

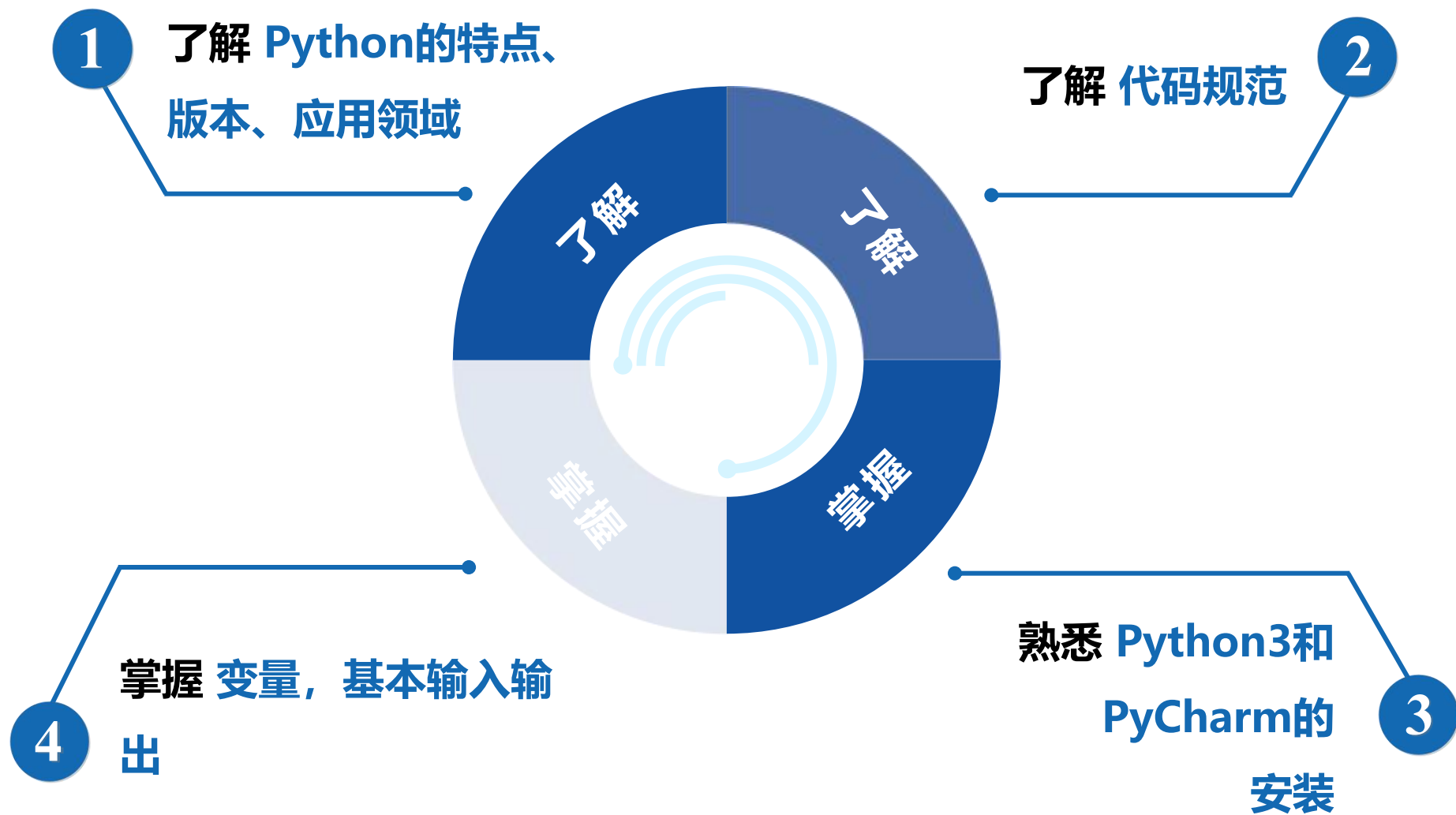
# 第1章 开启Python学习之旅



- Python概述
- 搭建Python开发环境
- 良好的编程约定
- 数据的表示——变量
- 基本输入输出



# 学习目标





# 过渡页



## 01 Python概述

02 搭建Python开发环境

03 快速开发Python程序

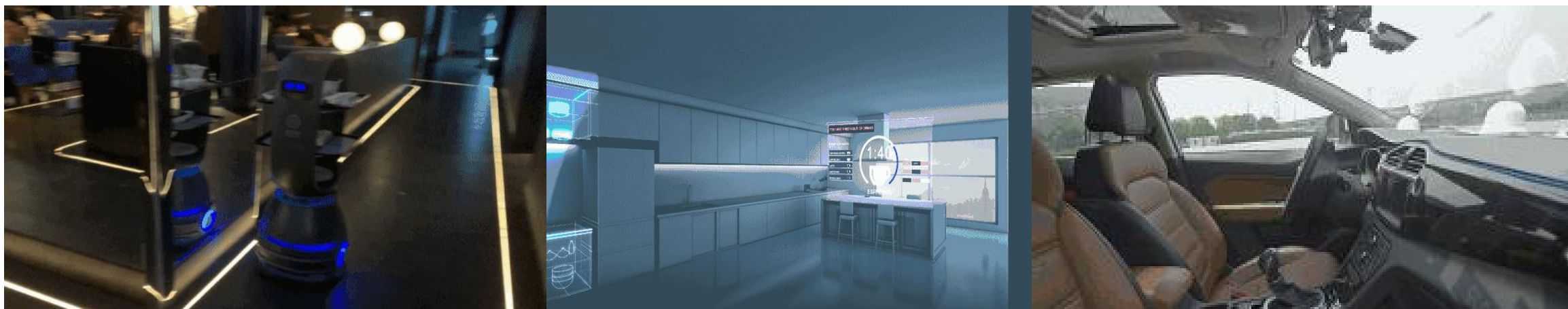
04 实例1：海洋单位距离的换算

05 实例2：打印名片



# Python概述

在方兴未艾的机器学习以及热门的大数据分析技术领域，Python语言的热度可谓是如日中天。



## 吉多·范罗苏姆

Python的作者，Guido von Rossum，荷兰人。

1982年，Guido从阿姆斯特丹大学 (University of Amsterdam) 获得了数学和计算机硕士学位。

之所以选中Python（大蟒蛇的意思）作为该编程语言的名字，是因为他是一个叫Monty Python的喜剧团体的[爱好者](#)。



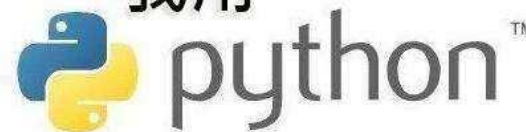


# Python的特点

Python语言之所以能够迅速发展，受到程序员的青睐，与它具有的特点密不可分。

- 简单易学
- 面向对象
- 免费开源
- 丰富的库
- 可移植性

人生苦短  
我用





# Python的特点

## 1. 简单易学

- 语法简单，接近自然语言，用少量关键字就可以实现条件、循环、函数等结构；代码量少

## 2. 免费开源

## 3. 良好的可移植性

- Python作为一种解释性语言，可以在任何安装有Python解释器的环境中运行（Windows, Linux, Mac OS...）



# Python的特点

## 4.面向对象

- 面向对象的程序质量高、效率高、易维护、易扩展。

## 5.丰富的库

- 内置了庞大的标准库
- 丰富的第三方库：
  - 图像处理库-pilow
  - 游戏开发库-pygame
  - 科学计算库-numpy





# Python的历史



[www.python.org](http://www.python.org)

Python 雏形	1991年
Python 1.0	1994年11月
Python 2.0	2000年

Python 2.7.13是Python 2的最后版本

Python 3.0	2008年
------------	-------

Python 3 不向后兼容Python 2

目前: Python 3.10.2





# Python应用领域

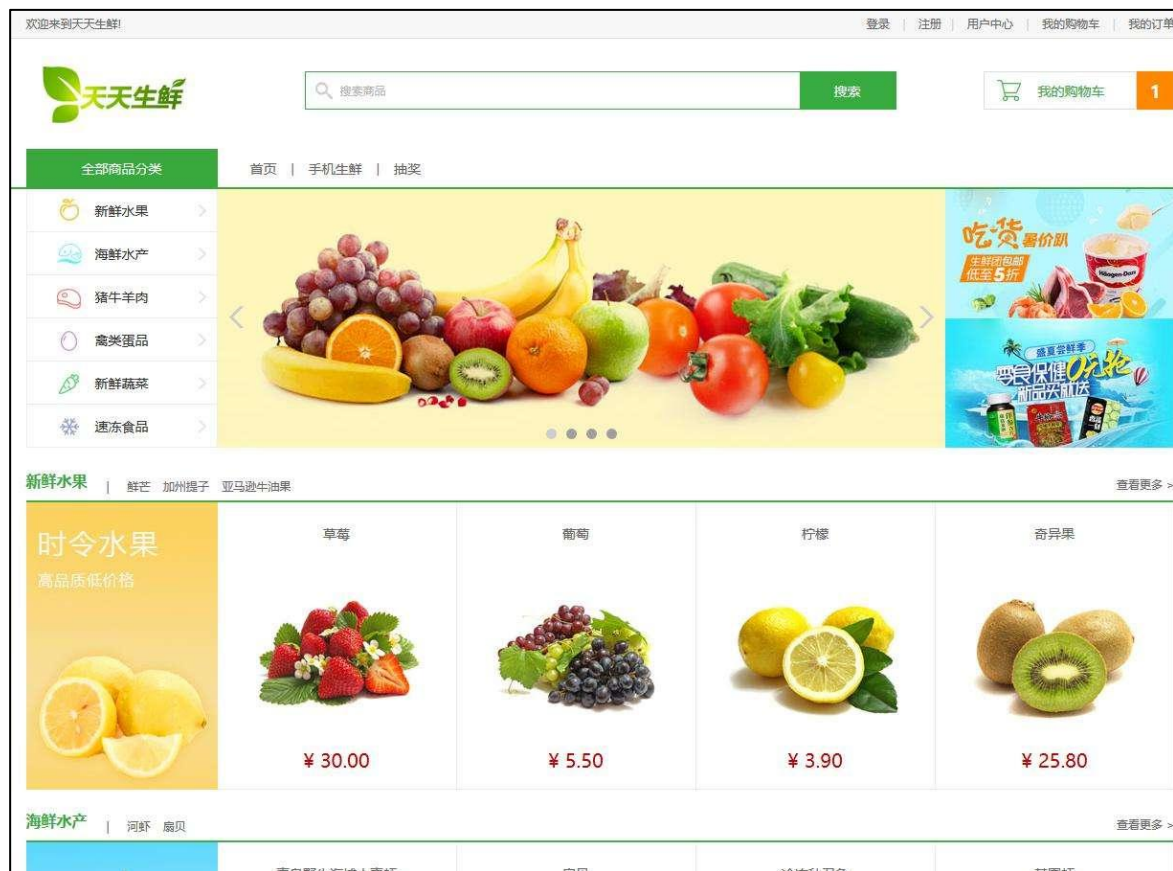
Python作为一门功能强大且简单易学的编程语言得到了广泛应用，它主要应用在以下领域。

- Web开发
- 科学计算与数据分析
- 自动化运维
- 网络爬虫
- 游戏开发
- 人工智能



# Python应用领域

- Web开发





# Python应用领域

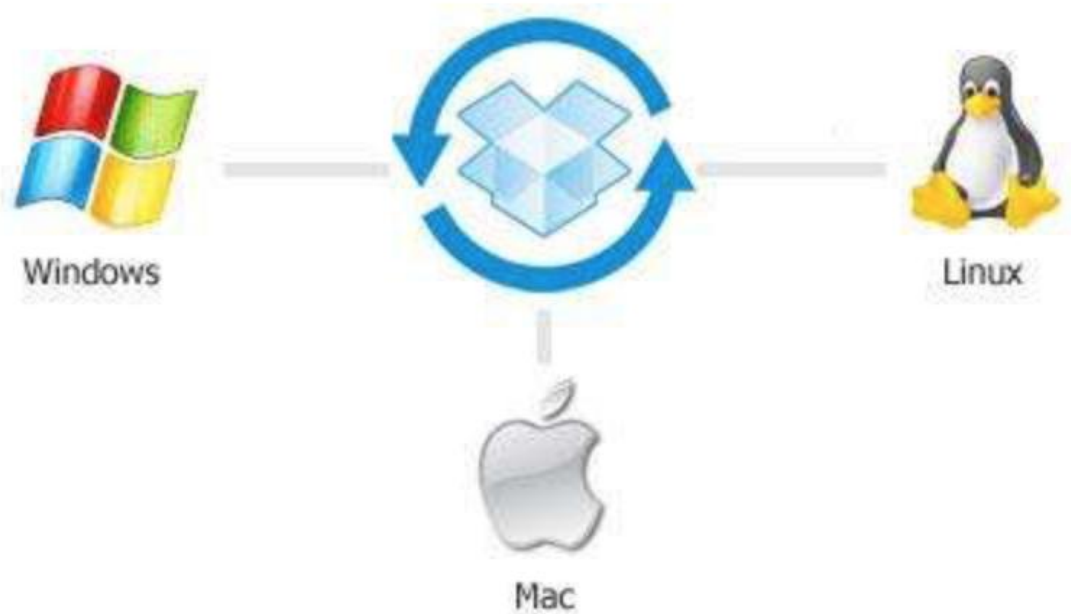
- 科学计算与数据分析





# Python应用领域

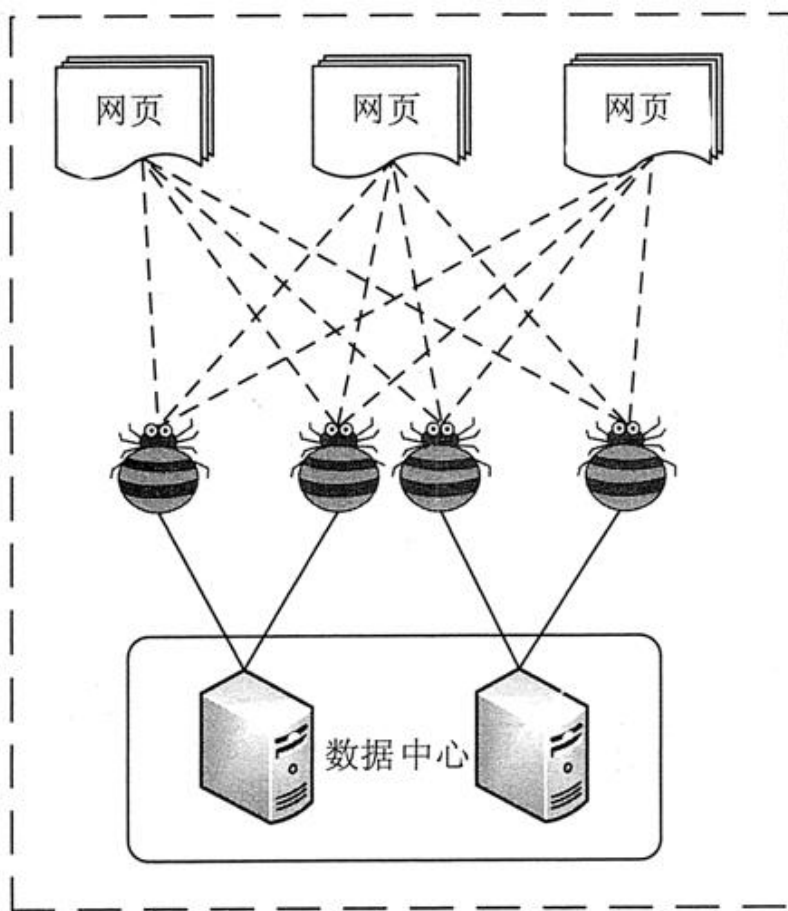
- 自动化运维





# Python应用领域

- 网络爬虫

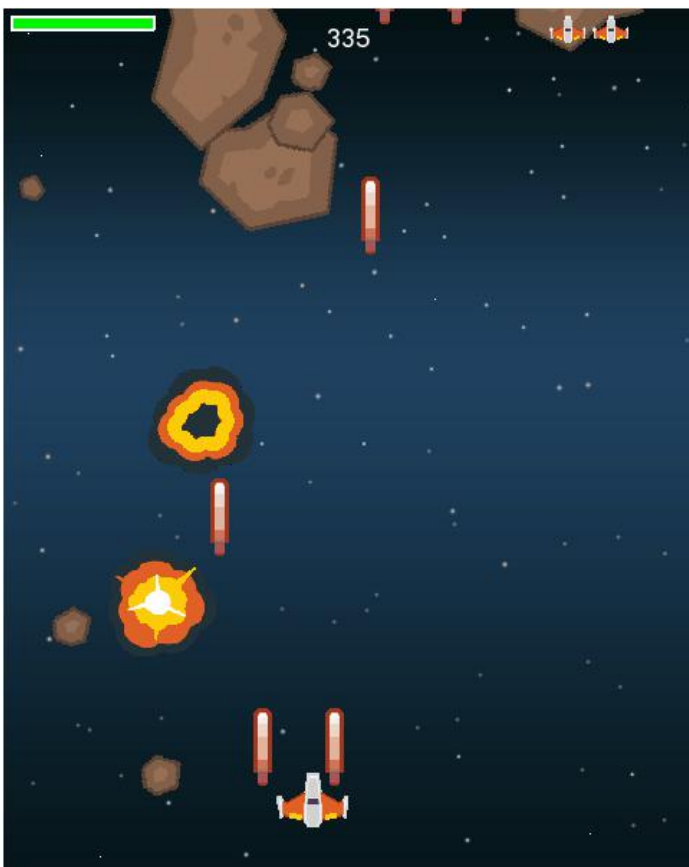






# Python应用领域

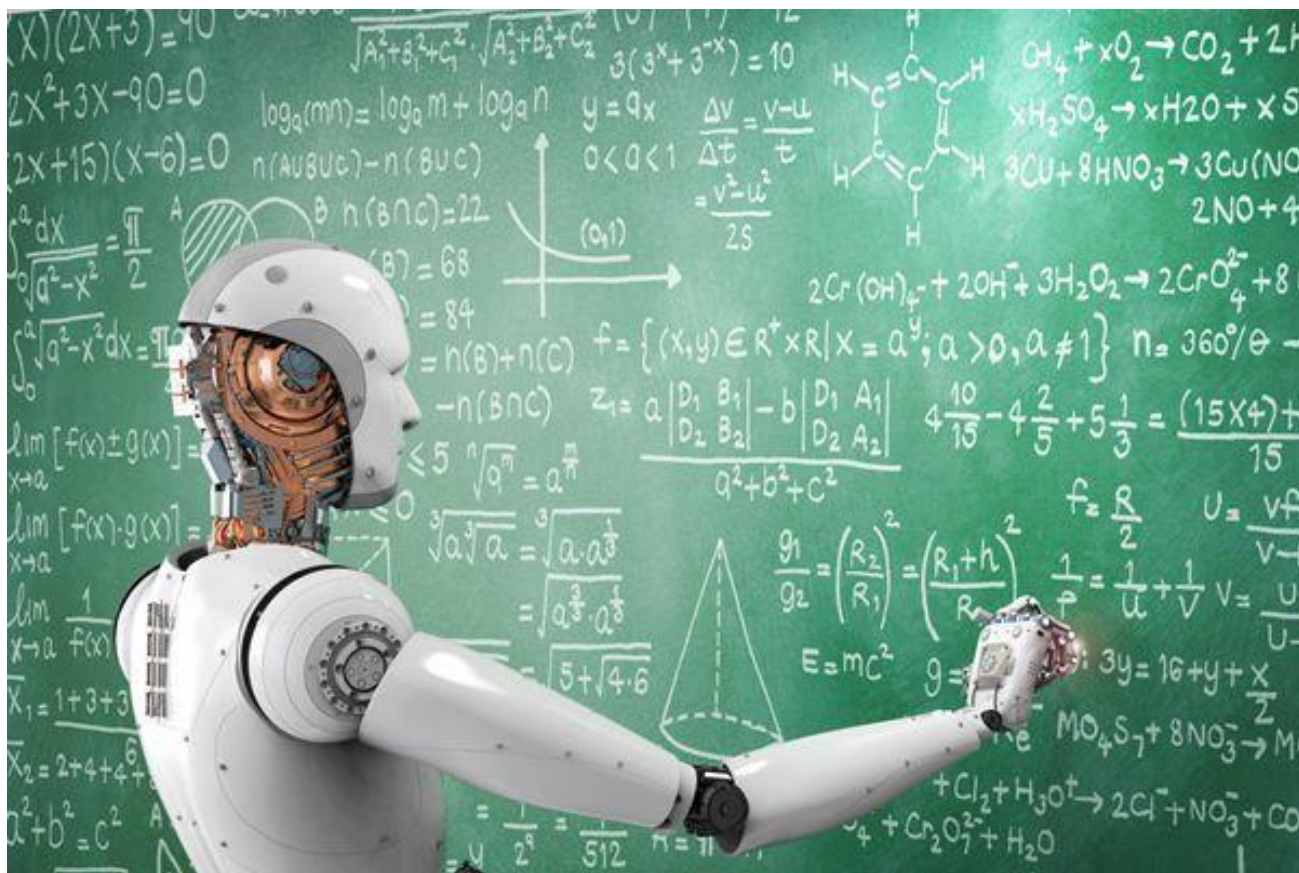
- 游戏开发





# Python应用领域

## ● 人工智能







# Python程序设计



**01 Python概述**

**02 搭建Python开发环境**

**03 快速开发Python程序**

**04 实例1：海洋单位距离的换算**

**05 实例2：打印名片**



# 搭建Python 开发环境

- 1. 下载并安装 Python 3.x
- 2. 下载并安装 pycharm-community-4.0.4  
(非必需, 但强烈推荐)

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

配置 pycharm

在pycharm建工程

编写、运行python程序



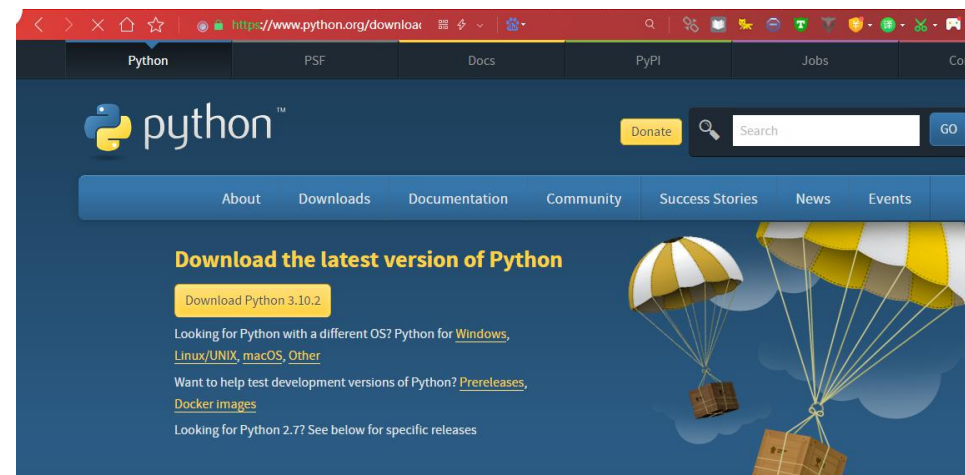
# 官网python

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

[Mac OS X 64-bit/32-bit installer](#)

[Windows x86-64 executable installer](#)

[Windows x86 executable installer](#)



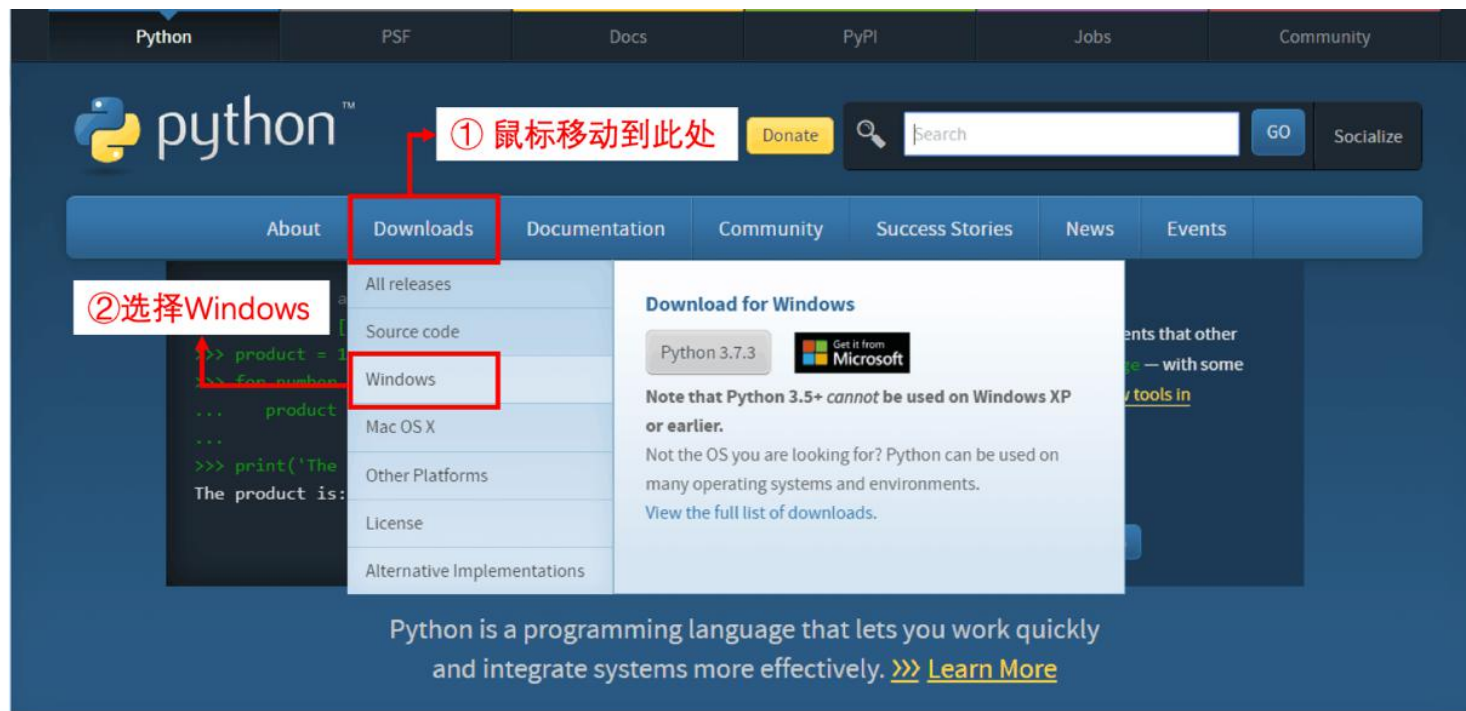
64位Windows

32位Windows



# Python 3.x

(1) 访问Python官网<https://www.python.org/downloads/>, 选择【Downloads】→【Windows】。





# Python3的安装

(2) 单击 “Windows”  
后跳转到Python下载页面，该页面中包含多个版本的安装包，根据自身需求下载相应的版本。

## Python Releases for Windows

- [Latest Python 3 Release - Python 3.10.2](#)
- [Latest Python 2 Release - Python 2.7.18](#)

### Stable Releases

- [Python 3.9.10 - Jan. 14, 2022](#)

**Note that Python 3.9.10 cannot be used on Windows 7 or earlier.**

- Download [Windows embeddable package \(32-bit\)](#)
- Download [Windows embeddable package \(64-bit\)](#)
- Download [Windows help file](#)
- Download [Windows installer \(32-bit\)](#)
- Download [Windows installer \(64-bit\)](#)

- [Python 3.10.2 - Jan. 14, 2022](#)

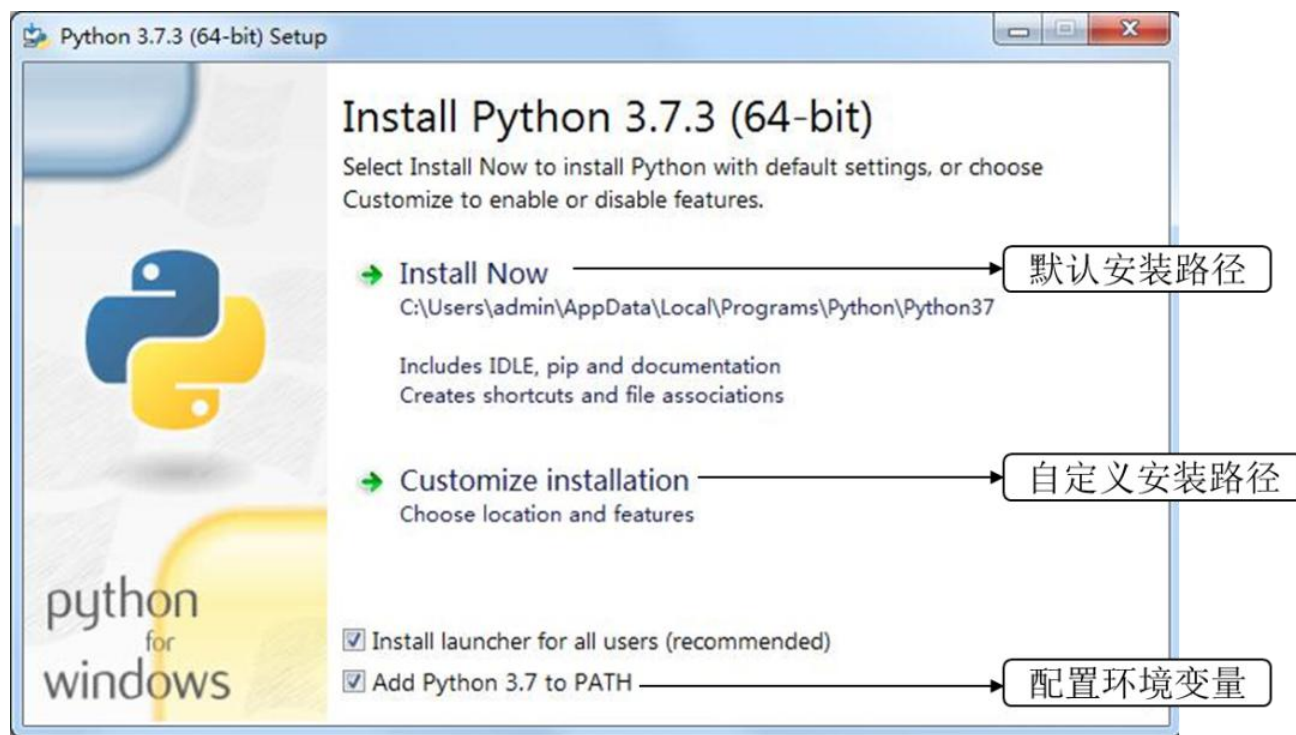
**Note that Python 3.10.2 cannot be used on Windows 7 or earlier.**

- Download [Windows embeddable package \(32-bit\)](#)
- Download [Windows embeddable package \(64-bit\)](#)
- Download [Windows help file](#)
- Download [Windows installer \(32-bit\)](#)
- [Download Windows installer \(64-bit\)](#)



# Python3的安装

(3) 这里选择下载64位离线安装包，下载成功后，双击打开进行安装。

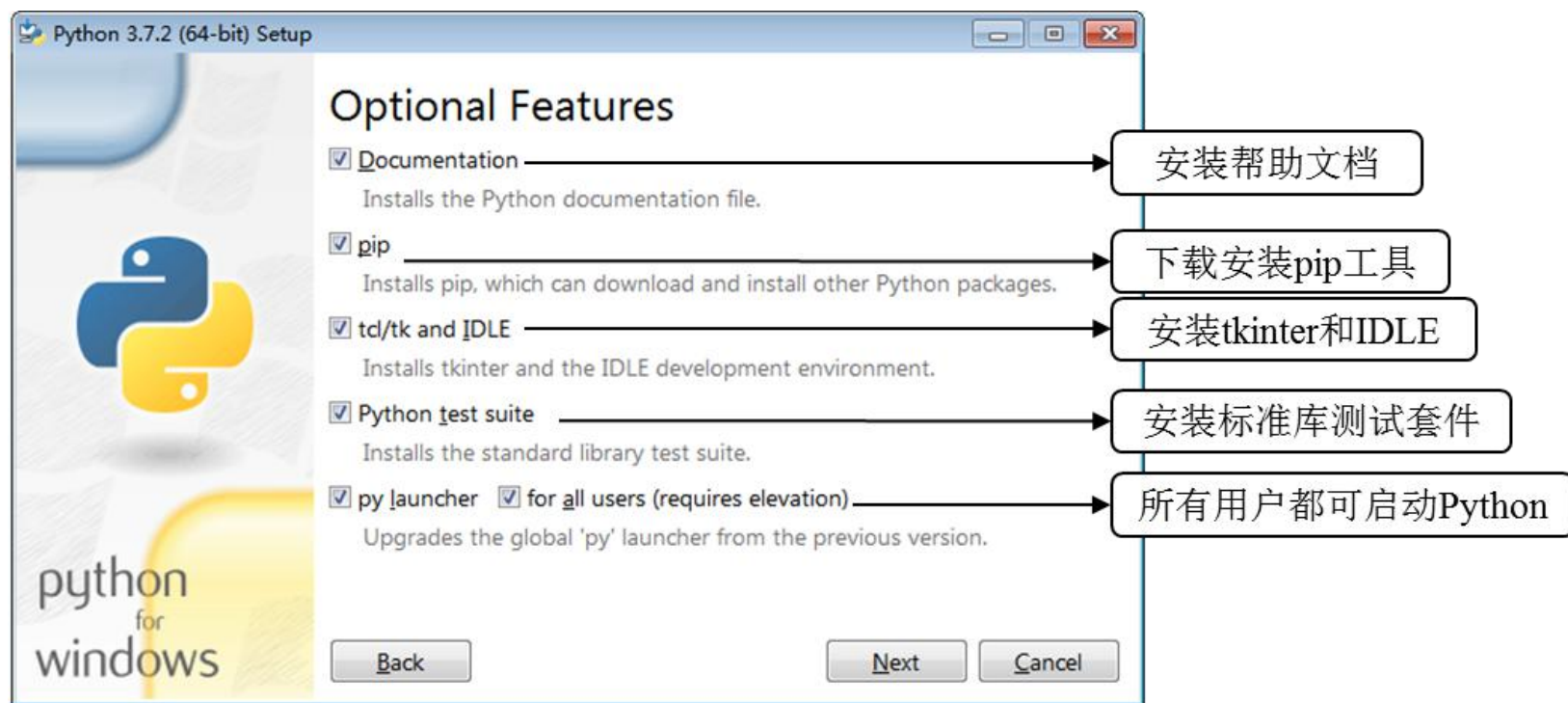






# Python3的安装

(4) 单击 “Customize installation” ，进入设置可选功能的界面。





# Python3的安装

(5) 保持默认配置。单击【Next】按钮进入设置高级选项的界面，用户在该界面中设置Python安装路径。

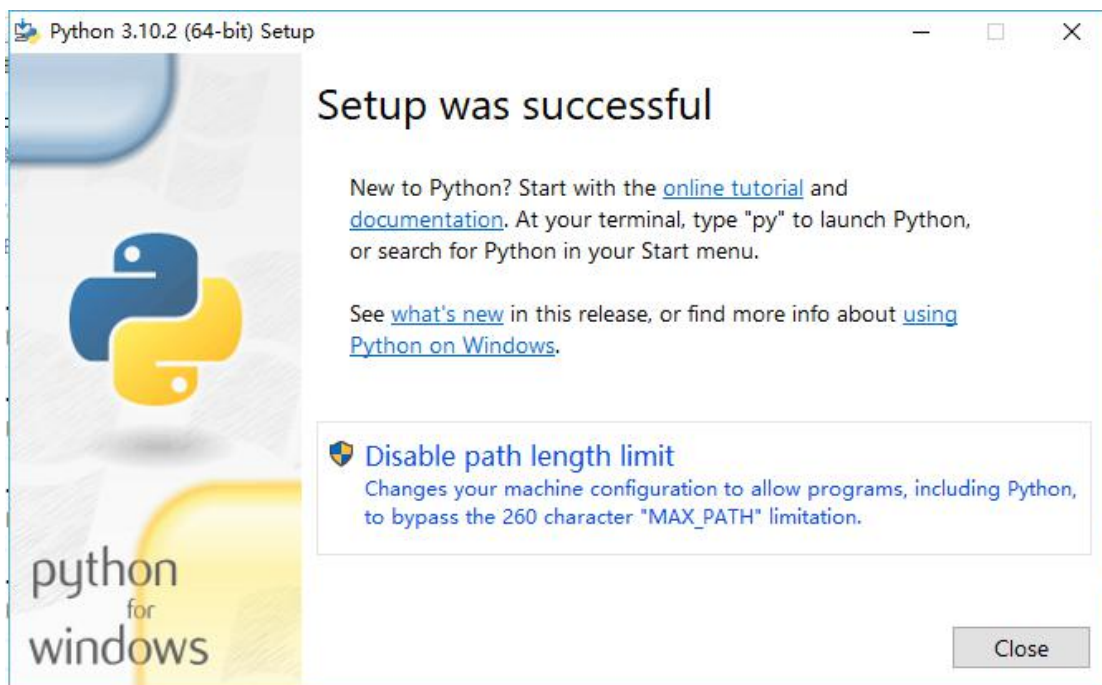






# Python3的安装

(6) 选定Python的安装路径后，单击【Install】按钮开始安装，安装成功后如下图所示。





# Python3的安装

(7) 在Windows系统中打开命令提示符，在命令提示符窗口中输入“python”后显示了Python的版本信息，表明安装成功。



```
管理员: C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\admin>python
Python 3.7.3 (tags/v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```



# IDLE的使用



**思考：**  
什么是IDLE?



# IDLE的使用

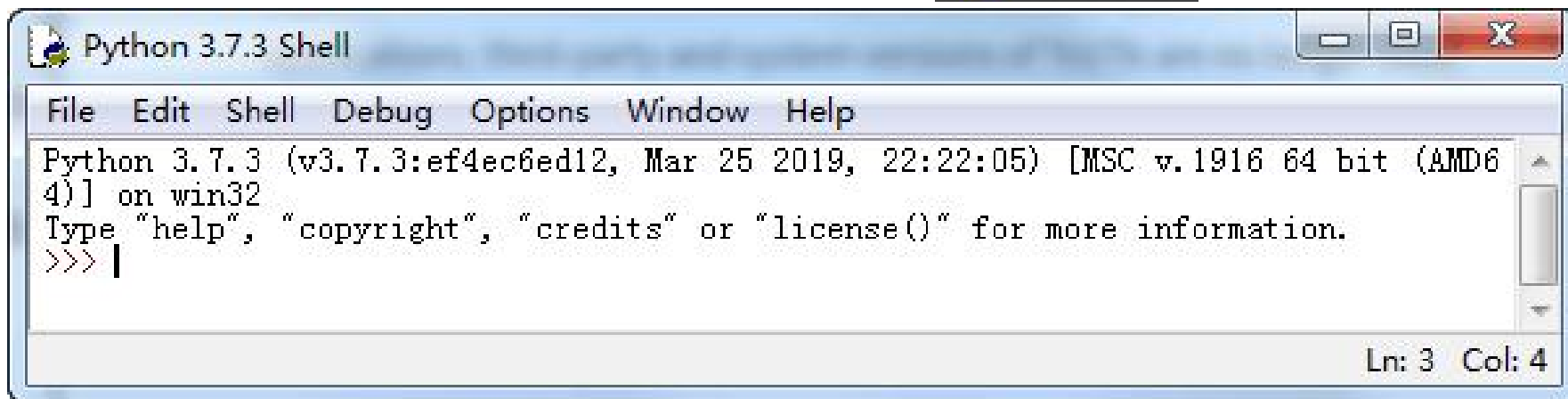


Python安装过程中默认自动安装了  
IDLE (**Integrated Development and  
Learning Environment**)，IDLE是  
Python自带的集成开发学习环境。



# IDLE的使用

在Windows系统的开始菜单的搜索栏中输入  
“IDLE” 或者单击IDLE (Python 3.7 64-bit) 进入IDLE界面。

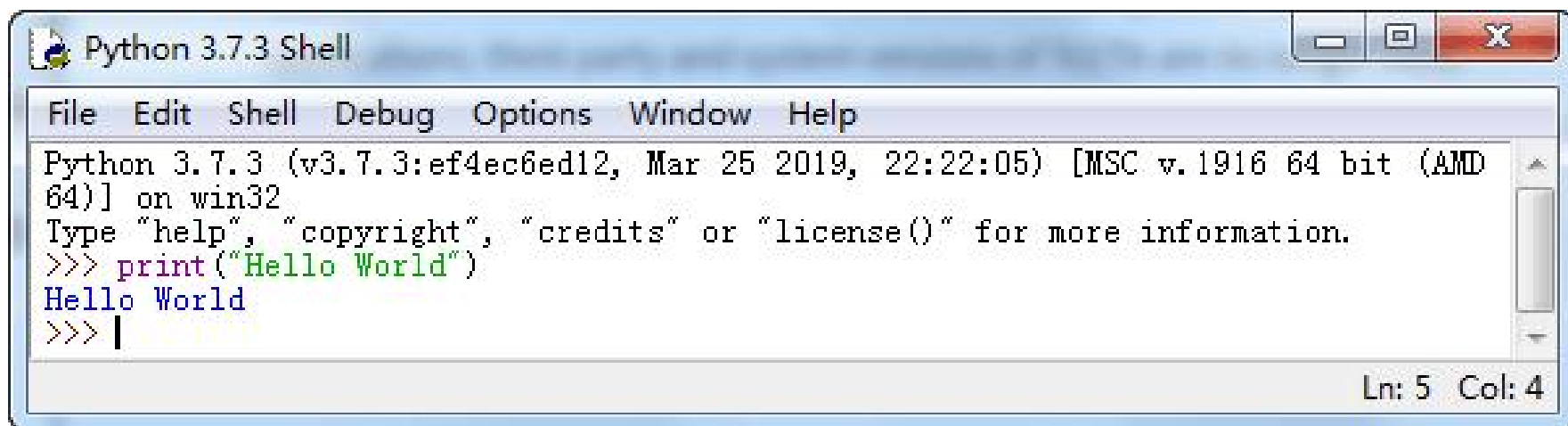




# IDLE的使用

可以在Shell界面中直接编写Python代码。

- 例如，使用print()函数输出 “hello world”



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD
64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> |
```

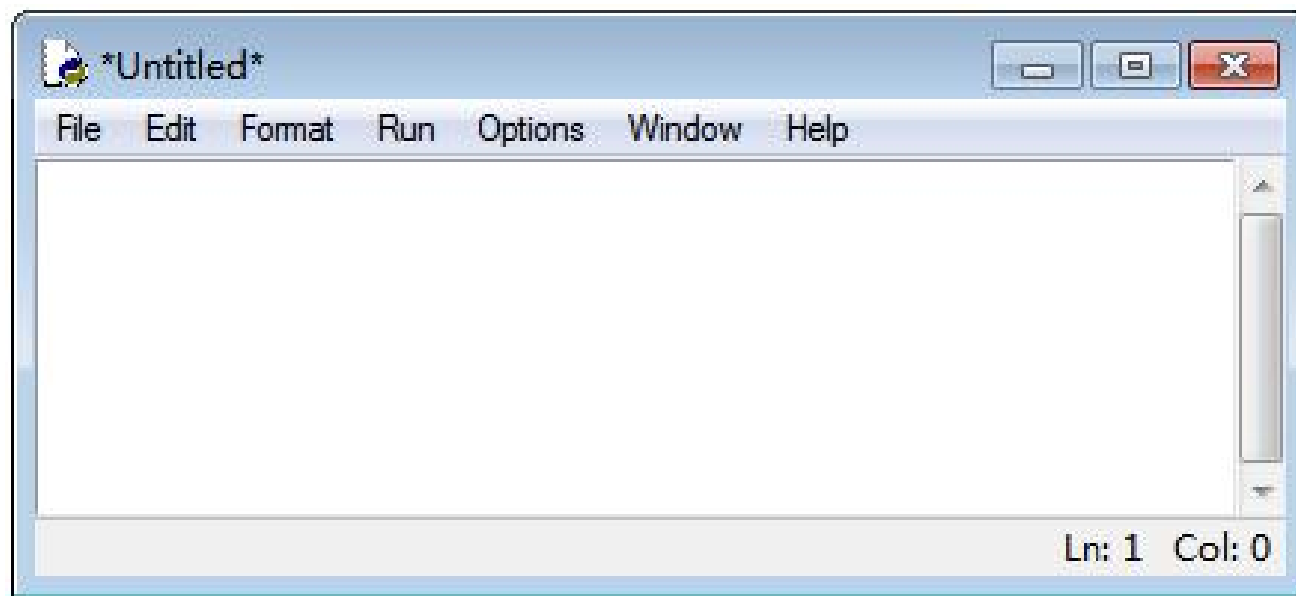
Ln: 5 Col: 4



# IDLE的使用

IDLE除了支持交互式编写代码，还支持文件式编写代码。

在交互式窗口中选择  
【File】 → 【New  
File】，创建并打开  
一个新的界面。





# IDLE的使用

在刚刚打开的界面中编写如下代码。

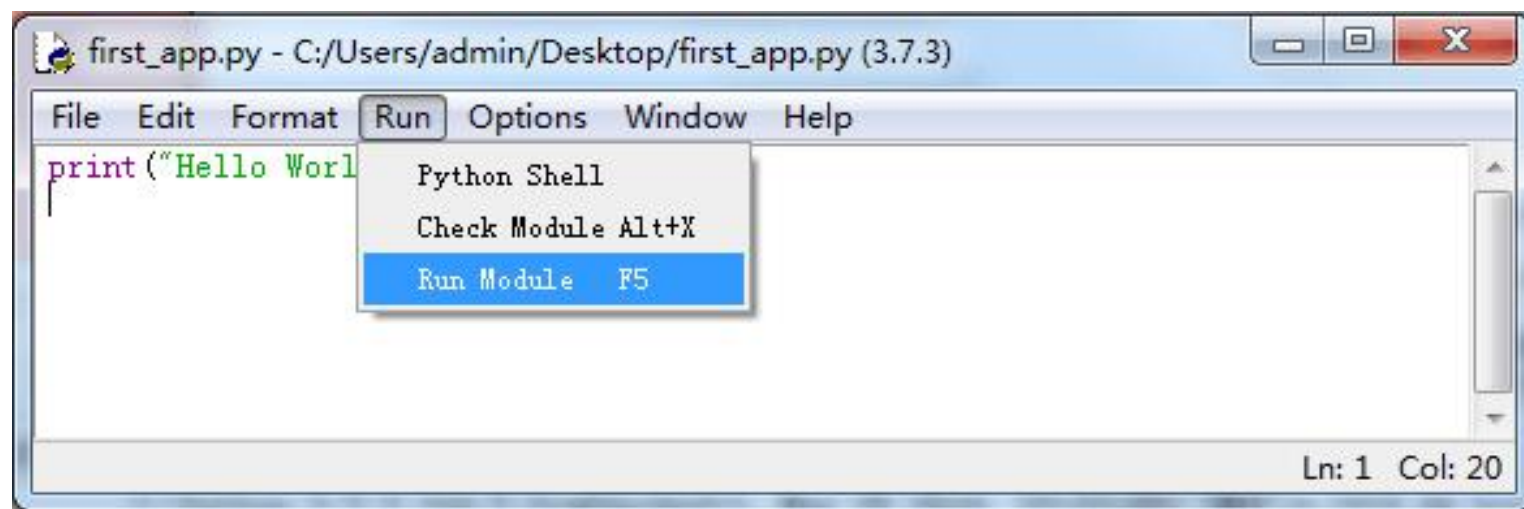
```
print("Hello World")
```





# IDLE的使用

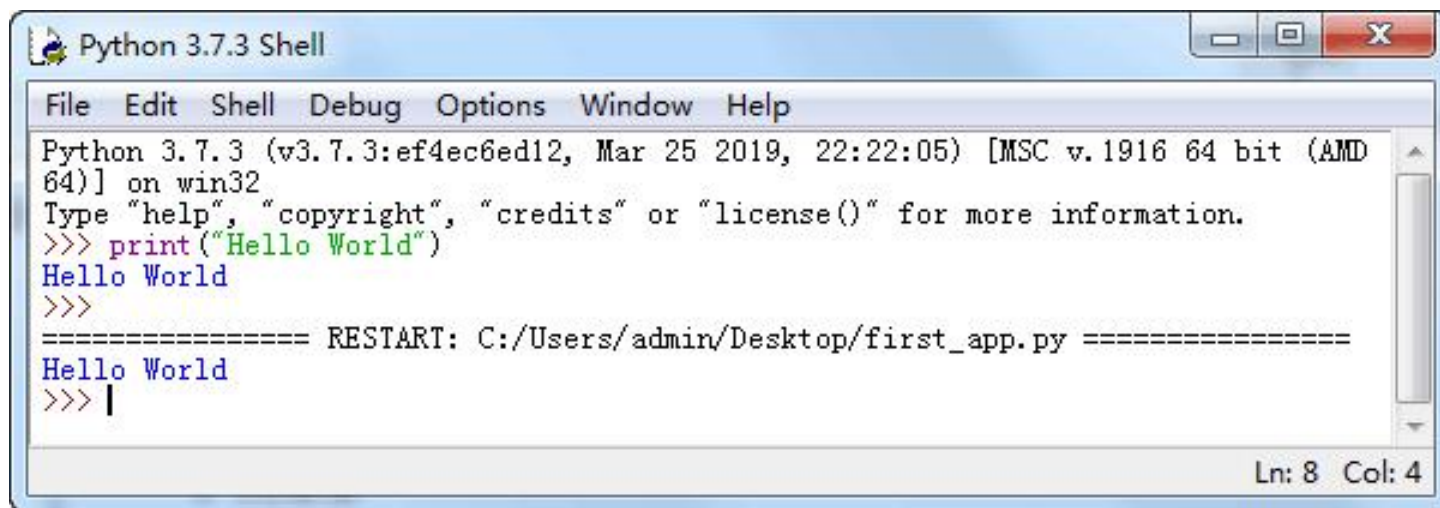
选择【File】→【Save As】将文件以  
“first\_app”命名并保存，之后在窗口中选  
择【Run】-【Run Module F5】运行代码。





# IDLE的使用

单击【Run Moudle F5】选项后，Python Shell窗口中显示了运行结果。



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD
64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>>
===== RESTART: C:/Users/admin/Desktop/first_app.py =====
Hello World
>>> |
```

Ln: 8 Col: 4



# 集成开发环境PyCharm的安装与使用

PyCharm是Python集成开发环境：

- 具有智能代码编辑器
- 智能提示
- 自动导入等功能
- Python专业开发人员广泛使用的开发工具。



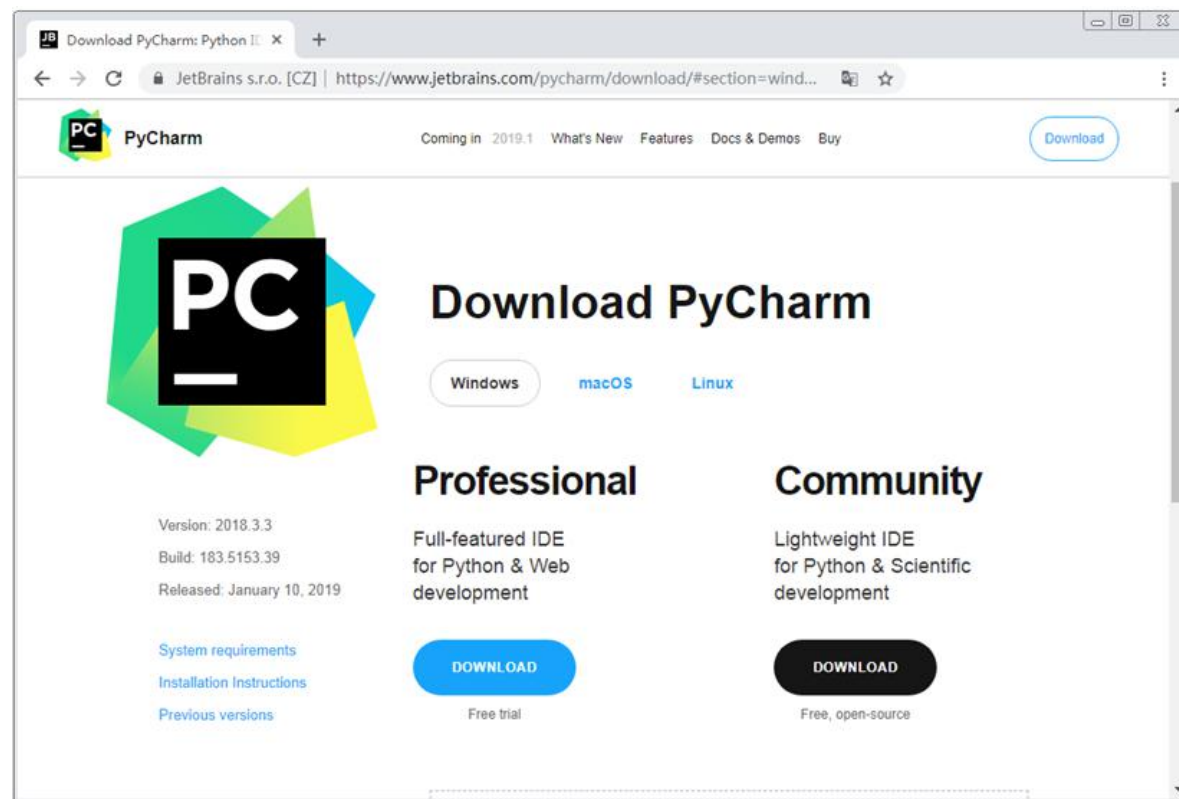


# 集成开发环境PyCharm的安装与使用

- **PyCharm的安装**

(1) 访问jetbrains  
官网中下载PyCharm  
工具的页面。

<https://www.jetbrains.com/pycharm/>





# 集成开发环境PyCharm的安装与使用

PyCharm包含Professional和Community  
(免费, 推荐) 两个版本。

- 提供Python IDE的所有功能;
- 支持Django、Flask等;
- 支持JavaScript、CoffeeScript等;
- 支持远程开发、Python分析器、数据库和SQL语句。

**Professional版本**

- 轻量级的Python IDE, 只支持Python开发;
- 免费、开源、集成Apache2的许可证; 智能编辑器、调试器;
- 支持重构和错误检查, 集成VCS版本控制。

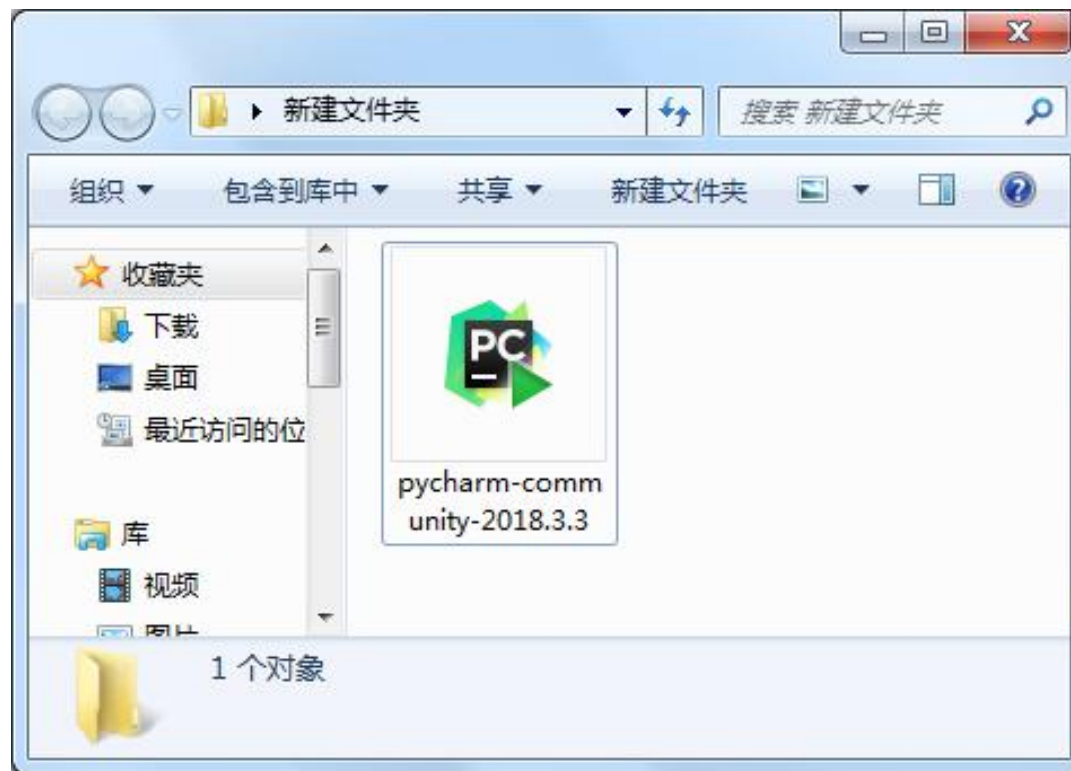
**Community版本**



# 集成开发环境PyCharm的安装与使用

- **PyCharm的安装**

(2) 单击相应版本下的【**DOWNLOAD**】按钮开始下载PyCharm安装包，这里下载Community版本。





# 集成开发环境PyCharm的安装与使用

- **PyCharm的安装**

(3) 下载成功后，  
双击“pycharm-  
community-2018.3.3”  
安装包弹出欢迎界面。





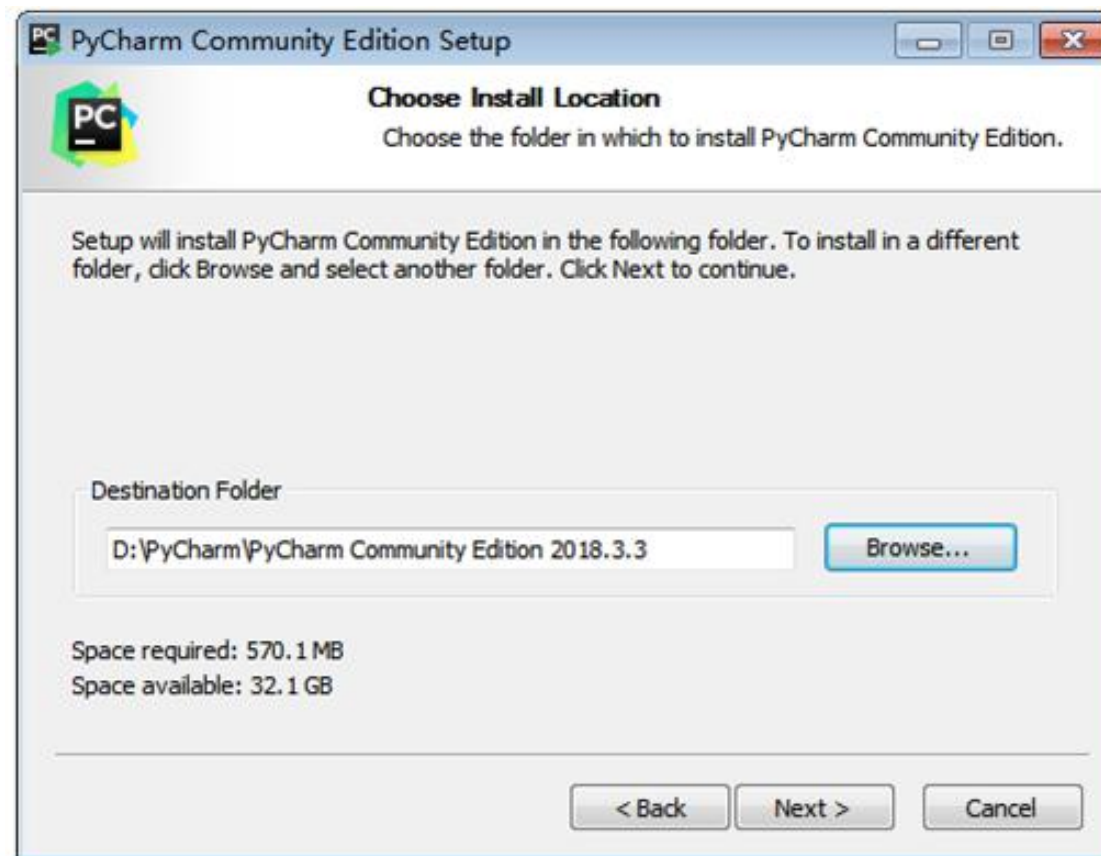


# 集成开发环境PyCharm的安装与使用

- **PyCharm的安装**

(4) 单击【Next】

按钮进入PyCharm选择安装路径的界面。



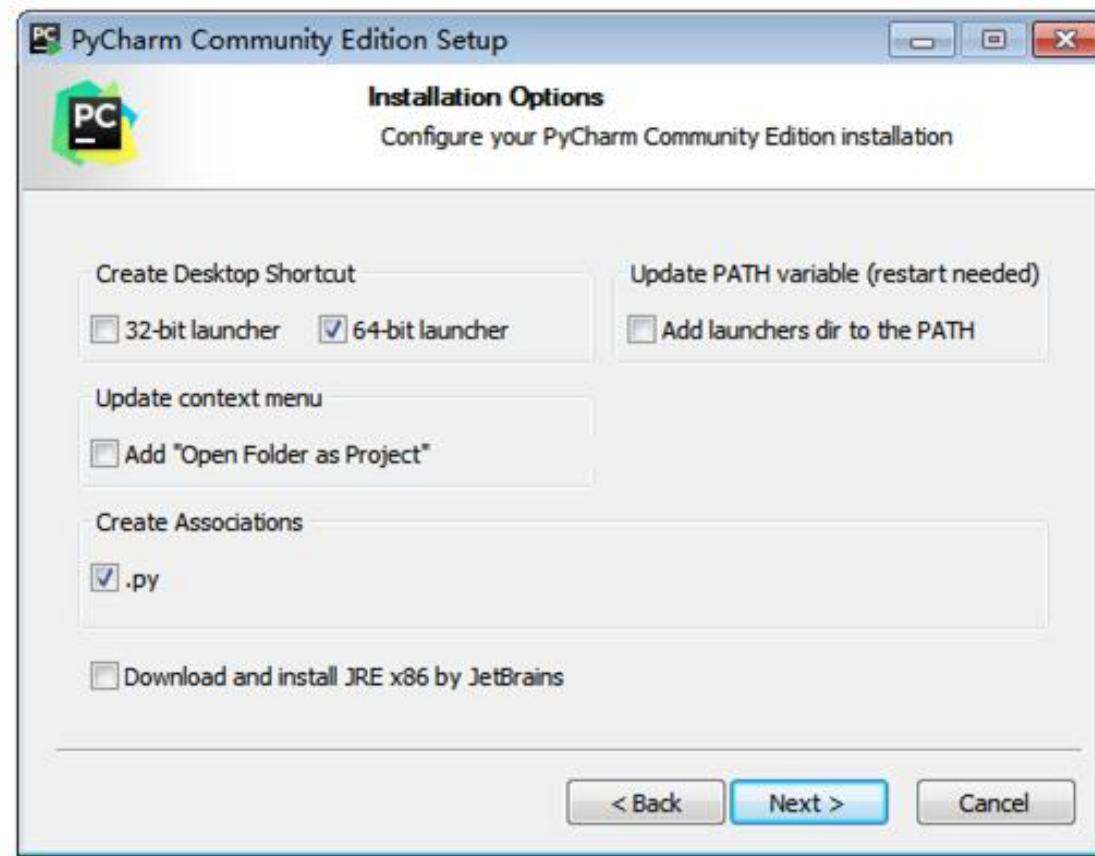




# 集成开发环境PyCharm的安装与使用

## • PyCharm的安装

(5) 确定好安装位置后，单击【Next】按钮进入安装选项界面，在该界面中用户可根据需求勾选相应功能。

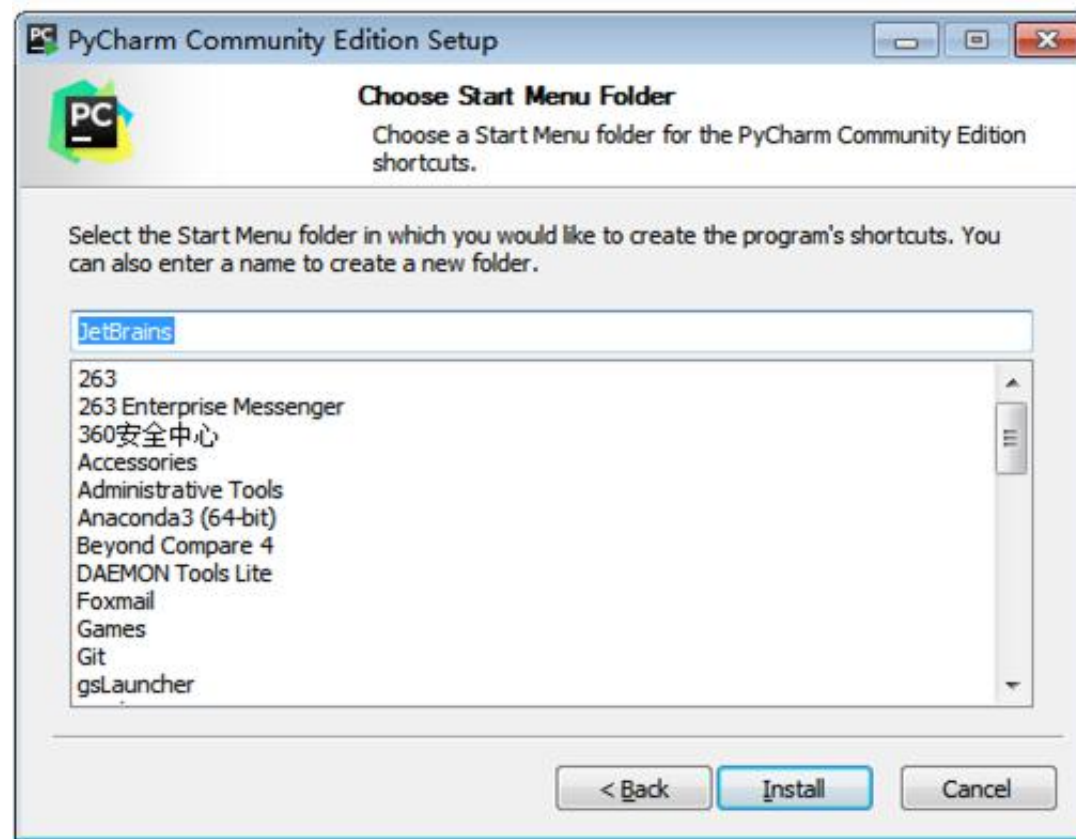




# 集成开发环境PyCharm的安装与使用

- **PyCharm的安装**

(6) 保持默认配置，单击【Next】按钮进入选择开始菜单文件夹的界面，该界面中依然保持默认配置。

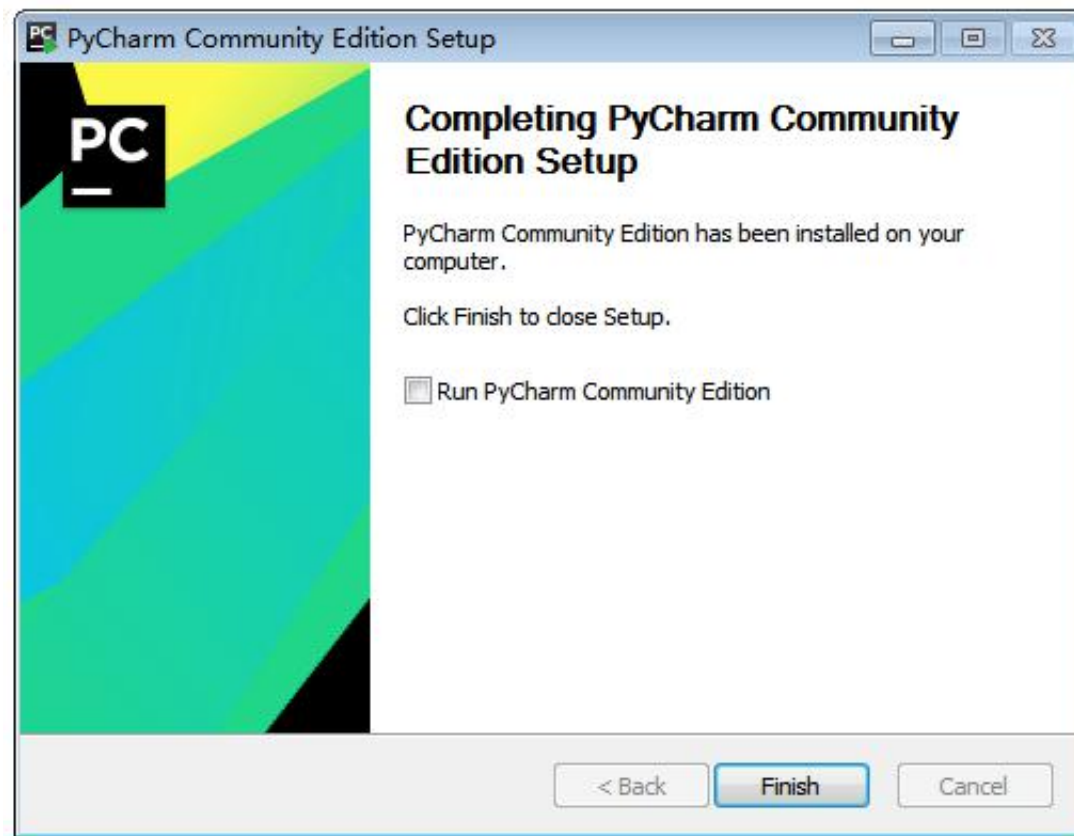




# 集成开发环境PyCharm的安装与使用

- **PyCharm的安装**

(7) 单击【Install】按钮安装PyCharm，安装完成后提示“Completing PyCharm Community Edition Setup”信息。



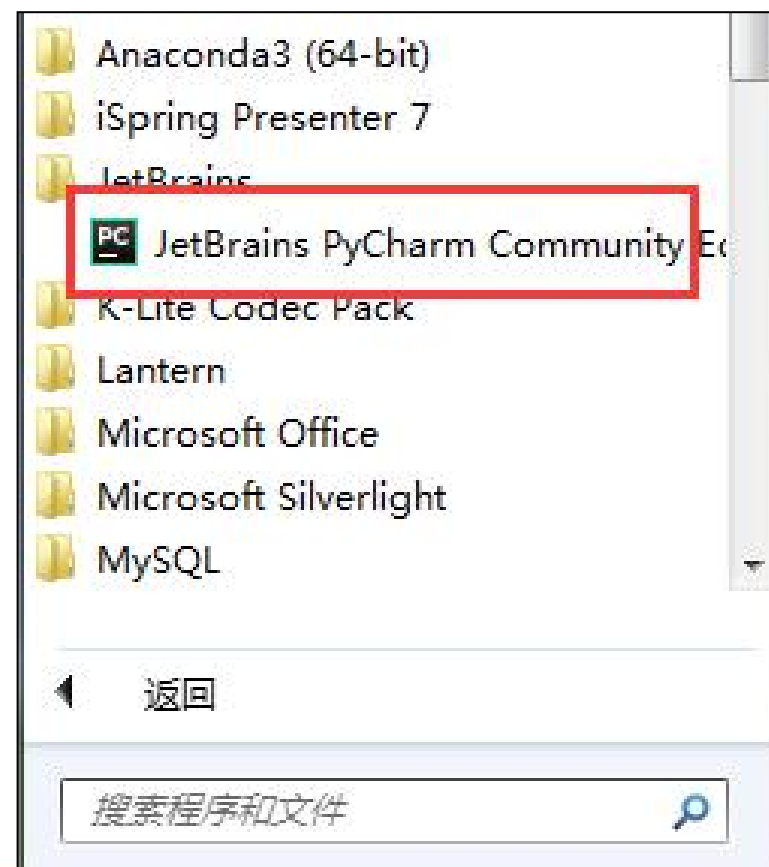


# 集成开发环境PyCharm的安装与使用

- **PyCharm的安装**

(8) 单击【Finish】

按钮结束PyCharm安装。

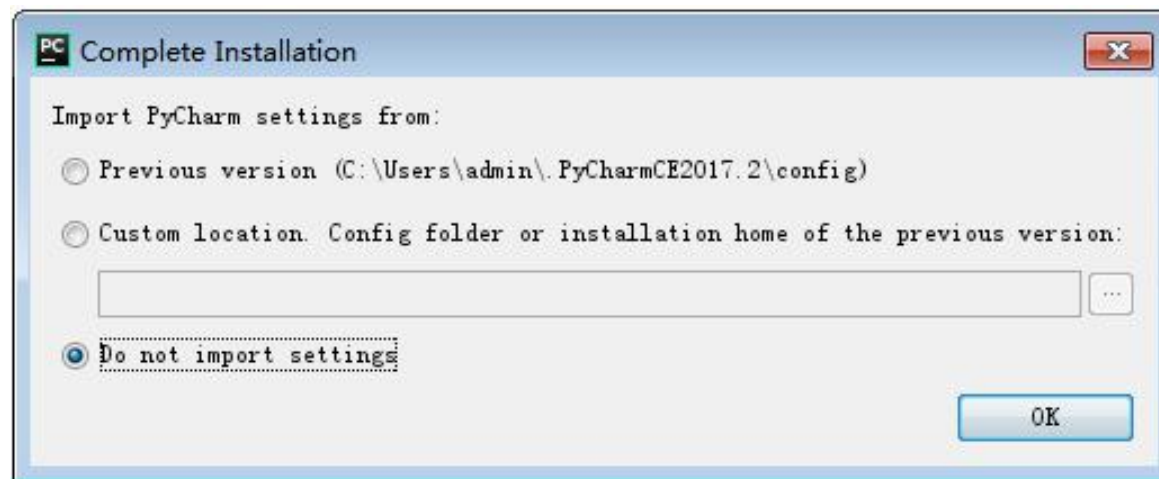




# 集成开发环境PyCharm的安装与使用

- **PyCharm的使用**

(1) 双击PyCharm快捷方式图标进入导入配置文件的界面。

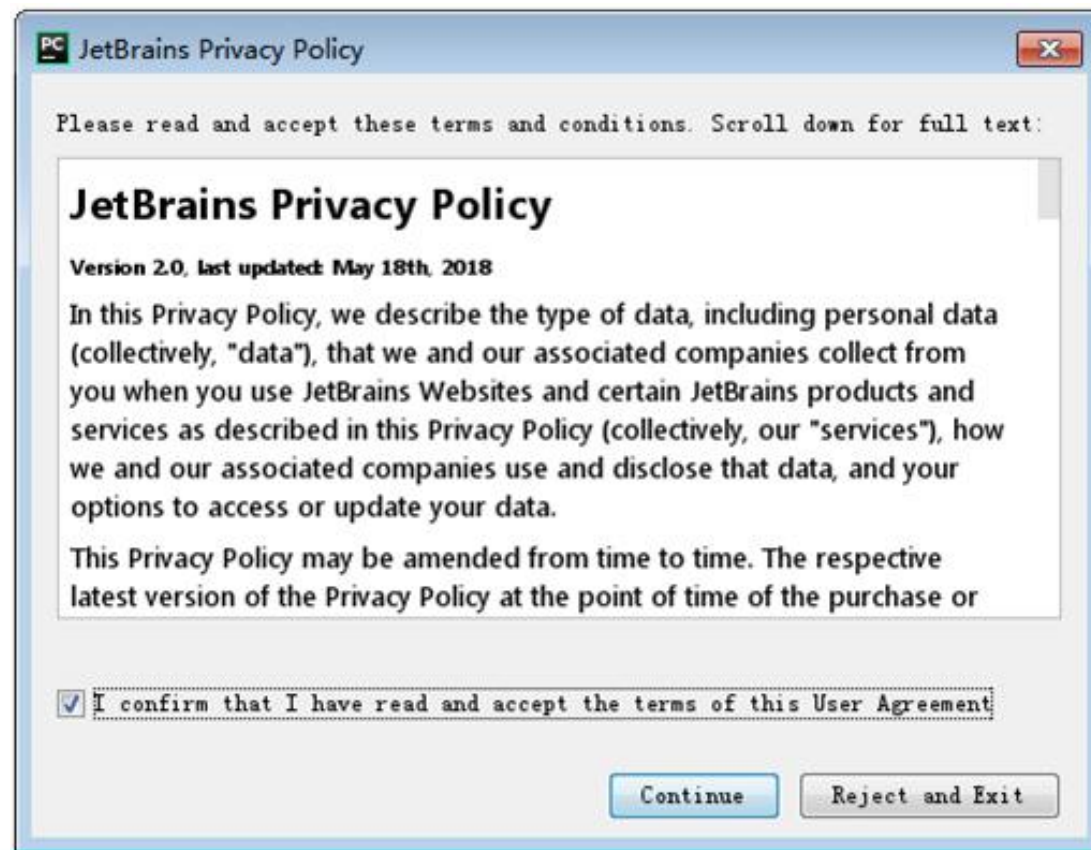




# 集成开发环境PyCharm的安装与使用

- **PyCharm的使用**

(2) 这里选择“Do not import settings”，单击【OK】按钮进入JetBrains用户协议界面。



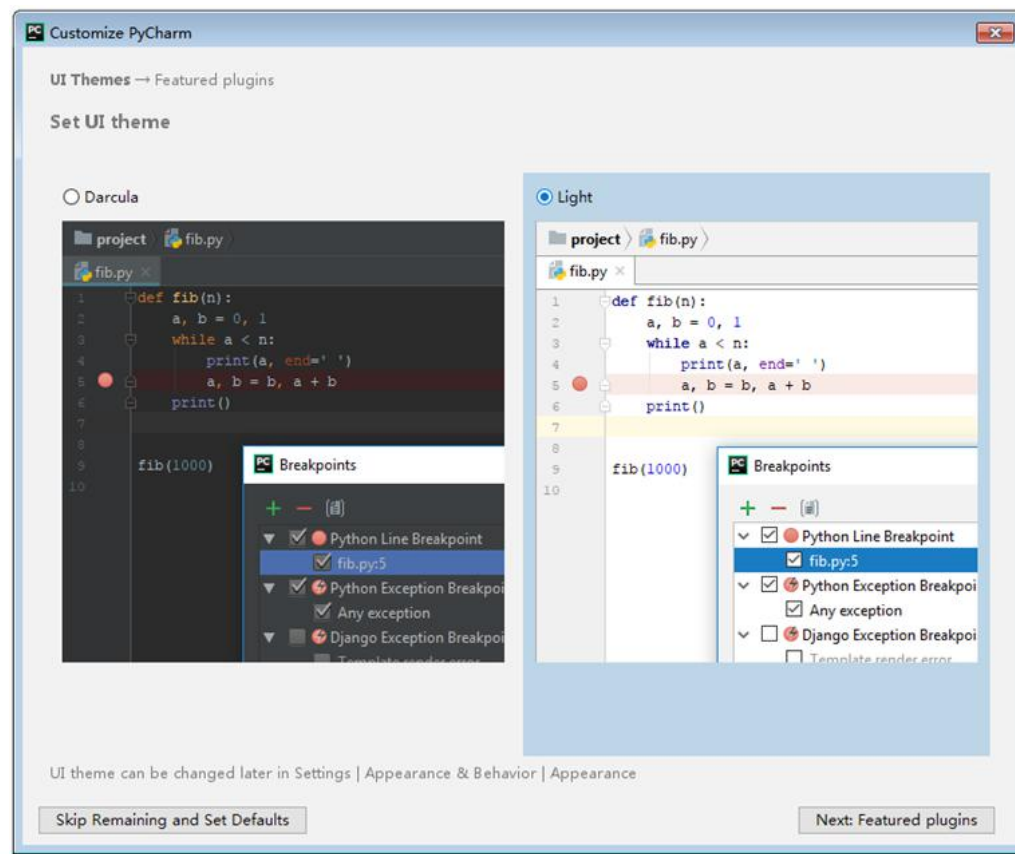




# 集成开发环境PyCharm的安装与使用

## • PyCharm的使用

(3) 单击【Continue】按钮进入环境设置界面，该界面中可设置用户主题，这里选择“Light”主题。



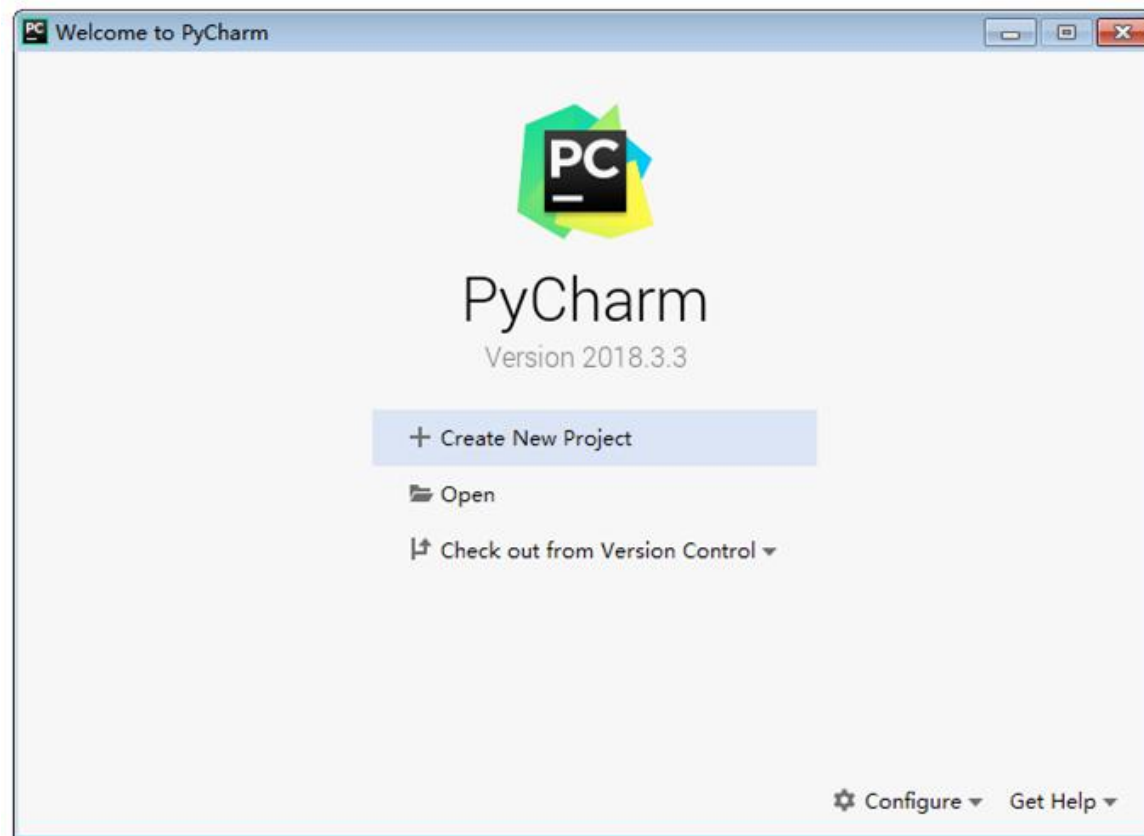




# 集成开发环境PyCharm的安装与使用

- **PyCharm的使用**

(4) 单击【Skip Remaining and Set Defaults】按钮进入PyCharm欢迎界面。

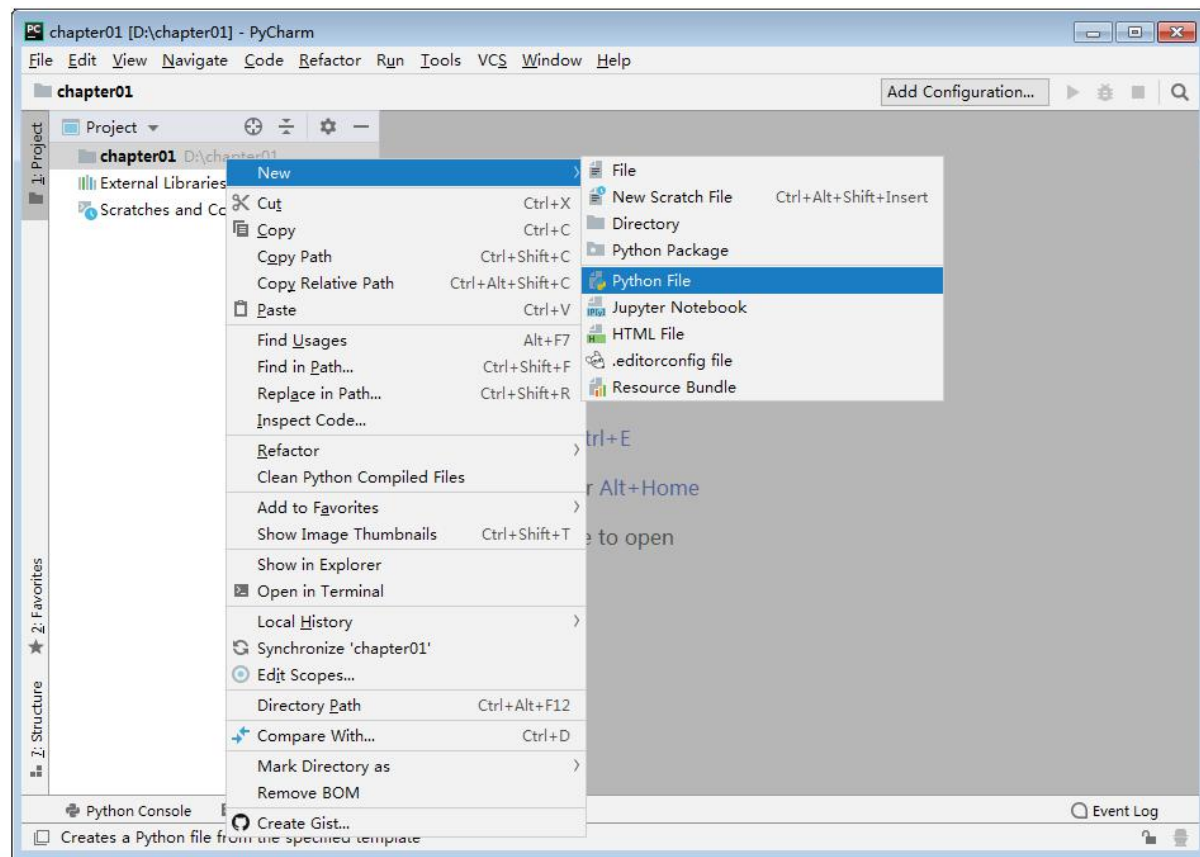




# 集成开发环境PyCharm的安装与使用

## • PyCharm的使用

(5) 单击【Create New Project】按钮创建一个Python项目chapter01，之后便可以在项目中创建一个py文件，具体操作为：选中项目名称chapter01，右击选择【New】→【Python File】选项。

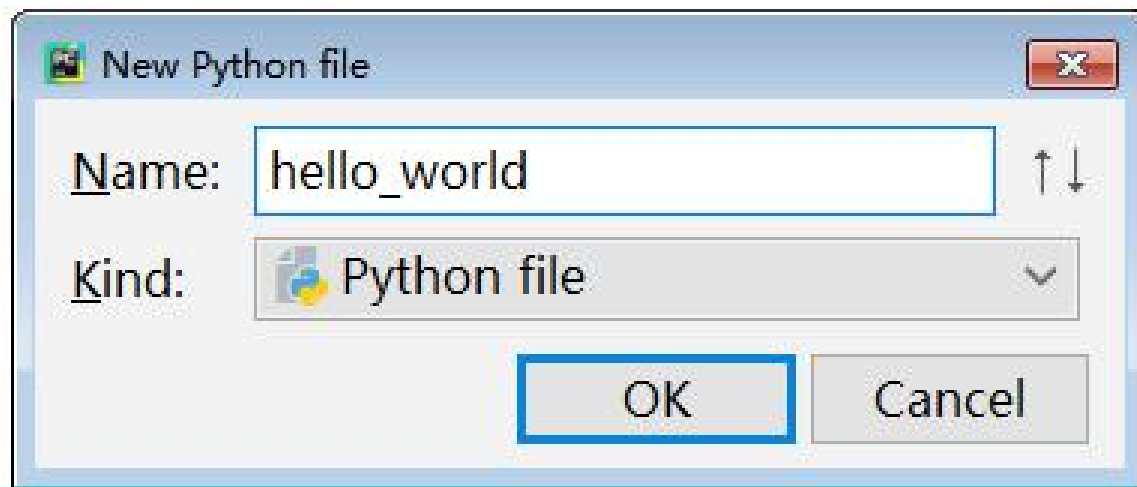




# 集成开发环境PyCharm的安装与使用

- **PyCharm的使用**

(6) 将刚刚新建的Python文件命名为hello\_world，使用默认文件类型Python file。

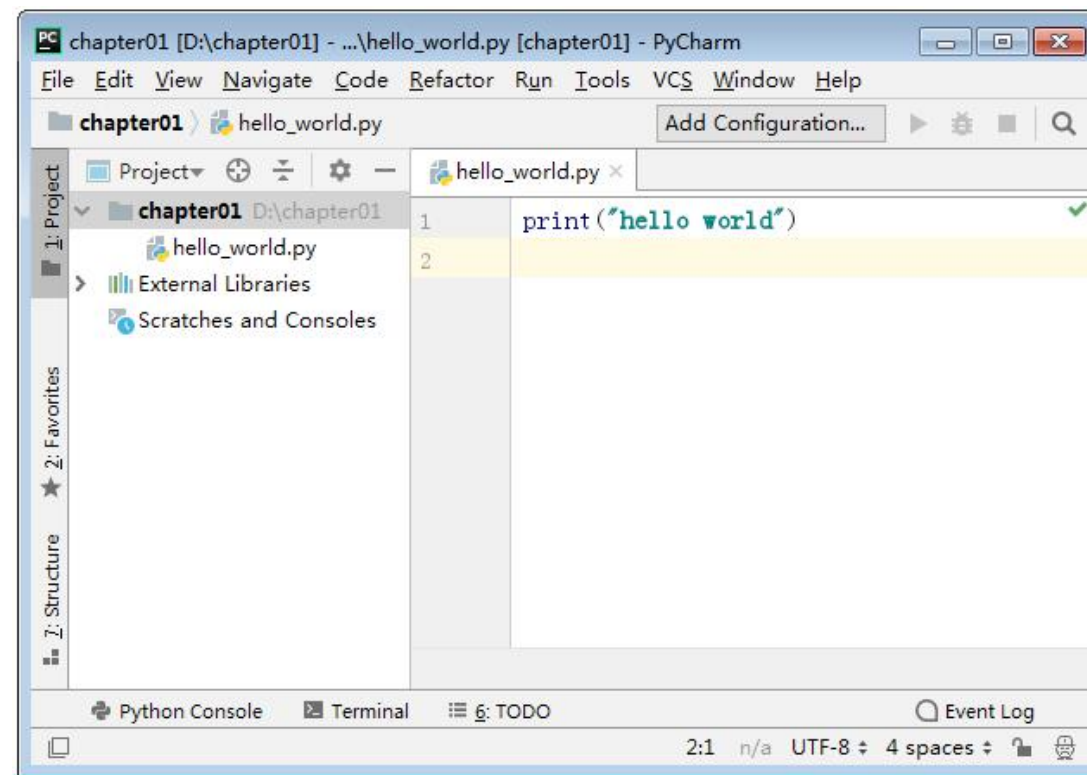




# 集成开发环境PyCharm的安装与使用

- PyCharm的使用

(7) 在创建好的  
“hello\_world.py”文件中  
编写如下代码：  
`print("hello world")`

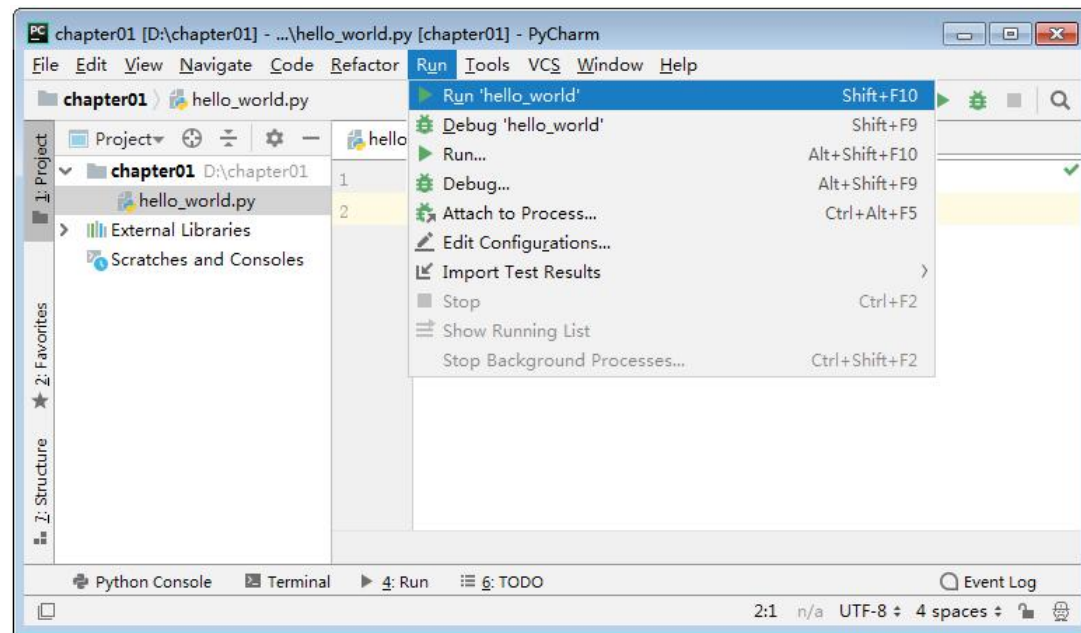




# 集成开发环境PyCharm的安装与使用

- **PyCharm的使用**

(8) 选择【Run】→  
【Run 'hello\_world'】运  
行“hello\_world.py”文件  
(也可以在编辑区中右击  
选择【Run 'hello\_world'】  
来运行文件)。

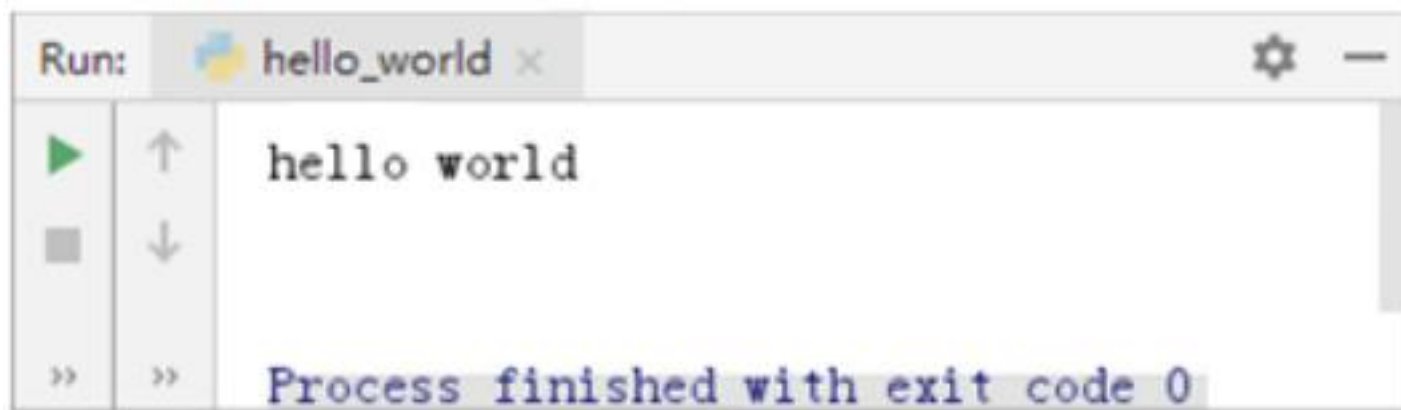




# 集成开发环境PyCharm的安装与使用

- **PyCharm的使用**

(9) 程序的运行结果会在PyCharm结果输出区进行显示。





# 过渡页



**01 Python概述**

**02 搭建Python开发环境**

**03 快速开发Python程序**

**04 实例1：海洋单位距离的换算**

**05 实例2：打印名片**





# 开发第一个Python程序：模拟手机充值

当手机卡余额不足时，会收到运营商发来的提示短信，此时用户可在某充值平台上输入要充值的手机号码和金额进行充值。

### 话费充值

150000000000 

中国移动

---

话费充值

10元 售价9.99元	20元 售价19.97元	30元 售价29.96元
50元 售价49.93元	100元 售价99.85元	200元 售价199.7元



# 开发第一个Python程序：模拟手机充值



**思考：**

如何使用Python  
模拟以上场景呢？



# 开发第一个Python程序：模拟手机充值

在编写代码前，先思考以下3个问题：

1. 如何接收用户输入的手机号码、充值金额。
2. 如何保存输入的手机号码与充值金额。
3. 如何提示用户充值成功。

我们可以使用Python中的`input()`函数给出提示并接收用户输入的数据，使用变量保存用户输入的数据，使用`print()`函数输出提示信息。



# 开发第一个Python程序：模拟手机充值

示例：

```
phone_num = input('请输入要充值的手机号码：')  
recharge_amount = input('请输入要充值的金额：')  
print('手机号码' + phone_num + '成功充值' + recharge_amount + '元')
```

请输入要充值的手机号码：15000000000  
请输入要充值的金额：100  
手机号码15000000000成功  
充值100元



# 良好的编程约定

我们在开发程序时要重视其编写规范，使程序不仅能够  
在机器上正确执行，还便于调试、维护及阅读。

- PEP8是一份关于Python编码规范指南，遵守该规范能够帮助Python开发者编写出优雅的代码，提高代码可读性。
- <https://python.freelycode.com/contribution/detail/47>



# 良好的编程约定

- 请认真阅读!



软件下载及安装

首页 / Python编辑器文章 / PEP8中文版 -- Python编码风格指南

## PEP8中文版 -- Python编码风格指南

Python部落组织翻译, 禁止转载

### 目录

- 缩进
- 制表符还是空格?
- 行的最大长度
- 空行
- 源文件编码
- 导入
- 无法忍受的
- 其它建议
- 注释块
- 行内注释
- 文档字符串
- 根本原则
- 描述: 命名风格
- 规定: 命名约定

```
"""
```

敏感词过滤

思路: 给定一个字符串, 判断字符串中的文字是否在用户输入的数据中, 如果存在使用\*替换

```
"""
```

```
sensitive_character = '考试' # 敏感词库
```

```
test_sentence = input('请输入一段话:')
```

```
for line in sensitive_character: # 遍历输入的字符是存在敏感词库中
```

```
    if line in test_sentence: # 判断是否包含敏感词
```

```
        test_sentence = test_sentence.replace(line, '*')
```

```
print(test_sentence)
```



# 良好的编程约定

## • 代码布局

标准Python风格  
中每个缩进级别  
**使用4个空格**，不  
推荐使用Tab。

缩进

行最大长度79，  
换行可以使用反  
斜杠，但**建议使用  
圆括号**。

行的最大长度

顶层**函数和**定义的**类之  
间空两行**，类中的**方法  
定义之间空一行**；函数  
内**逻辑无关的代码段之  
间空一行**，其它地方尽  
量不要空行。

空白行

敏感词过滤

思路：给定一个字符串，判断字符串中的文字是否在用户输入的数据中，如果存在使用\*替换

```
"""
```

```
sensitive_character = '考试' # 敏感词库
```

```
test_sentence = input('请输入一段话:')
```

```
for line in sensitive_character: # 遍历输入的字符是存在敏感词库中
```

```
    if line in test_sentence: # 判断是否包含敏感词
```

```
        test_sentence = test_sentence.replace(line, '*')
```

```
print(test_sentence)
```





# 良好的编程约定

## • 空格的使用

1. 右括号前不要加空格。
2. 逗号、冒号、分号前不要加空格。
3. 函数的左括号前不要加空格。如`fun(1)`。
4. 序列的左括号前不要加空格，如`list[2]`。
5. 操作符左右各加一个空格，如`a + b = c`。
6. 不要将多个语句写在同一行。
7. `if`、`for`、`while`语句中的执行语句必须另起一行。



# 良好的编程约定

## • 代码注释

块注释的每行开头使用一个#和一个空格，缩进至与代码相同的级别。

块注释

行内注释与代码至少由两个空格分隔，注释以一个#和一个空格开头。

行内注释

文档字符串是为所有公共模块、函数、类以及方法编写的文档说明。

文档字符串

敏感词过滤

思路：给定一个字符串，判断字符串中的文字是否在用户输入的数据中，如果存在使用\*替换

```
"""
sensitive_character = '考试' # 敏感词库
test_sentence = input('请输入一段话:')
for line in sensitive_character: # 遍历输入的字符是存在敏感词库中
    if line in test_sentence: # 判断是否包含敏感词
        test_sentence = test_sentence.replace(line, '*')
print(test_sentence)
```



# 良好的编程约定

## • 命名规范

1. 不要使用字母“l”（小写的L）、“0”（大写的O）、“1”（大写的I）作为单字符变量名。
2. 模块名、包名应简短且全为小写。
3. 函数名应该小写，如果想提高可读性，可以用下划线分隔小写单词。
4. 类名首字母一般使用大写。
5. 常量通常采用全大写命名。



# 数据的表示——变量

变量





# 数据的表示——变量



**思考：**  
什么是变量？



# 数据的表示——变量



Python程序运行的过程中随时可能产生一些临时数据，程序会将这些数据保存在内存单元中，并使用不同的标识符来标识各个内存单元。这些具有**不同标识、存储临时数据的内存单元称为变量**，标识内存单元的符则为变量名（亦称标识符），内存单元中存储的数据就是变量的值。



# 数据的表示——变量

Python中定义变量的方式非常简单，只需要指定数据和变量名即可。

变量名 = 数据






# 数据的表示——变量

变量名由字母、数字和下划线组成，且不以数字开头。



name  
\_age  
\_\_color

1\_name  
2e  
012





# 数据的表示——变量

变量名区分大小写。

andy和Andy是  
不同的标识符

andy和Andy是  
相同的标识符





# 数据的表示——变量

变量名应通俗易懂，见名知意。

将表示姓名的  
变量命名为

name

将表示姓名的  
变量命名为a





# 数据的表示——变量

变量名若由两个以上单词组成：

get\_num  
set\_time  
print\_menu



getNum  
settime  
printMenu





# 基本输入输出

程序要实现人机交互功能，需能够向显示设备输出有关信息及提示，同时也要能够接收从键盘输入的数据。





# 输入函数

`input()`函数用于接收一个标准输入数据，该函数返回一个字符串类型数据。

**<变量> = input(<提示性字符串>)**

```
sname = input("请输入你的名字:")  
print(sname + ", 你好!")
```

如果输入的是“张三”，

显示：张三，你好！

```
>>>a = input("请输入一个小数： ")  
请输入一个小数： 123.456  
>>>print(a) # 此时a是字符串  
"123.456"  
123.456
```



# 输出函数

print()函数用于向控制台中输出数据。

三种用法

- 第一种，输出字符串或数值常量，使用方式如下：

**print(<待输出字符串>)**

```
>>>print("我爱伟大的祖国")  
我爱伟大的祖国  
>>>print(100)  
100
```





# print() 函数

## ■ 第二种，输出一个或多个变量

**print(<变量1>, [<变量2>], ..., [<变量n>])**  
连续输出多项，以**空格分隔**，然后换行

```
>>>value1 = 10
>>>value2 = 20
>>>value3 = 30
>>>print(value1, value2, value3)
10 20 30
```



# print()函数

## 第三种

`print(objects, end='<设定的结束符>')`

参数如下：

- `objects` -- 表示输出的对象。
- `end` -- 用于设定以什么结束，默认为 `\n`。



# print函数

```
print(x, y, z..., end=":")
```

连续输出多项，以逗号分隔，不换行

```
print(1, 2, 3, end=":")
```

```
print("4")
```

```
#>>1 2 3:4
```



# print函数

```
1. >>> for x in range(0,10):  
2.     print(x)  
3.  
4.  
5. 0  
6. 1  
7. 2  
8. 3  
9. 4  
10. 5  
11. 6  
12. 7  
13. 8  
14. 9
```

```
1. >>> for x in range(0,10):  
2.     print (x,end = '')  
3.  
4.  
5. 0123456789
```



# print函数

```
1 for x in range(0, 5):
2     print(x, end=' ')
3
4 print('\n')
5
6 for x in range(0, 5):
7     print(x, end=',')
8
9 ...
10 0 1 2 3 4
11
12 0,1,2,3,4,
13 ...
```

出现这种情况，  
是因为print()  
本身就是默认换  
行的，再加上换  
行符，相当于换  
行两次。

```
1 for x in range(0, 5):
2     print(x, end=' ')
3
4 print() #本身自带换行，
5
6 for x in range(0, 5):
7     print(x, end=',')
8
9 ...
10 0 1 2 3 4
11 0,1,2,3,4,
12 ...
```



# 过渡页



**01 Python概述**

**02 搭建Python开发环境**

**03 快速开发Python程序**

**04 实例1：海洋单位距离的换算**

**05 实例2：打印名片**



## 实例1：海洋单位距离的换算

公里是陆地上距离的计量单位，海里是海洋距离的计量单位，两者可以通过以下公式计算：

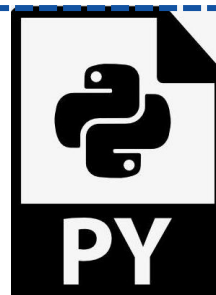
$$\text{海里} = \text{公里} / 1.852$$





# 实例1：海洋单位距离的换算

本实例要求编写程序，实现  
将海洋公里转为海里的换算。





# 目录

目录



**01 Python概述**

**02 搭建Python开发环境**

**03 快速开发Python程序**

**04 实例1：海洋单位距离的换算**

**05 实例2：打印名片**



## 实例2：打印名片

名片是标示姓名及其所属组织、公司单位和联系方法的纸片，也是新朋友互相认识、自我介绍的快速有效的方法。

传智播客教育科技有限公司

张先生      主管

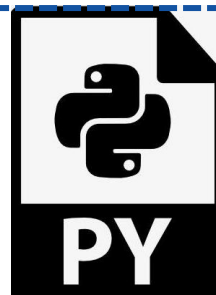
手机：18688888888

地址：北京昌平区建材城西路金燕龙办公楼



## 实例2：打印名片

本实例要求编写程序，模拟效果如图所示的名片。





# 本章小结

- 本章主要介绍了一些Python的入门知识，包括Python的特点、版本、应用领域、Python开发环境的搭建、编程规范、Python中的变量、输入输出函数等。
- 通过本章的学习，希望大家能够独立搭建Python开发环境，并对Python开发有一个初步认识，为后续学习做好铺垫。



# 作业

- 安装python环境（2种类型）
- 实践课件中的案例





# Thank You!

yx.boxuegu.com

