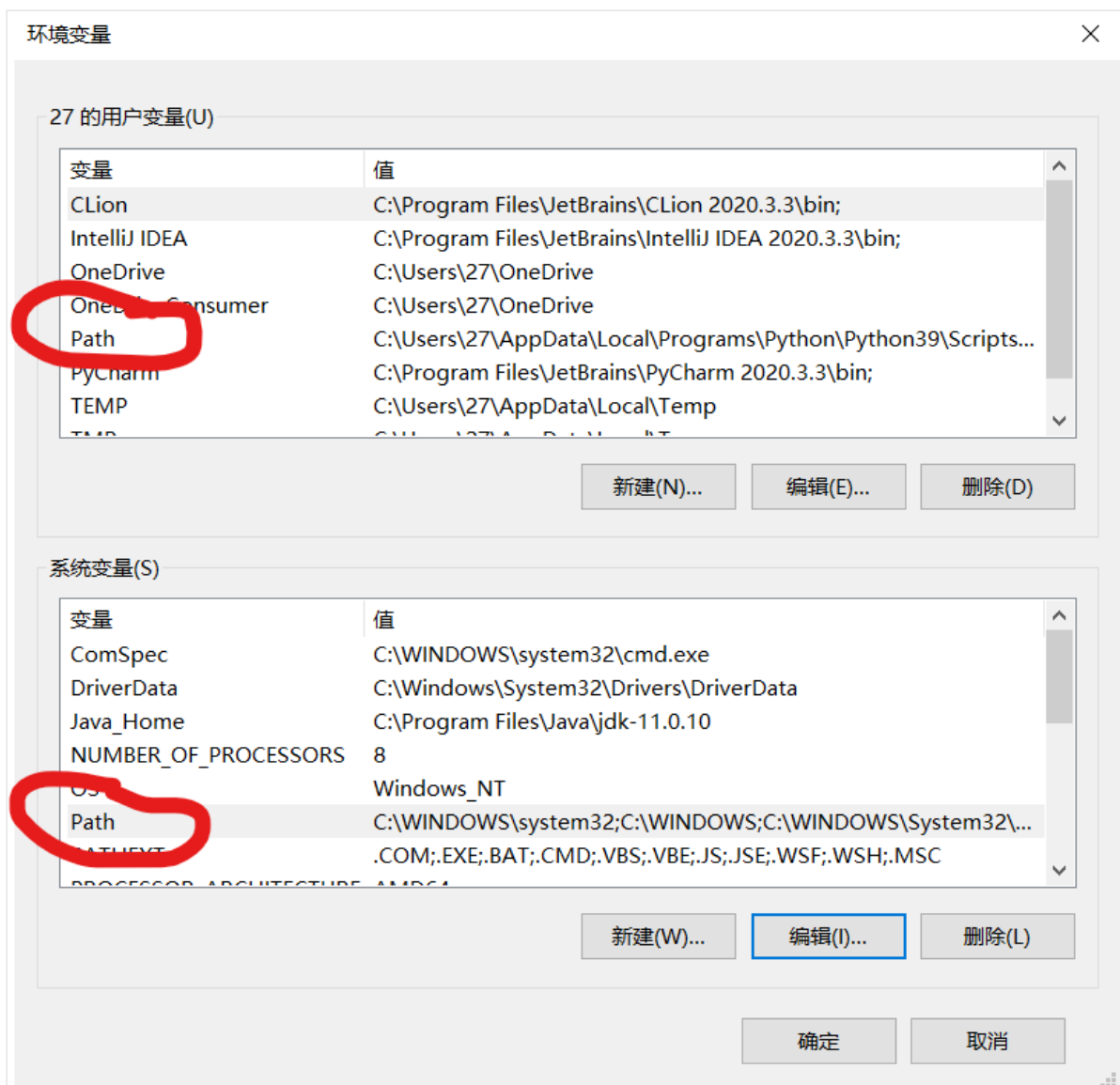
步骤三：点击 **系统变量** 的 **PATH** 或者 **用户变量** 的 **PATH**

系统变量 和 **用户变量** 的 **PATH** 都会被合成为环境变量的 **PATH**，但是 **系统变量** 的路径都排在 **用户变量** 前面，拥有更高的优先级。

系统变量 是对所有登录该计算机的用户可见的，而 **用户变量** 只对设置此变量的用户可见。

装载 **windows** 操作系统的电脑被称为PC，但是 **windows** 系统是支持多用户的，这就也导致，可以在 **windows** 上登录不同用户的账号，不同用户的用户变量是不同的，是相互隔离的。

例如，小明在电脑A中以xiaoming的身份登录，并设置了一个a=123作为 **系统变量**，b=456作为 **用户变量**。然后小明从电脑A退出xiaoming账户，小王以xiaowang的电脑A继续登录，此时小王看不到小明设置的b=456 **用户变量**，但可以看到a=123这个 **系统变量**。



以使用 用户变量 为例，点击 用户变量 下的PATH，点击新建，分别添加下面两行代码，并点击右侧的 上移，将这两条记录 置顶。

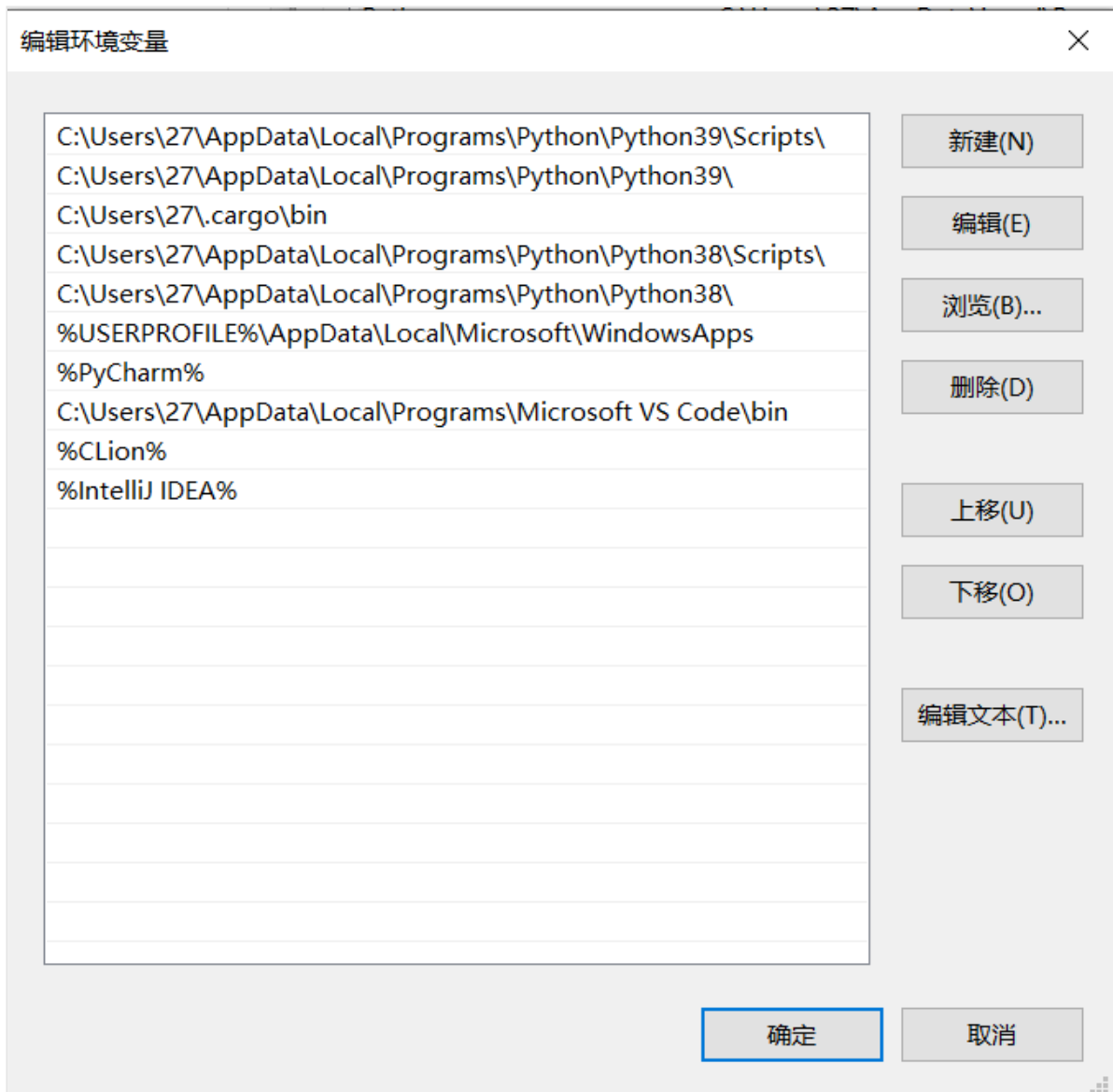
```
C:\Users\27\AppData\Local\Programs\Python\Python39\
```

```
C:\Users\27\AppData\Local\Programs\Python\Python39\Scripts\
```

Python39 文件夹中包含了 python.exe 文件，而 Python\Script 包含了 pip.exe 等脚本命令，缺一不可。

TIPS: 如果下载的python版本与教程中的版本不一致，请不要直接copy上面的两行代码，要根据实际位置进行修改。

配置完成后如下图所示。

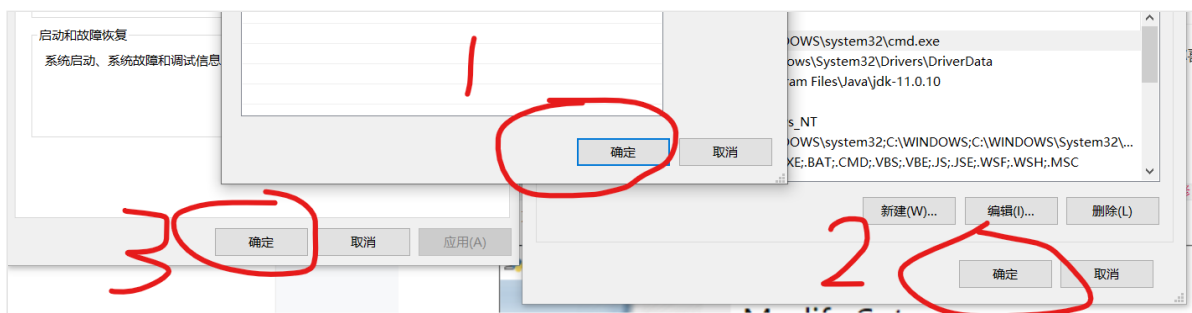


如果找到python.exe所在的路径:

通过python官方的 [安装器](#) 安装的python的默认位置是在

`C:\Users\27\AppData\Local\Programs\Python\` 文件夹下


连续点击三个 **确定**，保存我们的环境变量的修改,此时环境变量便修改成功。



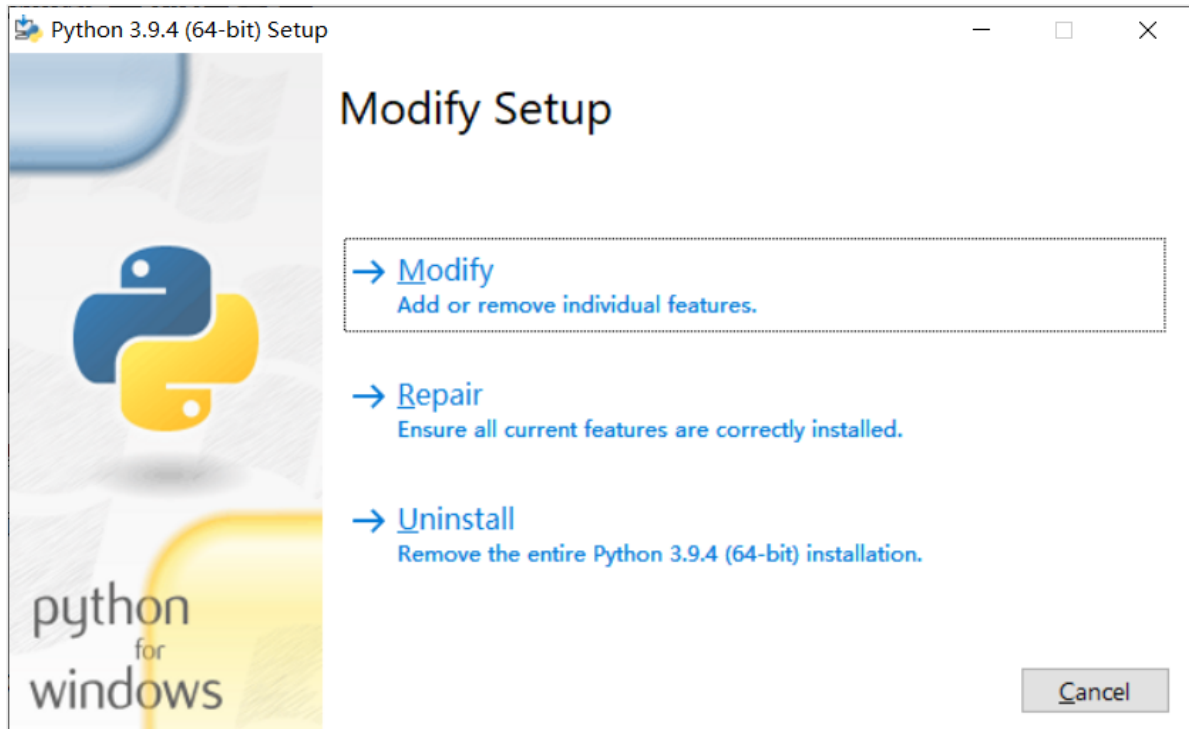
卸载 python

如果你需要更换你操作系统中的 `python` 版本，可以先卸载原有 `python`，在重复上面的步骤，找到你喜欢的版本并安装它。

下面介绍卸载方法

 python-3.9.4-amd64.exe

再次打开 `python-3.9.4-amd64.exe`，这时和安装python时候的界面就不一样了，三个功能分别是 **修改**、**修复**、**卸载**，此时点击 `uninstall` 就可以完全卸载python。




2. `pycharm` 的安装和配置

`pycharm` 是 `python` 开发工程师的首选，一款好用的工具往往可以使得事半功倍,可以帮助我们进行Web后端、网络爬虫、数据分析等方面的开发。

版本选择

`pycharm` 提供了收费的 **专业版** 和免费的 **社区版**，专业版本提供了对 `Django` 框架的支持，在编写 `Django` 应用的时候，拥有更好的智能补全以及模板语法的代码提示。因此我们选择专业版

TIPS：即便没有付费，也可以试用使用 `pycharm` 专业版本30天



Version: 2021.1
Build: 211.6693.115
7 April 2021

[System requirements](#)
[Installation Instructions](#)
[Other versions](#)

Download PyCharm

[Windows](#) [macOS](#) [Linux](#)

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)

Free trial

Community

For pure Python development


[Download](#)

Free, open-source

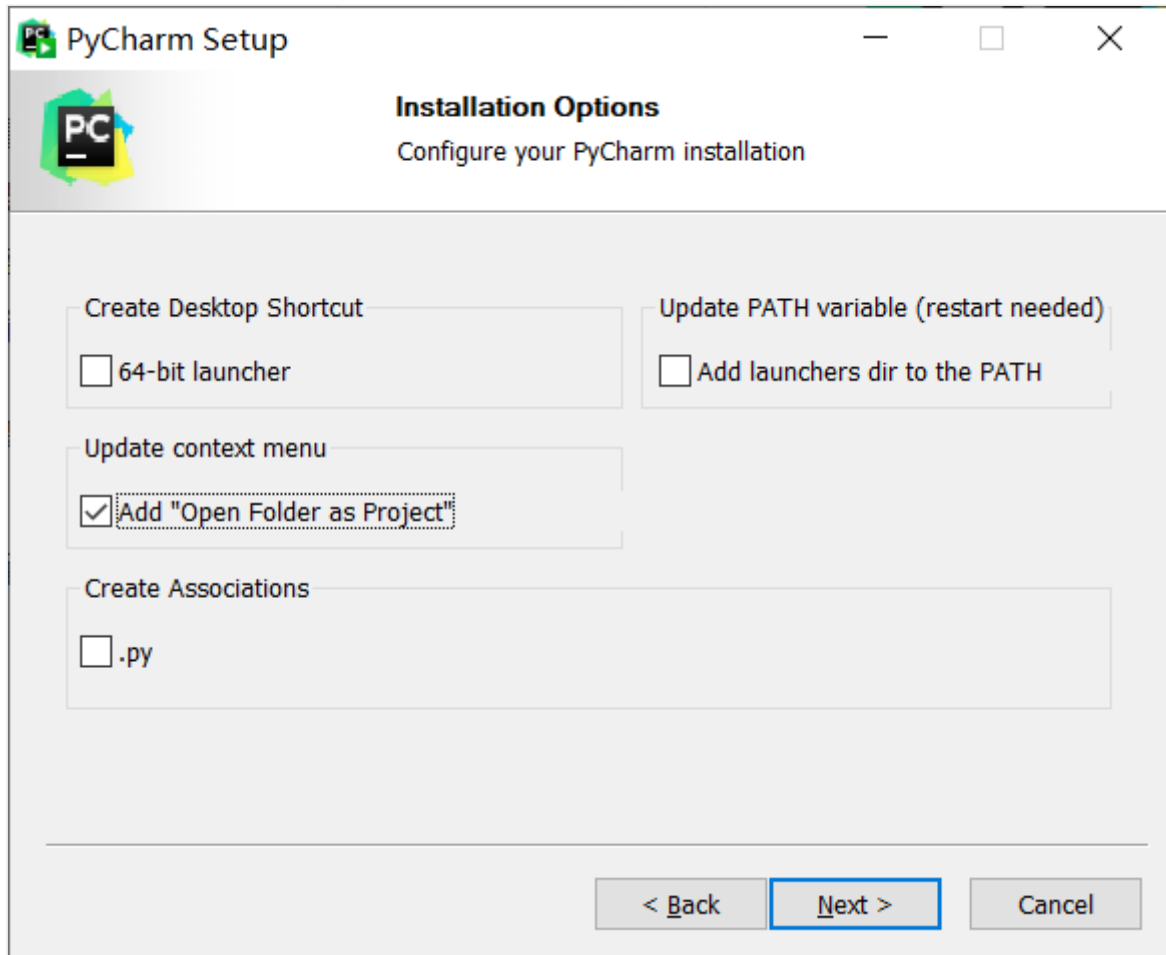
下载 `pycharm`

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

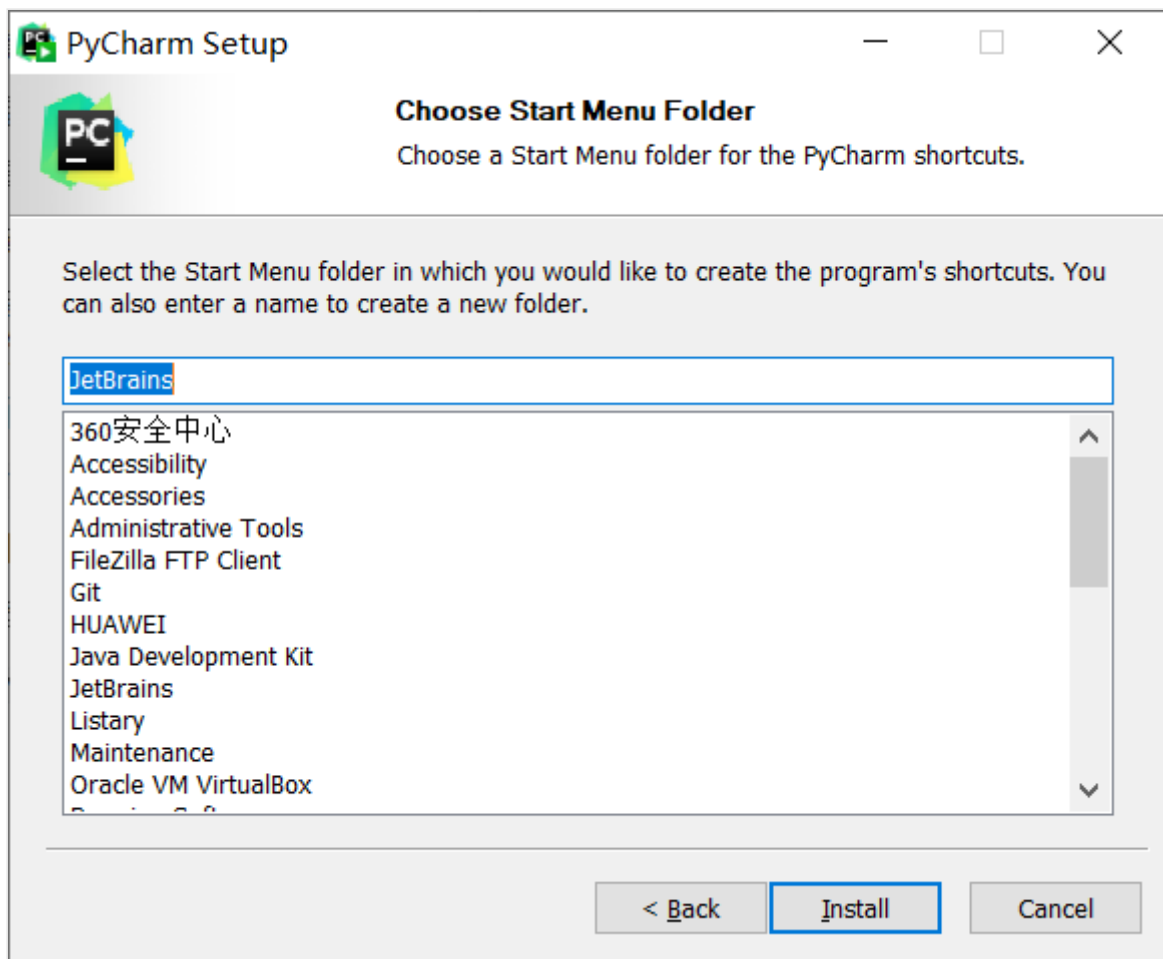
点击 **Download** 按钮便可进行下载

 **pycharm-professional-2021.1.exe**

双击便可进行安装, 一路 **next** 直到下图所示界面, 不要忘记勾选 **Add "Open Folder as Project"**, 这个功能可以使我们在文件夹 (**项目**) 上通过鼠标右键打开改文件夹 (**项目**)。



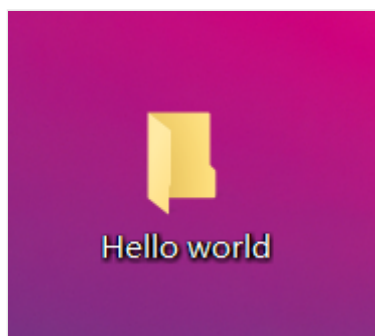
然后继续 **next**, 点击 **install** 开始安装 **pycharm**。



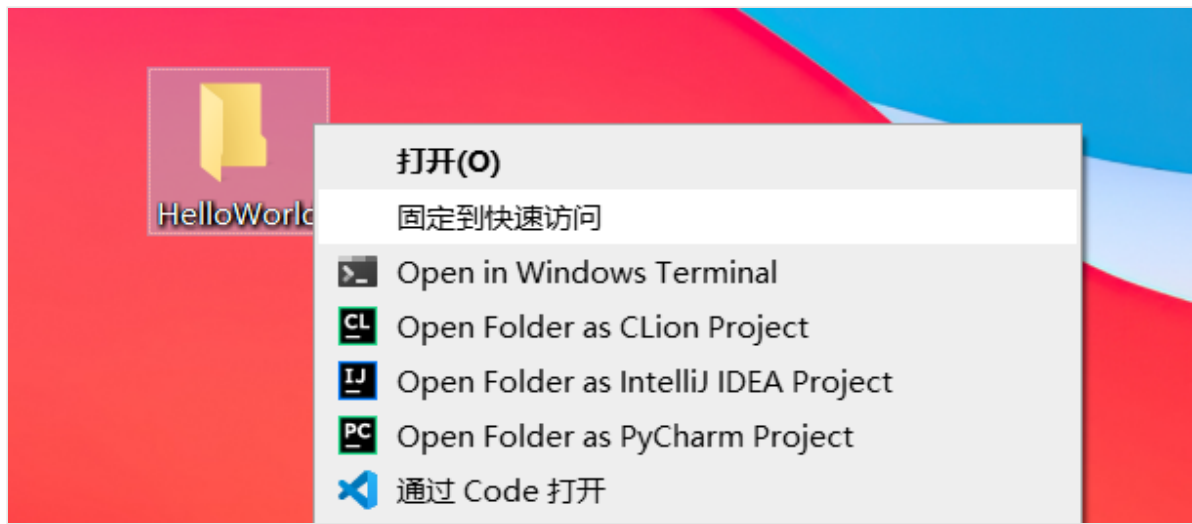
等待 `pycharm` 安装完成后，点击 `finish` 关闭安装窗口，此时先不要打开 `pycharm`。

创建第一个项目并运行

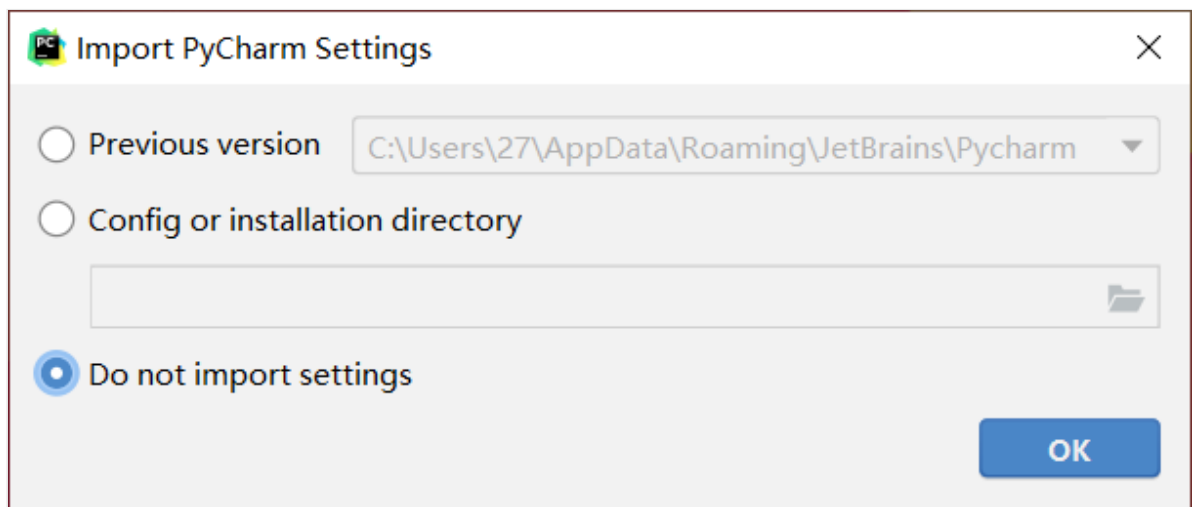
步骤一： 在桌面新建一个文件夹，并命名为 `Helloworld`



步骤二： 鼠标右键 `Helloworld` 文件夹，并选择 `Open Folder as Project` 打开。



步骤二：选择 `Do not import settings` ,然后点击 `ok`

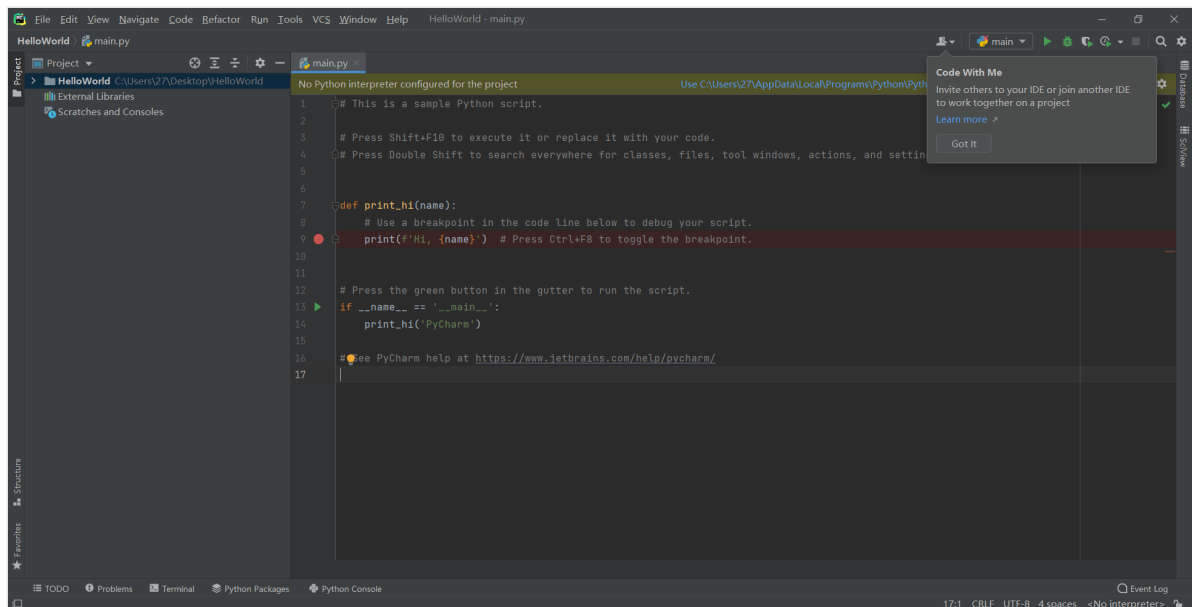


在随后弹出的窗口中选择 `试用` , 不需要登录账号等操作, 便可直接进入 `pycharm` 界面

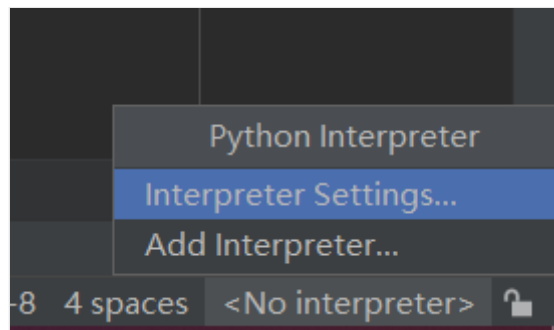
步骤三：配置解释器

这一步很关键, 因为想要 `pycharm` 运行 `python` 代码, 就需要告诉 `pycharm` `python.exe` 在哪里。

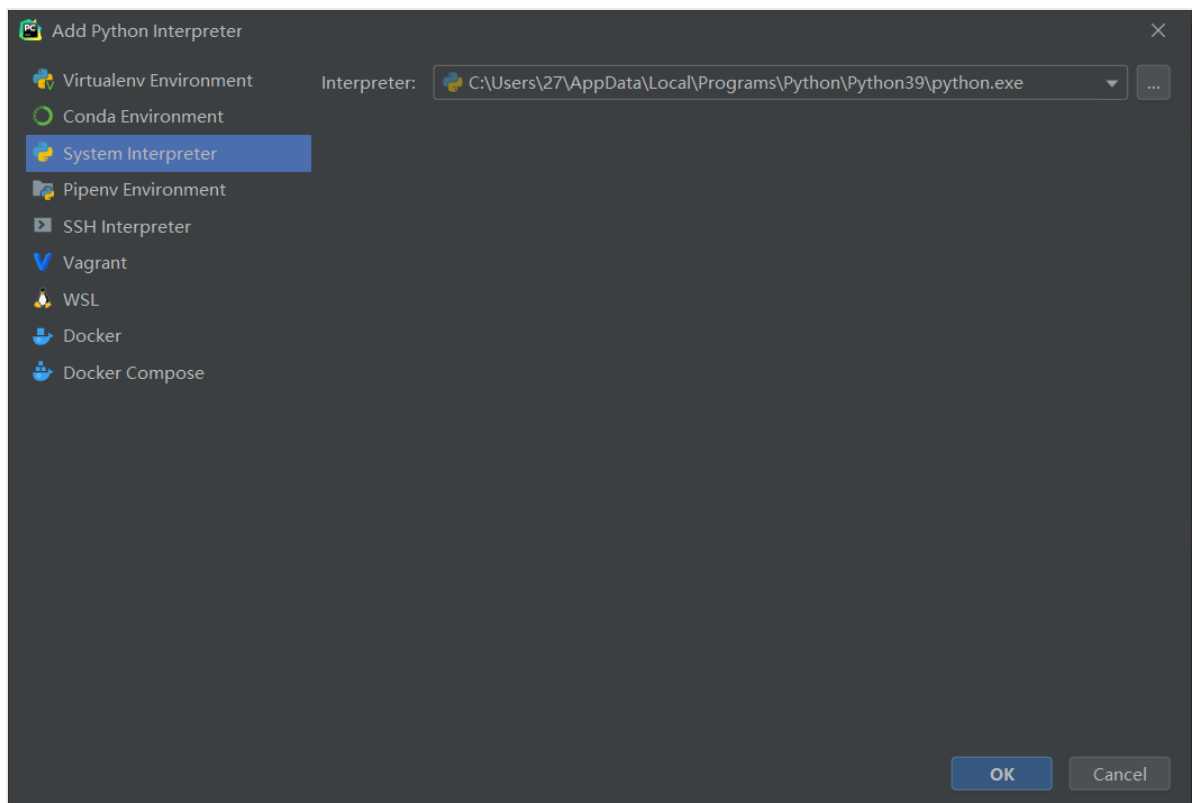
进入 `pycharm` 的界面如下图所示, 右下角显示 `<No interpreter>`。因为之前设置过环境变量, 这个时候 `pycharm` 其实是知道了我们之前安装的 `python.exe` 在哪里, 但是 `pycharm` 支持非常丰富的 `interpreter` 支持, 所以还需要我们手动告诉 `pycharm` , 我需要用到的 `python.exe` 就是刚刚安装的。



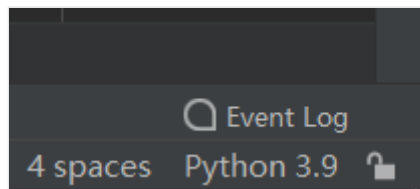
点击右下角的 **<No interpreter>**，再点击 **Add interpreter**



再弹出的窗口中选择 **System interpreter**，可以看到，**pycharm** 自动发现了我们的 **python3.9**，点击 **OK** 按钮即可



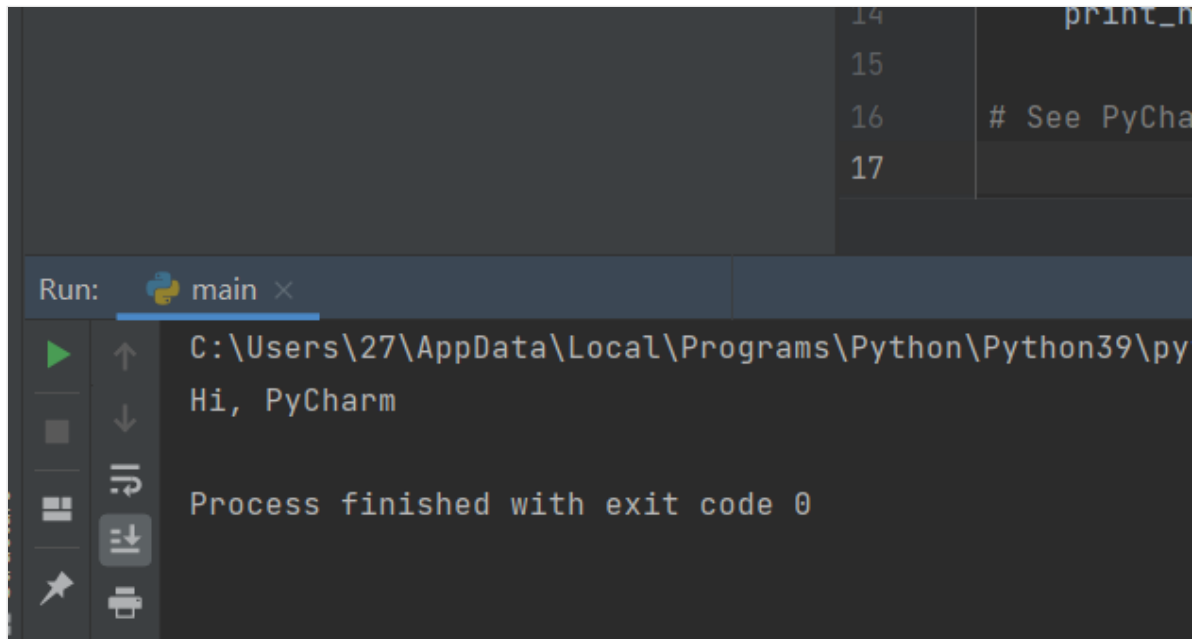
此时，之前的 **<No interpreter>** 就变成了 **python3.9** 了



运行python代码

在打开 `HelloWorld` 文件夹的时候，`pycharm` 为我们自动创建了一个名为 `main.py` 脚本，此时我们便可直接运行这个脚本，来领略一下 `python` 的魅力。

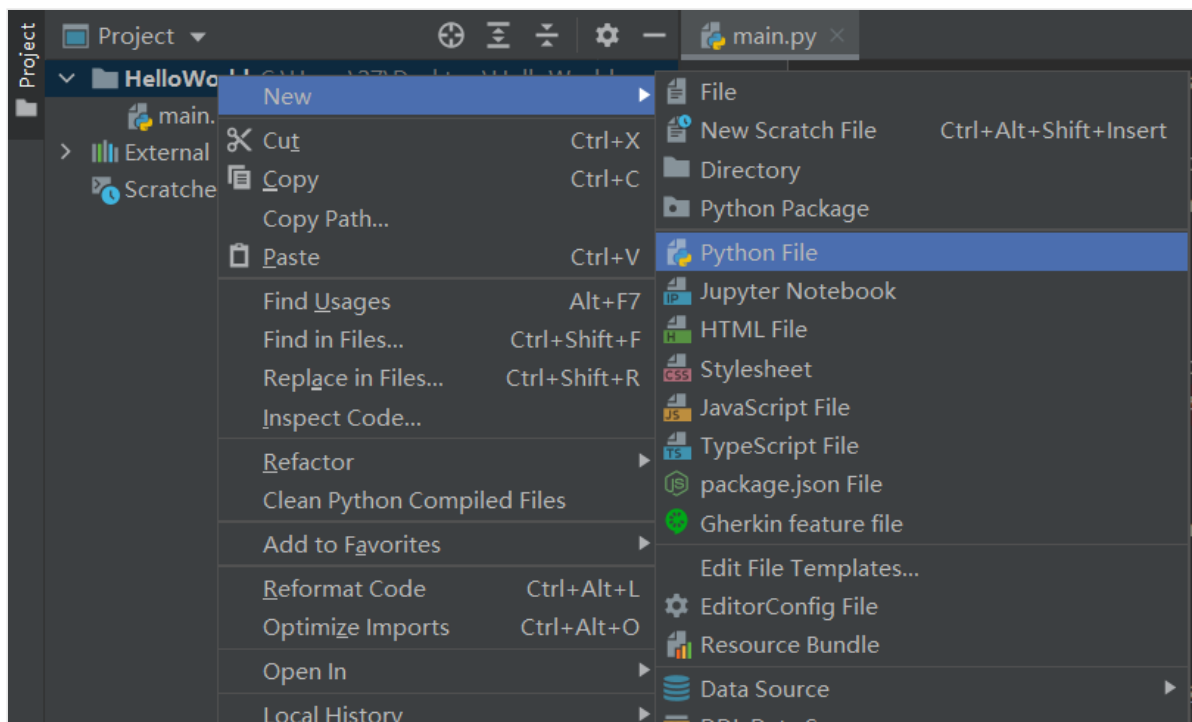
使用输入快捷键 `ctrl+shift+F10` 便可以运行，输出如下图所示。



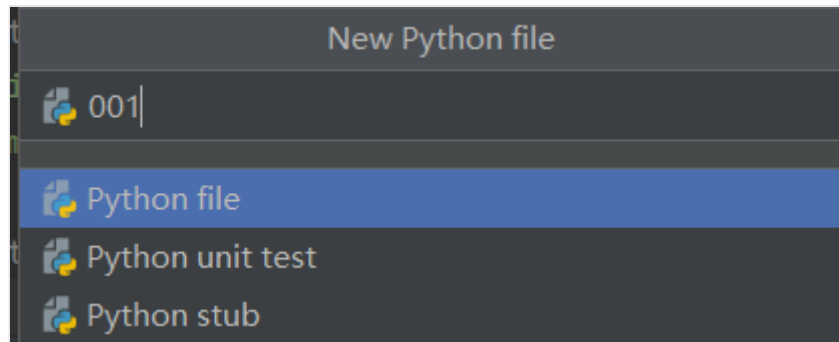
创建并运行自己的python脚本

接下来，我们用 `pycharm` 为我们自己创建脚本

在 `HelloWorld` 右键，鼠标悬停于 `New`，并点击 `Python File`

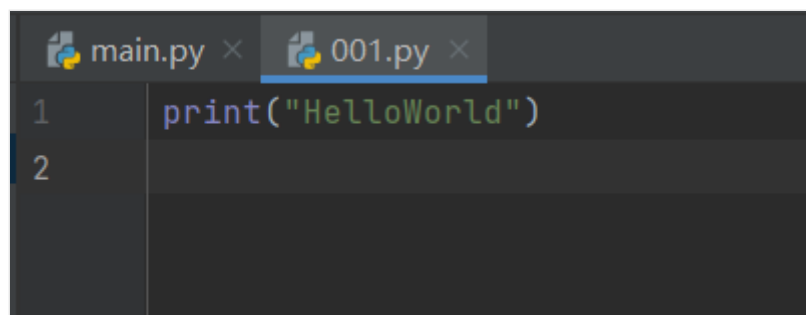


输入001并回车



在新的文件001.py中输入如下语句,并使用快捷键 `ctrl+shift+F10` 运行该脚本

```
print("HelloWorld")
```



便可获取输出


```
001 x
C:\Users\27\AppData\Local\Programs\Python\Python39\python.exe
HelloWorld

Process finished with exit code 0
```

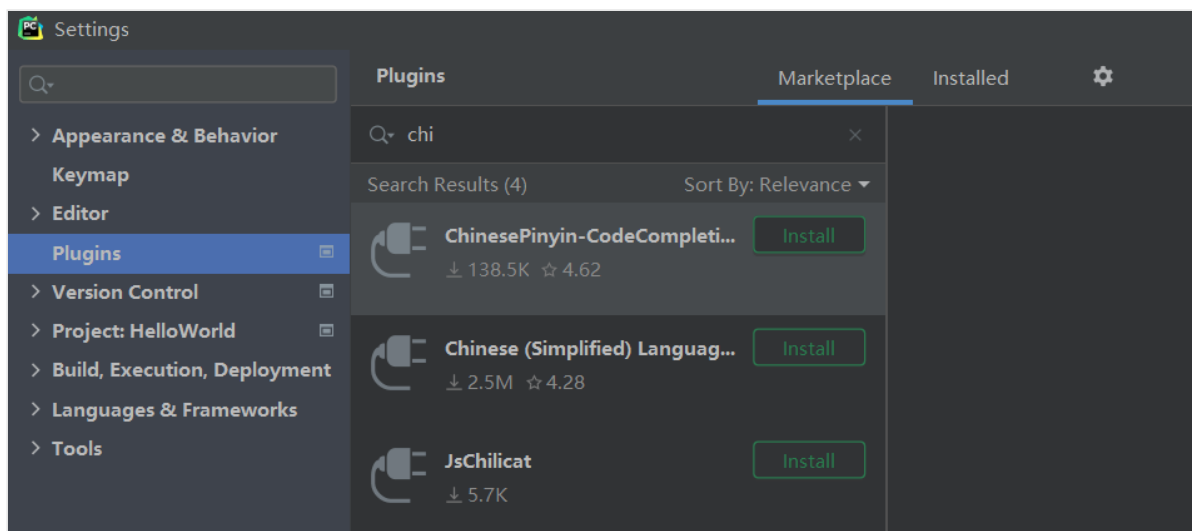
插件推荐

中文插件

如果你的英文不太好，又想快速上手pycharm，但又担心乱点导致一些乱七八糟的情况，那你可以尝试安装中文插件。

安装方式

点击 **pycharm** 主界面左上角 **file** -> **settings** -> **plugins**，在搜索框输入 **chi**，安装第二个2.5M下载量的插件（此为官网插件），选择 **Accept**，安装完成后，需要点击 **Restart IDE** 按钮 重启 pycharm即可。



彩虹括号

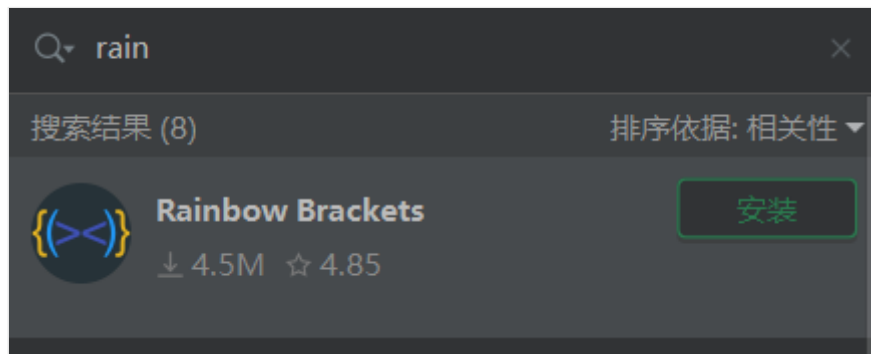
下面的代码充斥了很多的圆括号和方括号，让人看了头疼，需要花时间看左括号是和哪个右括号匹配

```
tweet.refresh_from_db()
self.assertEqual(tweet.likes_count, 3)
response = self.dongxie_client.get(tweet_url)
self.assertEqual(response.data['likes_count'], 3)
response = self.linghu_client.get(newsfeed_url)
self.assertEqual(response.data['results'][0]['tweet']['likes_count'], 3)
response = self.dongxie_client.get(newsfeed_url)
self.assertEqual(response.data['results'][0]['tweet']['likes_count'], 3)
```

此时，如果有一个彩虹括号，通过不同的颜色来区分不同的括号对，那将大大方便我们阅读代码。

安装方式

点击 `pycharm` 主界面左上角 `file` -> `settings` -> `plugins`，在搜索框输入 `rain`，安装 `Rainbow Brackets` 插件。

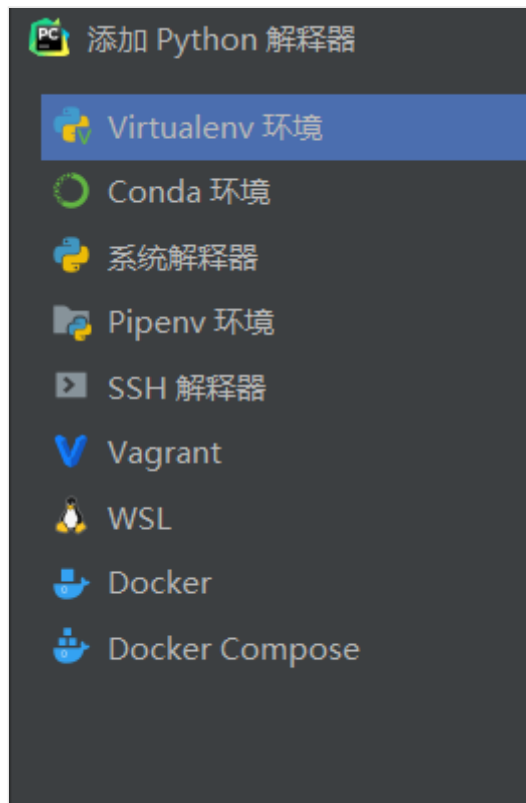


为 `pycharm` 选择不同的解释器

真实的开发很难使用一个解释器变打遍天下。

可能你接受了一个公司的老项目，用的还是 `python3.5`，而今天刚刚立项的新项目打算使用 `python3.9`

另外还有的项目跑在类似 `docker` 容器，又或者需要我们使用 `ssh`、`vagrant` 等虚拟技术。对面纷繁复杂的开发环境，`pycharm` 都能游刃有余,当需要不同开发环境的时候，只需要配置用过解释器即可。



3. 总结

至此，我们完成了 `python` 的下载和配置，使得安装的 `python` 成为我们的系统 `python` 解释器并且下载并配置了 `pycharm`，使得 `pycharm` 可以使用我们刚刚安装的 `python`，为我们运行代码