

教案（分教案）

课次： 1 学时： 2

章 节	第 1 章 Python 概述与输入输出 1.1 Python 概述 1.2 Python3.9.X 安装 1.3 PyCharm IDE 和 python 程序执行方法 1.4 代码规范 1.5 Python 输出 1.6Python 输入
教学目的和教学要求	了解 Python 的起源和发展历史，了解 Python 的应用领域，理解对于信息计算科学专业来说，学习 Python 的意义。掌握 Python3.9.X 的下载安装方法，并能够进行测试检验。掌握按照 Python 库的方法。熟悉 PyCharm2020 IDE 的工作界面，掌握创建项目和文件的方法，掌握 python 程序的各种执行方法。理解 python 语言的代码规范。掌握 Python 输入和输出，并能够根据要求编写简短的小案例。
教 学 重 点 难 点	教学重点： Python3.9.X 安装，PyCharm2020 IDE 和 Python 程序执行方法，代码规范，Python 输出，Python 输入。 教学难点： Python 输出，Python 输入。
教学方法和辅助手段	教学方法： 课堂讲解、案例分析，实际操作，提问讨论，反思质疑 辅助手段： 电脑、投影仪、教科书

课程之前的引言

学习本门课程的意义，课程的重要性，课程性质、教学目标，学习要求、学习技巧，参考资料，课程安排、成绩评定。

第 1 章 Python 概述与输入输出

Python 语言的学习，已经上升到国家战略的层面。以 AI——人工智能为核心代名词之一的第四次工业革命已经到来，Python 是最适合人工智能开发的语言。国家相关教育部门对“人工智能的普及”格外重视，将 Python 列入小学、中学和高中的教育体系中，为未来国家和社会的发展奠定了人工智能的人才基础，逐步由底层向高层推动“全面学 Python”，从而进一步推动人工智能技术和社会人才结构的更迭。

1.1 Python 概述

Python 的作者 Guido von Rossum 吉多·范罗苏姆，荷兰人，人称龟叔。

- Python 是龟叔在 1989 年在阿姆斯特丹过圣诞节期间，为了打发无聊的圣诞节而用 C 编写的一个编程语言；
- Python 的意思是蟒蛇，源于作者喜欢的一部电视剧——蒙提·派森的飞行马戏团（Monty Python’s Flying Circus）的爱好者；
- Python 在 1989 年创建，正式诞生于 1991 年，编译器是 C 语言实现。
- Python 的解释器如今有多个语言实现，常用的是 CPython（官方版本



的 C 语言实现），其他还有 Jython（可以运行在 Java 平台）、IronPython（可以运行在 .NET 和 Mono 平台）、PyPy（Python 实现的，支持 JIT 即时编译）











➤ Python 目前有两个版本，Python2 和 Python3，最新版分别为 2.7.14 和 3.9.X。

➤ **Life is short, you need Python. 人生苦短，我用 Python。**



TIOBE Index for August 2021¹

August Headline: Data Mining and AI languages are booming in the TIOBE index

Aug 2021	Aug 2020	Change	Programming Language		Ratings	Change
1	1			C	12.57%	-4.41%
2	3	▲		Python	11.86%	+2.17%
3	2	▼		Java	10.43%	-4.00%
4	4			C++	7.36%	+0.52%
5	5			C#	5.14%	+0.46%
6	6			Visual Basic	4.67%	+0.01%
7	7			JavaScript	2.95%	+0.07%
8	9	▲		PHP	2.19%	-0.05%
9	14	▲▲		Assembly language	2.03%	+0.99%
10	10			SQL	1.47%	+0.02%

2021 年 IEEE Spectrum 编程语言 Python 排名第一。

¹ <https://www.tiobe.com/tiobe-index/>

Choose a Ranking

IEEE Spectrum

Trending

Jobs

Open

Custom

Create Custom Ranking

Language Types

Web

Enterprise

Mobile

Embedded

(Click to hide)

Language Ranking: Jobs

Rank	Language	Type	Score
1	Python	Web	100.0
2	C	Mobile	96.5
3	Java	Web	96.1
4	C++	Mobile	89.3
5	Go	Web	88.5
6	JavaScript	Web	87.1
7	R	Mobile	82.2
8	HTML	Web	79.8
9	C#	Web	75.7
10	SQL	Mobile	73.7

Choose a Ranking

IEEE Spectrum

Trending

Jobs

Open

Custom

Create Custom Ranking

Language Types

Web

Enterprise

Mobile

Embedded

(Click to hide)

Language Ranking: Open

Rank	Language	Type	Score
1	Python	Web	100.0
2	Java	Web	93.5
3	JavaScript	Web	88.4
4	C++	Mobile	87.9
5	C	Mobile	87.9
6	HTML	Web	81.9
7	C#	Web	77.9
8	Go	Web	76.6
9	R	Mobile	74.0
10	Swift	Mobile	73.0

Python 语言的特点：

- ✧ 简单。Python 遵循“简单、优雅、明确”的设计哲学。
- ✧ 高级。Python 是一种高级语言，相对于 C，牺牲了性能而提升了编程人员的效率。它使得程序员可以不用关注底层细节，而把精力全部放在编程上。
- ✧ 面向对象。Python 既支持面向过程，也支持面向对象。
- ✧ 可扩展。可以通过 C、C++ 语言为 Python 编写扩充模块。
- ✧ 免费和开源。Python 是 FLOSS(自由/开放源码软件)之一，允许自由的发布软件的备份、阅读和修改其源代码、将其一部分自由地用于新的自由软件中。
- ✧ 边编译边执行。Python 是解释型语言，边编译边执行，跨平台好。
- ✧ 可移植。Python 能运行在不同的平台上。
- ✧ 动态语言。指数数据类型的检查是在运行时做。
- ✧ 丰富的库。Python 拥有许多功能丰富的库。
- ✧ 可嵌入性。Python 可以嵌入到 C、C++ 中，为其提供脚本功能。

Python 可以应用于众多领域，如：数据分析、人工智能、组件集成、网络服务、图像处理、数值计算和科学计算等众多领域。目前业内几乎所有大中型互联网企业都在使用 Python，如：Youtube、Dropbox、BT、Quora（中国知乎）、豆瓣、知乎、Google、Yahoo!、Facebook、NASA、百度、腾讯、汽车之家、美团等。

目前 Python 主要应用领域：

- ✧ 云计算：云计算最火的语言，典型应用 OpenStack。
- ✧ WEB 开发：众多优秀的 WEB 框架，众多大型网站均为 Python 开发，Youtube, Dropbox, 豆瓣。。。典型 WEB 框架有 Django、Tornado 和 Flask。其中的 Python + Django 架构，应用范围非常广，开发速度非常快，学习门槛也很低，能够帮助你快速的搭建起可用的 WEB 服务。

- ✧ **科学运算**：典型库 NumPy, SciPy, Matplotlib, Enthought librarys,pandas。
- ✧ **人工智能**：Python 在人工智能大范畴领域内的机器学习、神经网络、深度学习等方面都是主流的编程语言，得到广泛的支持和应用。
- ✧ **自动化运维**：这几乎是 Python 应用的自留地，作为运维工程师首选的编程语言，Python 在自动化运维方面已经深入人心，比如 Saltstack 和 Ansible 都是大名鼎鼎的自动化平台。
- ✧ **金融分析**：量化交易，金融分析，在金融工程领域，Python 不但在用且用的最多，而且重要性逐年提高。原因：作为动态语言的 Python，语言结构清晰简单，库丰富，成熟稳定，科学计算和统计分析都很牛逼，生产效率远远高于 c、c++、java，尤其擅长策略回测
- ✧ **图形 GUI**：PyQT、WxPython、TkInter。
- ✧ **数据科学**：在大量数据的基础上，结合科学计算、机器学习等技术，对数据进行清洗、去重、规格化和针对性的分析是大数据行业的基石。Python 是数据分析的主流语言之一。
- ✧ **网络爬虫**：也称网络蜘蛛，是大数据行业获取数据的核心工具。没有网络爬虫自动地、不分昼夜地、高智能地在互联网上爬取免费的数据，那些大数据相关的公司恐怕要少四分之三。能够编写网络爬虫的编程语言有不少，但 Python 绝对是其中的主流之一，其 Scripy 爬虫框架应用非常广泛。

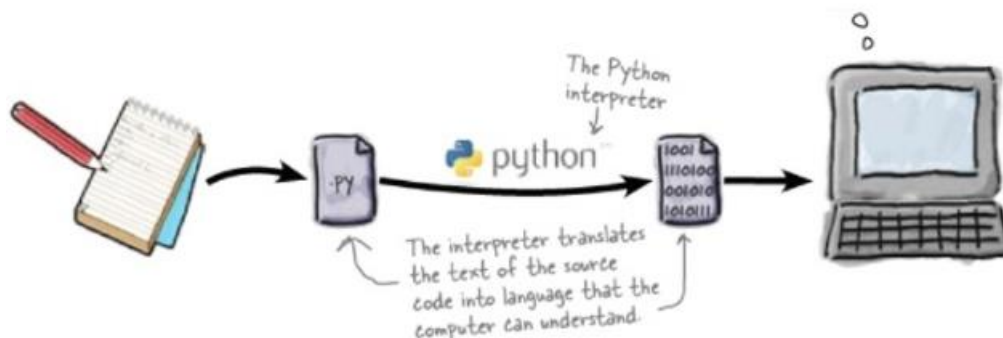
Python 在一些公司的应用：

- ✧ **谷歌**：Google App Engine 、code.google.com 、Google earth 、谷歌爬虫、Google 广告等项目都在大量使用 Python 开发
- ✧ **CIA**：美国中情局网站就是用 Python 开发的
- ✧ **NASA**：美国航天局(NASA)大量使用 Python 进行数据分析和运算
- ✧ **YouTube**：世界上最大的视频网站 YouTube 就是用 Python 开发的
- ✧ **Dropbox**：美国最大的在线云存储网站，全部用 Python，每天网站处理 10 亿个文件的上传和下载
- ✧ **Instagram**：美国最大的图片分享社交网站，每天超过 3 千万张照片被分享，全部用 python 开发
- ✧ **Facebook**：大量的基础库均通过 Python 实现的
- ✧ **Redhat**：世界上最流行的 Linux 发行版本中的 yum 包管理工具就是用 python 开发的
- ✧ **豆瓣**：公司几乎所有的业务均是通过 Python 开发的
- ✧ **知乎**：国内最大的问答社区，通过 Python 开发(国外 Quora)
- ✧ **春雨医生**：国内知名的在线医疗网站是用 Python 开发的

除上面之外，还有搜狐、金山、腾讯、盛大、网易、百度、阿里、淘宝 、土豆、新浪、果壳等公司在 使用 Python 完成各种各样的任务。

1.2、Python3.9.X 解释器与 Anaconda3 的下载安装

1、Python 解释器的下载与安装



下载地址：<https://www.python.org/downloads/windows/>，以 win10 系统为例，选择相应版本。

(1) 官方下载界面，**注意下载 64 位 Stable Releases**



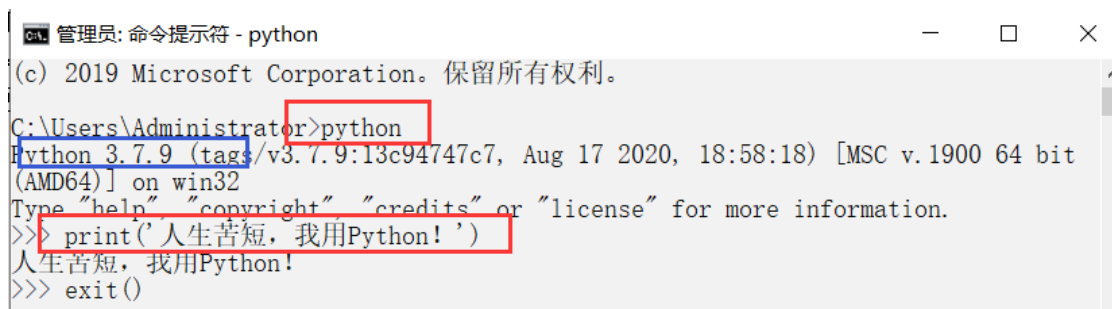
(2) 以 3.7.9 为例，3.9.7 类似。安装时选择 “Add Python3.7.9 to Path”



注意：安装时 python 加入到 windows 的环境变量中，如果忘记打勾，则需要手工加到环境变量中。

(3) 安装成功后测试

启动 cmd, 输入 python, 会列出当前安装的 python 版本信息。在 >>> 后输入 print("人生苦短, 我用 Python!")，回车执行，显示运行结果。**注意，Python 语言中所有标点符号均为英文状态下的标点符号。**

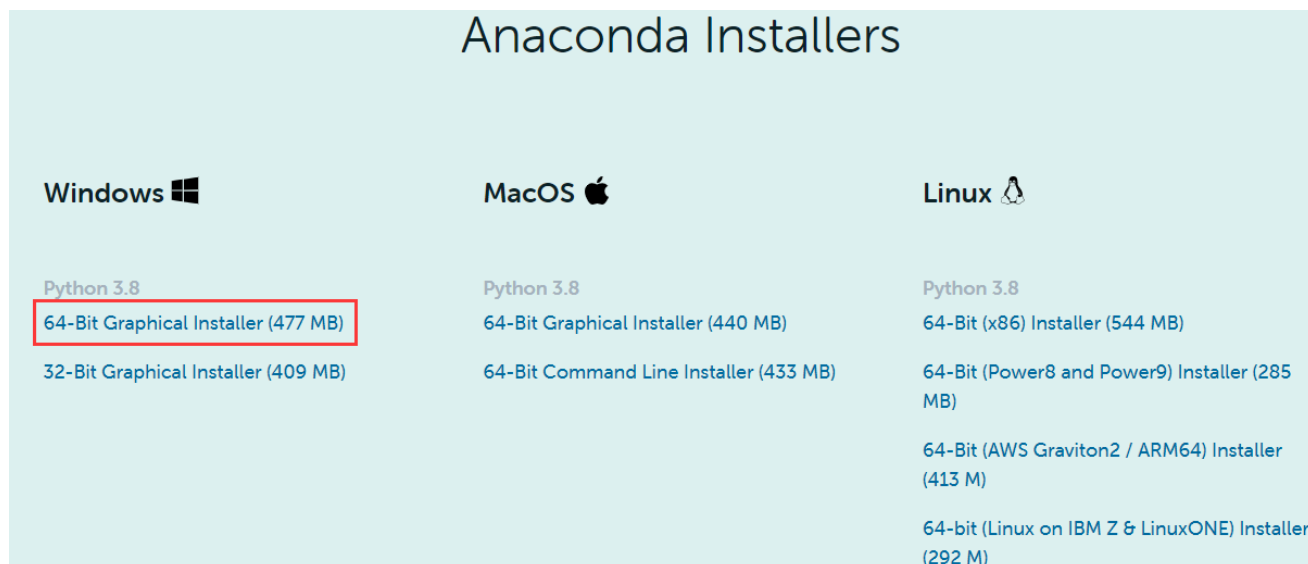


2、Anaconda3 的下载与安装

Anaconda 指的是一个开源的 Python 发行版本，其包含了 conda、Python 等 180 多个科学包及其依赖项。因为包含了大量的科学包，Anaconda 的下载文件比较大（约 477MB），如果只需要某些包，或者需要节省带宽或存储空间，也可以使用 Miniconda 这个较小的发行版（仅包含 conda 和 Python）。

Conda 是一个开源的包、环境管理器，可以用于在同一个机器上安装不同版本的软件包及其依赖，并能够在不同的环境之间切换。Anaconda 包括 Conda、Python 以及一大堆安装好的工具包，比如：numpy、pandas 等。Miniconda 包括 Conda、Python。

1、Anaconda 下载地址：<https://www.anaconda.com/download/>



安装过程请百度参考其安装步骤，这里省略。

2、配置环境变量

如果安装过程没有选择添加到环境变量，则 windows 系统：我的电脑->属性->高级系统配置->环境变量->Path->编辑->新建->将 Anaconda 的安装目录的 Scripts 文件夹，比如我的路径（默认安装路径）是 C:\ProgramData\Anaconda3\Scripts，看个人安装路径不同需要自己调整。之后就可以打开命令行（最好用管理员模式打开）输入 `conda --version`，验证。

```
管理员: 命令提示符
Microsoft Windows [版本 10.0.18362.53]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>conda --version
conda 4.7.10

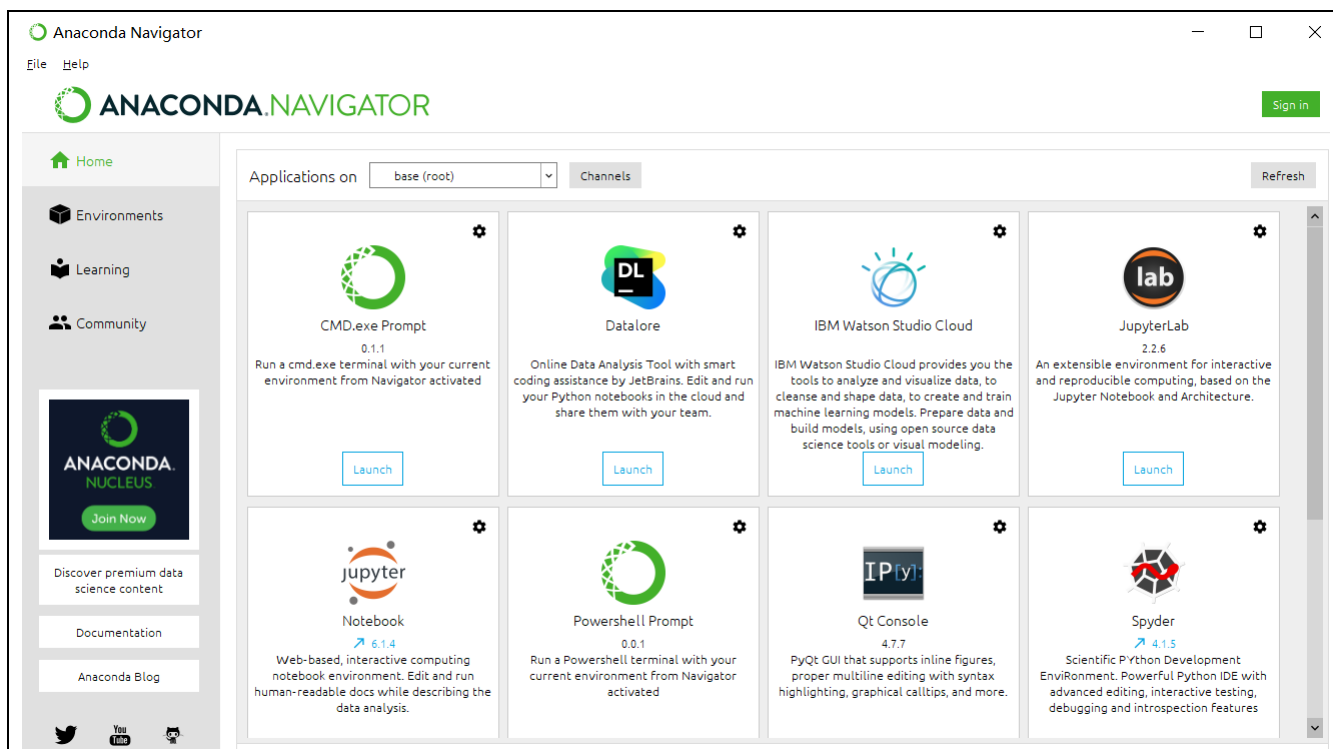
C:\Users\Administrator>
```

3、Anaconda 软件界面


Anaconda 软件中集成了很多开发工具，对于我们较为有用的是 Jupyter Notebook 和 Spyder。

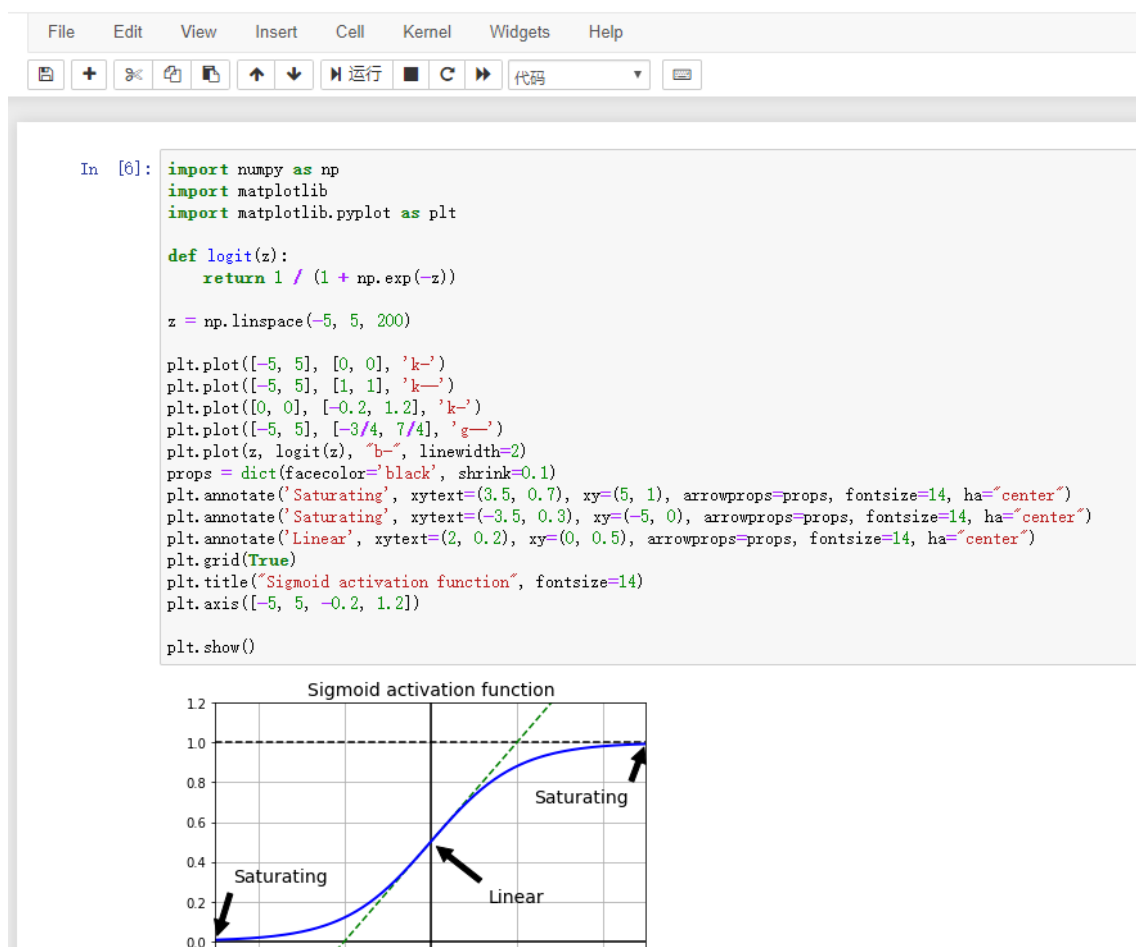
Jupyter Notebook 是基于网页的用于交互计算的应用程序。其可被应用于全过程计算：开发、文档编写、运行代码和展示结果。——Jupyter Notebook 官方介绍。

简而言之，Jupyter Notebook 是以网页的形式打开，可以在网页页面中直接编写代码和运行代码，代码的运行结果也会直接在代码块下显示。如在编程过程中需要编写说明文档，可在同一个页面中直接编写，便于作及时的说明和解释。



Jupyter Notebook 有两种键盘输入模式。编辑模式，允许你往单元中键入代码或文本；这时的单元框线是绿色的。命令模式，键盘输入运行程序命令；这时的单元框线是灰色。更多内容和使用方法在数据科学或机器学习章节介绍。

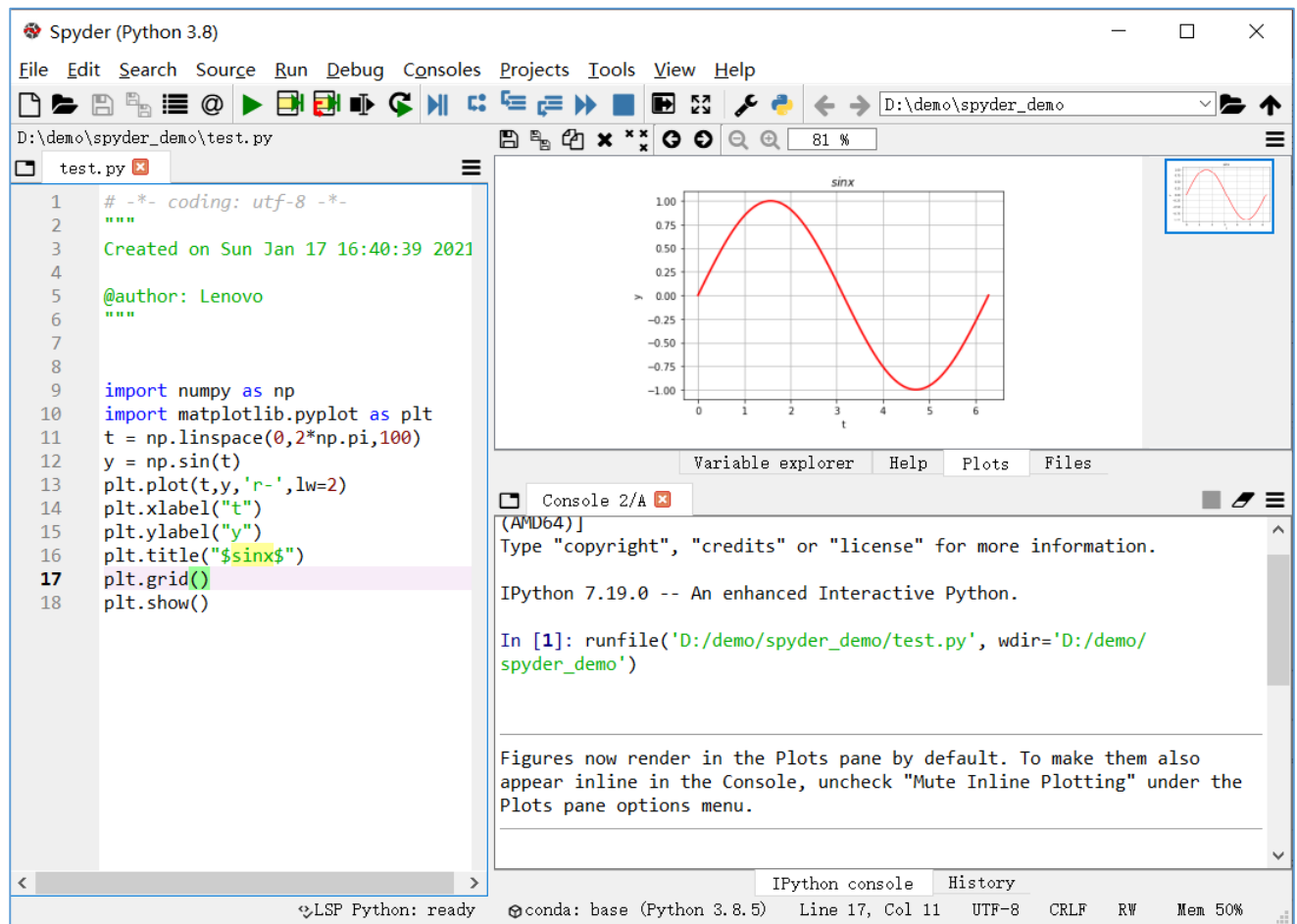
 jupyter Sigmoid_activation_function 最后检查 2 分钟前 (自动保存)



Spyder 是一个简单的集成开发环境。和其他的 Python 开发环境相比，它最大的优点就是模仿 MATLAB

的“工作空间”的功能，可以很方便地观察和修改数组的值。

Spyder 的界面由许多窗格构成，用户可以根据自己的喜好调整它们的位置和大小。当多个窗格出现在一个区域时，将使用标签页的形式显示。



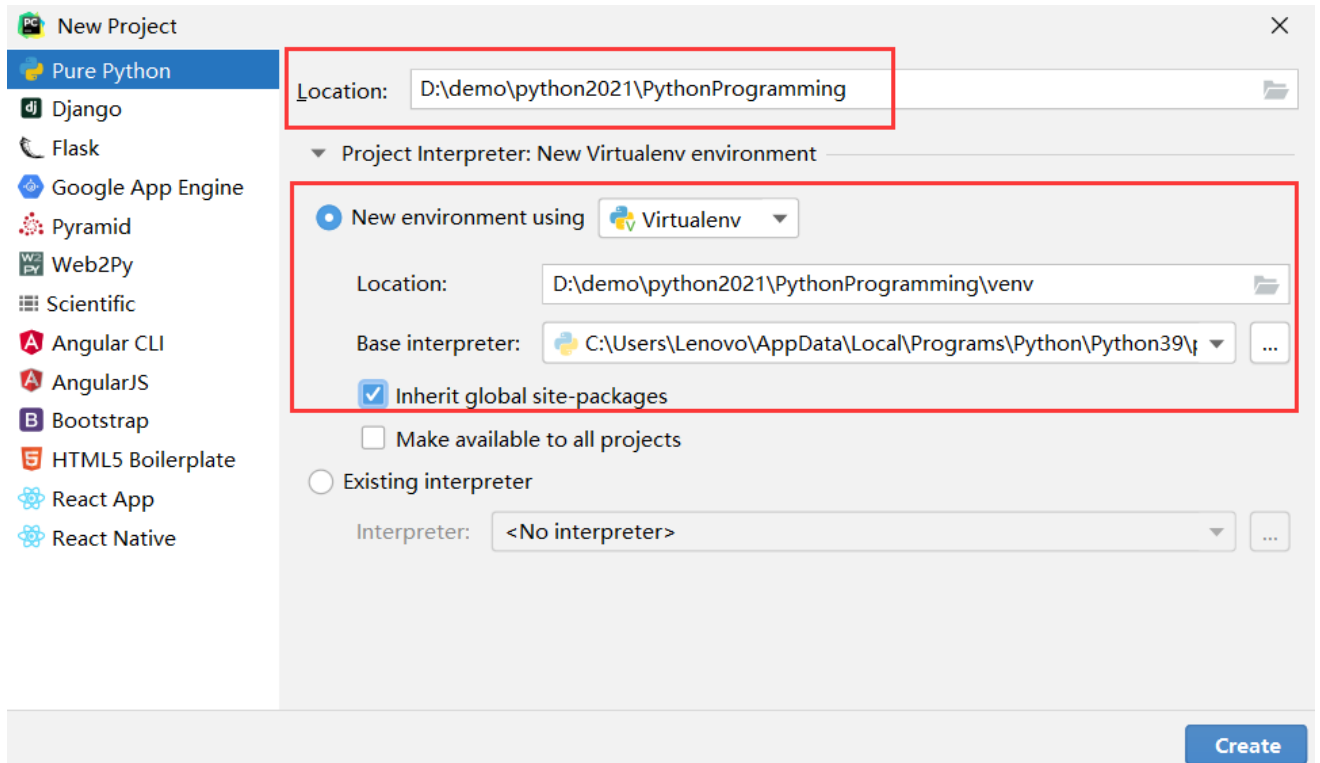
1.3 PyCharm2020 IDE 和 python 程序执行方法

PyCharm 是目前最流行最好用的一种 Python IDE，带有一整套可以帮助用户在使用 Python 语言开发时提高效率的工具，比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制。此外，该 IDE 提供了一些高级功能，以用于支持 Django 框架下的专业 Web 开发。

1、新建项目

- 在 Location 中填写项目路径、项目名。可在电脑除 C 盘（包括桌面）外的其他盘新建一个文件夹，如 python2021（一个有意义的文件名，不带中文和特殊符号），然后定位到该文件夹，并在后面输入项目名称，如 PythonProgramming（Python 命名，单词之间不加空格，不以数字或特殊符号开头，不加中文）。
- Project Interpreter: New Virtualenv environment, 项目解释器和新建虚拟环境。对于一个新安装的 pycharm 自然要设置此项内容。
 - new environment using 为项目创建一个新的环境，三个选项：Virtualenv、Pipenv 和 Conda
 - 1) Virtualenv Environment 虚拟环境，是一款工具，Pycharm 中集成了，用以创建独立的虚拟环境。Virtual Environment 主要解决的库依赖和版本依赖、以及间接授权等问题。它会创建一个独立的虚拟环境，可以使一个 Python 程序拥有独立的库 library 和解释器 interpreter，而不用与其他 Python 程序共享同一个 library 和 interpreter 程序，避免了不同 Python 程序间的互相影响，独立的使用一个 Python 解释器，不会与本地解释器产生影响。

- 2) **pipenv** 包管理模块。**pipenv** 可以说是 **virtualenv** 和 **pip** 的结合体，它不但会自动项目创建和管理 **virtualenv**（即第一个选项），而且在安装/卸载软件包时通过 **Pipfile** 文件自动添加/删除软件包，故对于管理包来说是非常便捷。**cmd** 环境下需命令安装 **pip install pipenv**。
- 3) **conda**:在 **conda** 环境下创建一个虚拟环境，**venv** 在 **anaconda** 下，项目文件在 **venv** 下，前提安装了 **Anaconda** 软件。



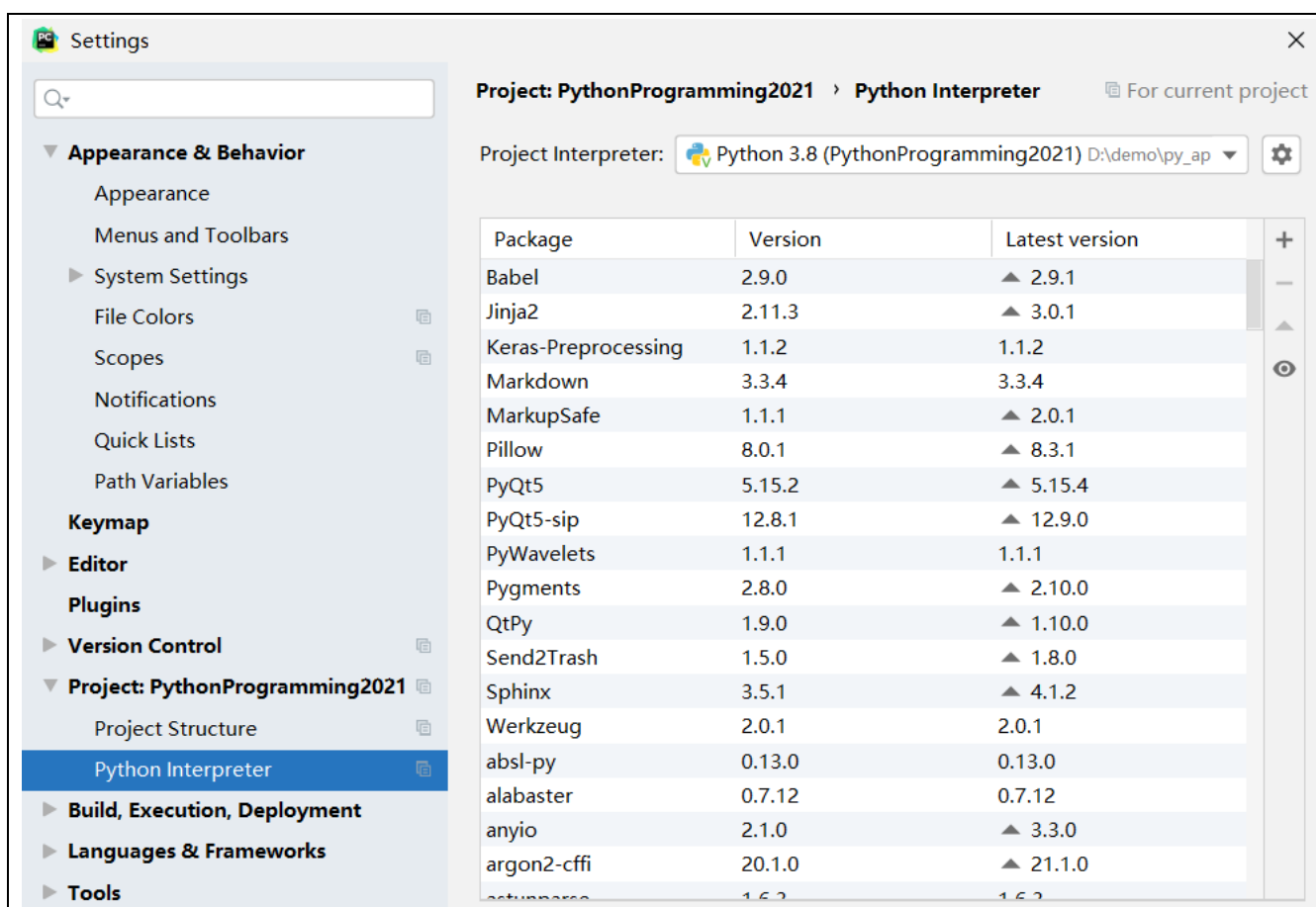
- 在 **Base interpreter** 下拉框中选择 **Python** 解释器；
- 勾选 **Inherit global site-packages** 可以使用 **base interpreter** 中的第三方库，不选将和外界完全隔离；尤其是通过 **pip install XXX** 安装 **python** 库后，勾选可以自动加载新安装的库，作为学习者建议勾选。
- 勾选 **Make available to all projects** 可将此虚拟环境提供给其他项目使用，勾选与否视情况而定。

勾选之后，默认选择了所有已经安装过的 **python** 库文件，对于一个实际的新建项目而言，不建议勾选，不同的项目所需的库是不同的，如果勾选，可能存在不兼容。但对于目前初学者，尤其是以数学、信息为内容载体的学习者，建议勾选。

如果不勾选，则项目中只有 **pip** 和 **setuptools** 两个 **Package**，如果需要增加，则可单击右侧“+”号搜索添加，也可以通过 **cmd** 环境下安装，命令 **pip install xxxxx**。

cmd 环境下安装包的方法：

```
C:\Users\Lenovo>pip install seaborn
Collecting seaborn
  Downloading seaborn-0.11.2-py3-none-any.whl (292 kB)
    |████████████████████████████████████████| 292 kB 595 kB/s
Requirement already satisfied: numpy>=1.15 in c:\users\lenovo\appdata\
from seaborn) (1.19.4)
Requirement already satisfied: matplotlib>=2.2 in c:\users\lenovo\app
es (from seaborn) (3.3.3)
Requirement already satisfied: scipy>=1.0 in c:\users\lenovo\appdata\l
rom seaborn) (1.5.4)
Requirement already satisfied: pandas>=0.23 in c:\users\lenovo\appdata
(from seaborn) (1.2.0)
```



常用的一些库：

(1) 科学计算与统计

- ✧ NumPy (Numerical Python) —— NumPy (Numerical Python 的简称) 是高性能科学计算和数据分析的基础包。它是几乎所有高级工具的构建基础。
- ✧ SciPy (Scientific Python) —— SciPy 基于 NumPy 开发，增加了用于科学处理的程序，例如积分、微分方程、额外的矩阵处理等。scipy.org 负责管理 SciPy 和 NumPy。
- ✧ StatsModels —— 为统计模型评估、统计测试和统计数据研究提供支持。

(2) 数据处理与分析：

Pandas 是基于 NumPy 数组构建的，使数据预处理、清洗、分析工作变得更快、更简单。Pandas 是一个开源的 Python 库，使用其强大的数据结构提供高性能的数据处理和分析工具。



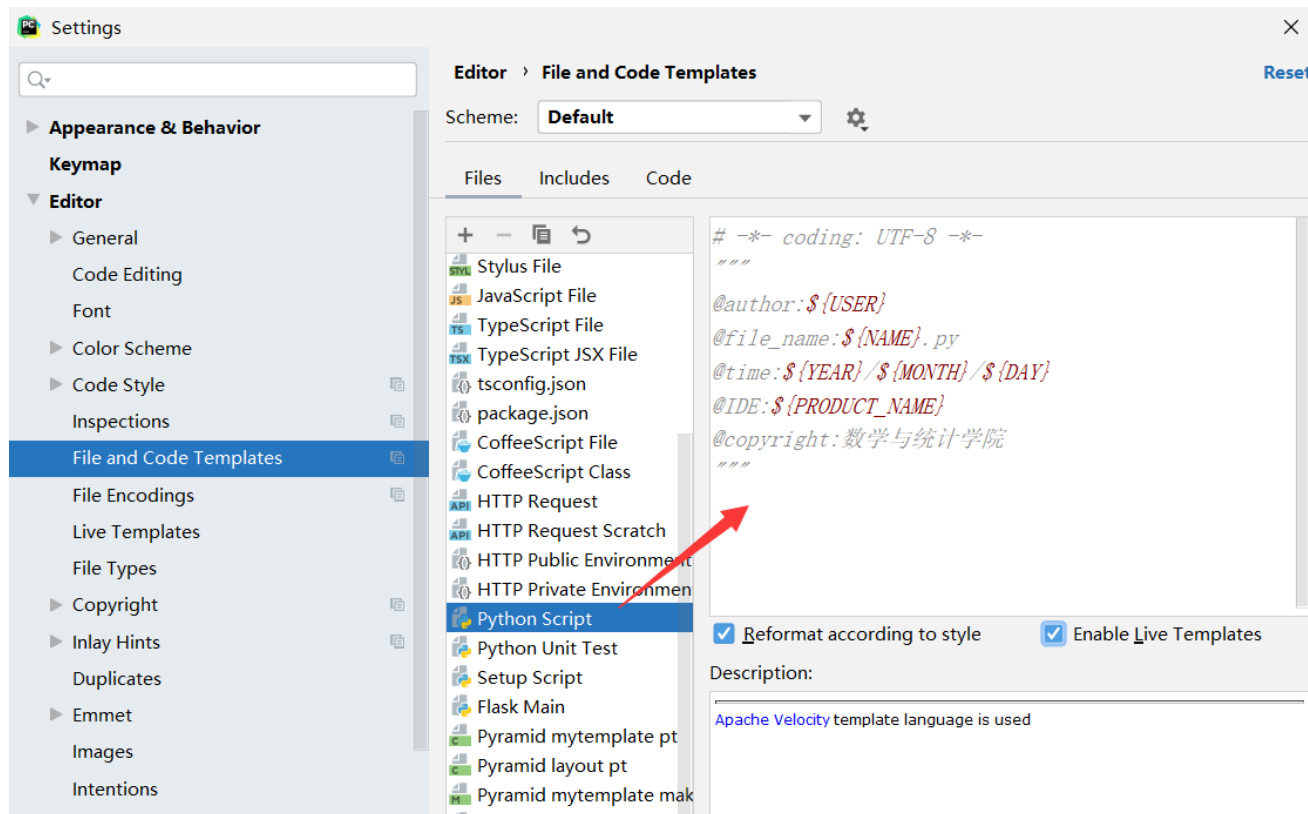
(3) 可视化

- ✧ Matplotlib —— 可高度定制的可视化和绘图库。Matplotlib 可以绘制正规图、散点图、柱状图、等高线图、饼图、网格图、极坐标图、3D 图以及添加文字说明等。
- ✧ Seaborn —— 基于 Matplotlib 构建的更高级别的可视化库。与 Matplotlib 相比，Seaborn 改进了外观，增加了可视化的方法，并且可以使用更少的代码创建可视化。

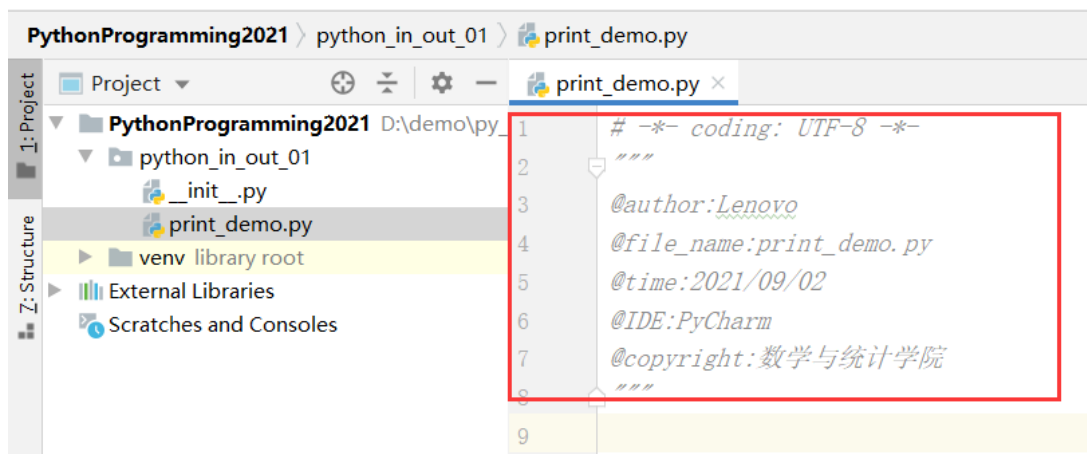
PyCharm 开发环境配置，其中作者等信息的设置。\${表达式}用来输出或者计算一个表达式的内容，即 EL (Expression Language)。内容如下：

```
# -*- coding: UTF-8 -*-
"""
```

```
@author:${USER}
@file_name:${NAME}.py
@time:${YEAR}/${MONTH}/${DAY}
@IDE:${PRODUCT_NAME}
@copyright:数学与统计学院
"""
```



编写的模板内容自动添加到代码窗口，如下：



2、新建包和 Python 文件

新建 Directory 和 Python package 的区别

pycharm 创建普通的 directory 和 package 时都是在硬盘上建立一个文件夹。

- ✧ Dictionary 在 pycharm 中就是一个文件夹，放置资源文件，对应于在进行 JavaWeb 开发时用于放置 css/js 文件的目录，或者说在进行物体识别时，用来存储背景图像的文件夹。该文件夹其中并不包含 `__init__.py` 文件
- ✧ 对于 Python package 文件夹而言，与 Dictionary 不同之处在于其会自动创建 `__init__.py` 文件。简

单的说，python package 就是一个目录，其中包括一组模块和一个 `__init__.py` 文件。**`__init__.py` 文件定义了包的属性和方法。其实它可以什么也不定义；可以只是一个空文件，但是必须存在。**如果 `__init__.py` 不存在，这个目录就仅仅是一个目录，而不是一个包，它就不能被导入或者包含其它的模块和嵌套包。

项目名称

Directory, data
存储数据文件

Directory, file
存储其他非文件

Package, 有意义的规范名称
包含该章节下所写的python文件

`__init__.py` 文件定义了包的属性和方法。其实它可以什么也不定义；可以只是一个空文件，但是必须存在。如果 `__init__.py` 不存在，这个目录就仅仅是一个目录，而不是一个包，它就不能被导入或者包含其它的模块和嵌套包。

```

PythonProgramming2021 > python_in_out_01 > print_demo.py
1  # -*- coding: UTF-8 -*-
2  """
3  @author:Lenovo
4  @file_name:print_demo.py
5  @time:2021/09/02
6  @IDE:PyCharm
7  @copyright:数学与统计学院
8  """
9
10
11 def print_example():
12     """
13     Python基础测试: 简单输出
14     :return:
15     """
16     print("人生苦短, 我用Python!")
17     hard_work = 1.01 ** 365 # 表示1.01的365次幂
18     print("每天努力一点点, 一年后的效果: ", hard_work)
19     lazy_work = 0.99 ** 365
20     print("每天颓废一点点, 一年后的效果: ", lazy_work)
21     print("不积跬步无以至千里!")
22
23
24 # 主函数调用
25 if __name__ == '__main__':
26     print_example()
  
```

代码内容:

```

# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:print_demo.py
@time:2021/09/02
@IDE:PyCharm
@copyright:数学与统计学院
"""

def print_example():
    """
    Python 基础测试: 简单输出
    :return:
    """

    print("人生苦短, 我用 Python!")
    hard_work = 1.01 ** 365 # 表示1.01的365次幂
    print("每天努力一点点, 一年后的效果: ", hard_work)
    lazy_work = 0.99 ** 365
  
```



```
print("每天颓废一点点，一年后的效果：", lazy_work)
print("不积跬步无以至千里！")
```

主函数调用

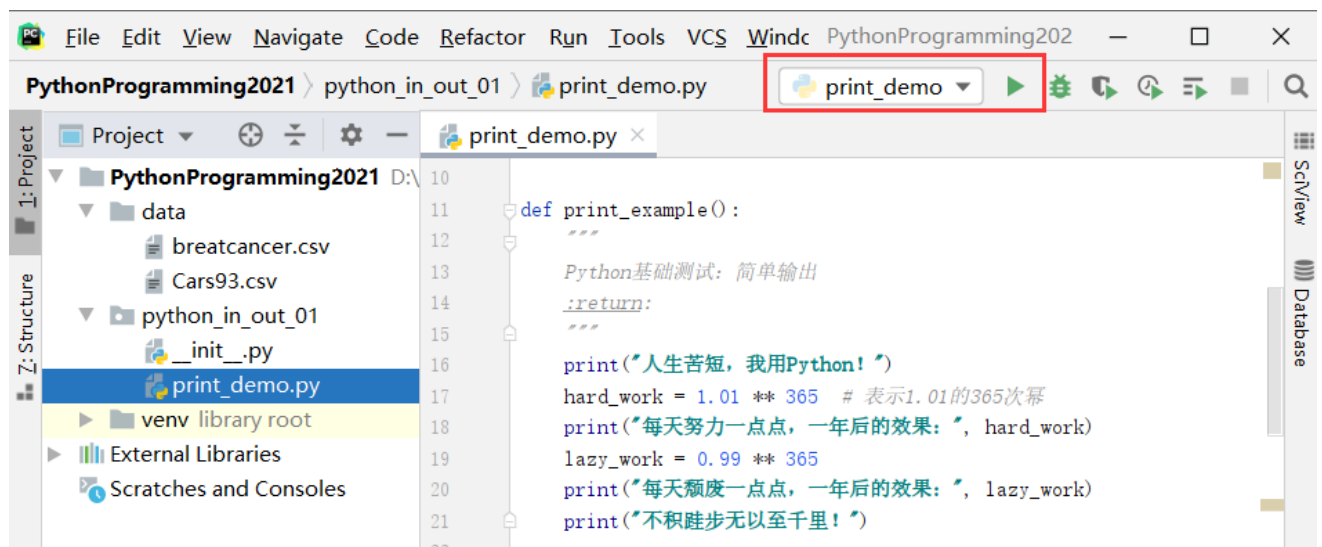
```
if __name__ == '__main__':
    print_example()
```

3、PyCharm 终端运行

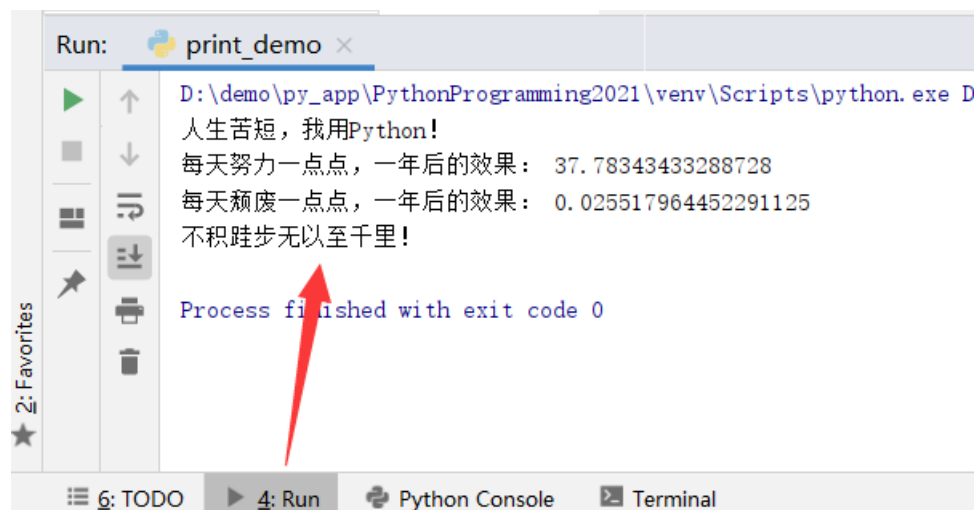
方式一：当前文件下单击鼠标右键，选择“Run print_demo”；

方式二：右上角选择要执行的 python 文件，单击类似于播放的按钮 run 程序

方式三：快捷键 shift+F10，前提是方式二中首先选择要运行的文件。



运行结果截图



4、Terminal 终端运行

默认路径为当前项目名称路径，需要通过 cd 命令切换到要执行的 Python 文件所在的目录下，然后通过命令 python 文件名+后缀名的方式调用。如 python print_demo.py。如果文件中包含有包之间的模块导入，则需要在导入前设置如下内容。假设项目文件存储路径：D:\demo\py_app\，则需输入：

```
import sys
sys.path.append(".././")
```

```
from data_interpolation_01.lagrange_interpolation import LagrangeInterpolation
```

NumericalAnalysis_ScientificCalculation > data_interpolation_01 > test > test_lagrange.py

Project Structure: Z: Structure

- NumericalAnalysis_ScientificCalculation
 - data_interpolation_01
 - test
 - __init__.py
 - test_aitken_stepwise_eps.py
 - test_aitken_stepwise_interp.py
 - test_gauss_interp.py
 - test_lagrange.py
 - test_newton_diff_quotient.py
 - test_newton_difference.py
 - test_runge.py
 - utils
 - __init__.py
 - aitken_stepwise_interp_eps.py
 - aitken_stepwise_interpolation.py
 - gauss_interpolation.py
 - lagrange_interpolation.py

File Content (test_lagrange.py):

```

1  @author:Lenovo
2  @file:test_lagrange.py
3  @time:2021/08/30
4
5  """
6
7
8  import numpy as np
9  import sys
10 sys.path.append("../..")
11 from data_interpolation_01.lagrange_interpolation import LagrangeInterpolation
12
13 if __name__ == '__main__':
14
15     x = np.linspace(0, 2 * np.pi, 10, endpoint=True)
16     y = np.sin(x)
17     x0 = np.array([np.pi / 2, 2.158, 3.58, 4.784])
18     # x = np.linspace(0, 24, 13, endpoint=True)
19     # y = np.array([12, 9, 9, 10, 18, 24, 28, 27, 25, 20, 18, 15, 13])

```

不存在包之间的导入问题，则直接调用：

Terminal: Local x +

Microsoft Windows [版本 10.0.19042.1165]
(c) Microsoft Corporation。保留所有权利。

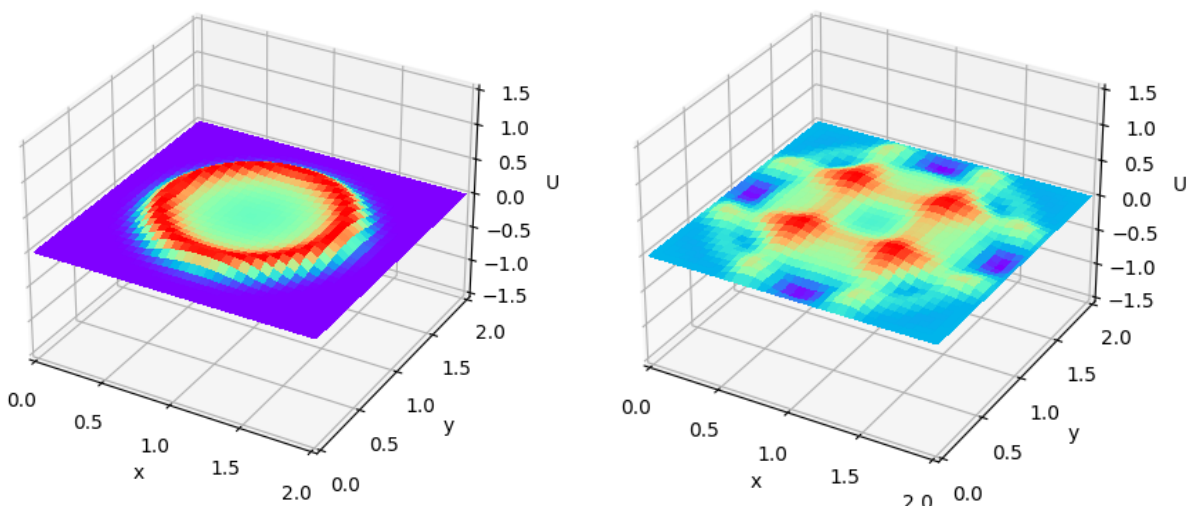
(venv) D:\demo\py_app\PythonProgramming2021>cd python_in_out_01

(venv) D:\demo\py_app\PythonProgramming2021\python_in_out_01>python print_demo.py

人生苦短，我用Python！
每天努力一点点，一年后的效果： 37.78343433288728
每天颓废一点点，一年后的效果： 0.025517964452291125
不积跬步无以至千里！

(venv) D:\demo\py_app\PythonProgramming2021\python_in_out_01>

有时需要演示动态效果，则使用此终端运行（如下为二维波动方程²动态演示过程中的两幅截图）：



² <https://zhuanlan.zhihu.com/p/111640958>

5、CMD 终端运行

注意：切换到当前运行文件的目录下。

```
命令提示符
Microsoft Windows [版本 10.0.19042.1165]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Lenovo>d:

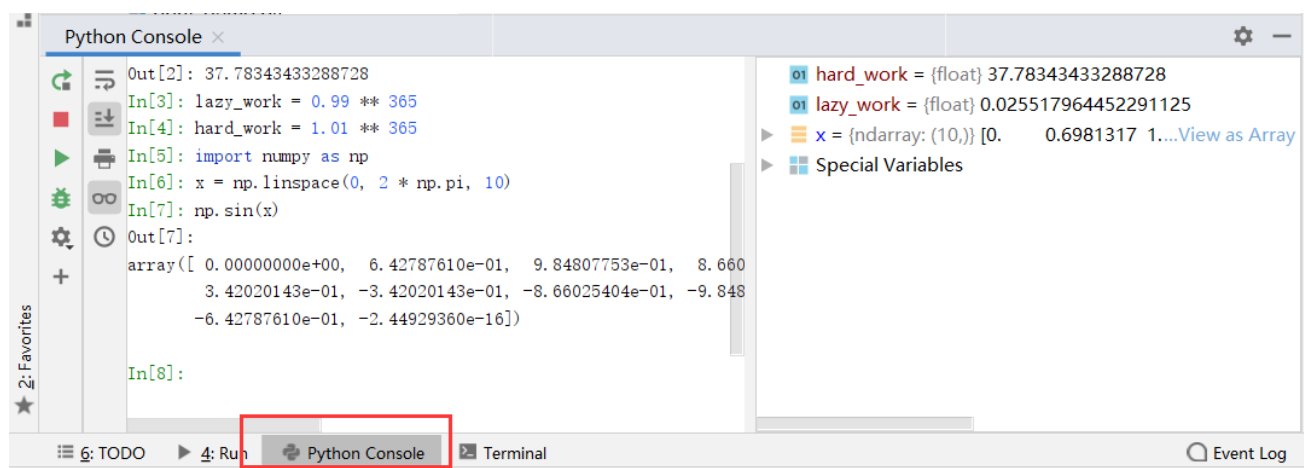
D:\>cd demo\py_app\PythonProgramming2021\python_in_out_01

D:\demo\py_app\PythonProgramming2021\python_in_out_01>python print_demo.py
人生苦短，我用Python！
每天努力一点点，一年后的效果： 37.78343433288728
每天颓废一点点，一年后的效果： 0.025517964452291125
不积跬步无以至千里！

D:\demo\py_app\PythonProgramming2021\python_in_out_01>_
```

6、Python Console 交互式环境

类似于 MATLAB 环境



1.4 Python 代码规范

1、代码缩进

pycharm 编辑器的缩进和取消缩进快捷键：

- 整体缩进：选中所需缩进的代码，键盘 **tab** 键
- 整体取消缩进：**shift+tab**

如果缩进不合理，则会出现错误，如 **IndentationError: expected an indented block;**

2、编码规范

- ◇ 每个 **import** 语句只导入一个模块，尽量避免一次导入多个模块；
- ◇ 不要在行尾添加分号“;”，也不用分号将两条命令放在同一行；
- ◇ 建议每行不超过 80 个字符，如果超过，建议使用小括号“()”将多行内容隐式的连接起来，而不推荐使用反斜杠“\”进行连接；

```
print("我一直认为我是一只蜗牛。我一直在爬，也许还没有爬到金字塔的顶端。"  
      "但是只要你在爬，就足以给自己留下令生命感动的日子。")
```

- ◇ 使用必要的空行可以增加代码的可读性。

3、空格

在二元运算符两边各空一格[=, -, +=, ==, >, in, is not, and]:

(1) 正确的写法 #注释后为不推荐的写法

```
i = i + 1    # i=i+1
submitted += 1    # submitted +=1
x = x * 2 - 1    # x = x*2 - 1
hypot2 = x * x + y * y    # hypot2 = x*x + y*y
c = (a + b) * (a - b)    # c = (a+b) * (a-b)
```

(2) 函数的参数列表中, 之后要有空格

正确的写法

```
def complex(real, imag):
```

```
    pass
```

不推荐的写法

```
def complex(real,imag):
```

```
    pass
```

(3) 函数的参数列表中, 默认值等号两边不要添加空格

正确的写法

```
def complex(real, imag=0.0):
```

```
    pass
```

不推荐的写法

```
def complex(real, imag = 0. 0):
```

```
    pass
```

(4) 左括号之后, 右括号之前不要加多余的空格

正确的写法: spam(ham[1], {eggs: 2})

不推荐的写法: spam(ham[1], { eggs : 2 })

(5) 字典对象的左括号之前不要多余的空格

正确的写法: dict['key'] = list[index]

不推荐的写法: dict ['key'] = list [index]

(6) 不要为对齐赋值语句而使用的额外空格

正确的写法

```
x = 1
y = 2
long_variable = 3
不推荐的写法
x           = 1
y           = 2
long_variable = 3
```

4、注释

为程序添加注释可以用来解释程序某些部分的作用和功能，提高程序的可读性。除此之外，注释也是调试程序的重要方式。在某些时候，不希望编译、执行程序中的某些代码，这时就可以将这些代码注释掉。

当然，添加注释的最大作用还是提高程序的可读性！很多时候，笔者宁愿自己写一个应用，也不愿意去改进别人的应用，没有合理的注释是一个重要原因。

虽然良好的代码可自成文档，但我们永远也不清楚今后读这段代码的人是谁，他是否和你有相同的思路。或者一段时间以后，自己也不清楚当时写这段代码的目的了。通常而言，合理的代码注释应该占源代码的 1/3 左右。

✧ 单行注释：若注释独占一行，#号顶头，空 1 格后写注释；若是行尾注释，空 2 格后#号再空 1 格写注释；

✧ 多行注释：三对双引号（推荐使用）和三对单引号；

复杂逻辑一定要写注释，除非这个项目就你一个人管一辈子。

建议：在代码的关键部分(或比较复杂的地方)，能写注释的要尽量写注释。比较重要的注释段，使用多个等号隔开，可以更加醒目，突出重要性。如：

案例：拉马努金圆周率公式

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!}{(k!)^4} \frac{1103 + 26390k}{396^{4k}}$$

参考代码如下，示例注释：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:cal_pi.py
@time:2021/09/02
@IDE:PyCharm
@copyright:http://maths.xynu.edu.cn
"""

import math
from decimal import *

getcontext().prec = 200 # 高精度计算，设置显示精度为 200 位

def calculate_pi(k):
    """
    拉马努金圆周率 pi 公式，计算给定项数的 pi 近似值
    :param k: 计算圆周率所需的累加次数 k
    :return: approximate_pi, 即满足项数 k 的 pi 近似值
    """
    pi = Decimal(0.0) # 圆周率变量，初始化为 0.0
    approximate_pi = Decimal(0.0) # 初始化最终计算的近似圆周率值
    coef = Decimal(2 * math.sqrt(2) / 9801) # 拉马努金公式系数
    # =====
    # 对于每一项，分别按照拉马努金公式循环计算 k 此，然后累加求解近似值
    # 其中 math.factorial() 为阶乘函数
```

```
# Decimal() 为高精度计算类
# 打印输出每次计算后的近似值
# =====
for i in range(k):
    # 拉马努金核心公式
    pi += Decimal(math.factorial(4 * i) / math.factorial(i) ** 4) * \
        (1103 + 26390 * i) / Decimal(396 ** (4 * i))
    approximate_pi = Decimal(1 / pi / coef) # 与系数乘积
    print(approximate_pi) # 打印输出
return approximate_pi

if __name__ == '__main__':
    calculate_pi(20)
```

运行结果，由于 200 位截图过大，这里仅仅截图 100 位的前几次 k 的累加近似值：

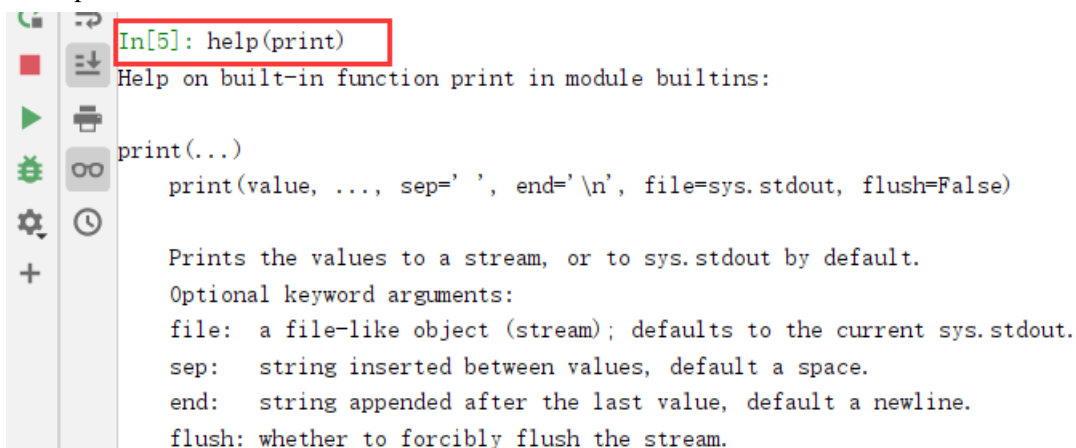
```
3.141592730013305208444717995203717126531013288517742636124601534477823338717090547987864722380077198
3.141592653589793426129638624818682080194037572702402616976775539537035496491276900042027404075451609
3.141592653589792786593381864215519871483022148624295014066782547419298914902482425239424035871266740
3.141592653589792786593376181792316246487136206826985965832052425760890937276308465178535950245539256
3.141592653589792786593376181792263857524825814547172016434118569136809324683810766037684629620405991
3.141592653589792786593376181792263857524331395793988225272579937332536986619798840293844991472040400
3.141592653589792786593376181792263857524331395789247213550000673202233641103788337103039886386149965
3.141592653589792786593376181792263857524331395789247213504012023490318900731463238898747214544357752
3.141592653589792786593376181792263857524331395789247213504012023040320984359753697766253436697830192
3.141592653589792786593376181792263857524331395789247213504012023040320979926477644538985578540740158
3.141592653589792786593376181792263857524331395789247213504012023040320979926477600624212822564049089
3.141592653589792786593376181792263857524331395789247213504012023040320979926477600624212385603972549
3.141592653589792786593376181792263857524331395789247213504012023040320979926477600624212385603968186
```

Python 学习文档网址：<https://docs.python.org/3.9/index.html>

1.5 python 输出

1、Python 输出的语法规则

使用 Python 的“帮助”：多数函数，需要导入包，再 help，如 help(np.linspace)。个人习惯，常在交互式命令窗口 help 函数的使用方法。



输出格式语法：**print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)**

参数的具体含义如下：

- ✧ `value` --表示输出的对象。输出多个对象时，需要用 `,`（英文逗号）分隔。
- ✧ `sep` -- 用来间隔多个对象。
- ✧ `end` -- 用来设定以什么结尾。默认值是换行符 `\n`，可以换成其他字符。
- ✧ `file` -- 要写入的文件对象。
- ✧ `flush` -- 用于控制输出缓存，该参数一般保持为 `False` 即可，这样可以获得较好的性能。

代码演示说明：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:print_format.py
@time:2021/09/03 10:04
@IDE:PyCharm
@copyright:数学与统计学院
"""

def print_syntax_format():
    """
    演示打印输出语法格式：
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
    :return:
    """

    user_name, user_age, user_motto = 'Charlie', 18, "人生苦短，我用 Python!" # 简单变量赋值
    # 1、同时输出多个变量和字符串
    print("读者名：", user_name, "， 年龄：", user_age, "， 箴言：", user_motto)

    # 2、同时输出多个变量和字符串，指定分隔符
    print("读者名：", user_name, "年龄：", user_age, "箴言：", user_motto, sep='||')

    # 3、设置 end 参数，指定输出之后不再换行，默认为换行\n
    print(85, "\t", end="") # 打印两个对象，一个数字，一个水平制表符\t
    print(88, "\t", end="")
    print(89, "\t", end="\n") # 输出完毕后，换行\n

    # 4、file 参数指定 print 函数的输出目标，默认为 sys.stdout，即系统标准输出
    fw = open("../file/poem.txt", 'w', encoding="utf-8") # 打开文件以便写入
    print("沧海月明珠有泪", file=fw) # 写入到文件
    print("蓝田日暖玉生烟", file=fw) # 写入到文件
    fw.close() # 写入完成，关闭文件

    fr = open("../file/poem.txt", 'r', encoding="utf-8") # 打开文件以便读取
    print(fr.read()) # 打印读取的文件内容
    fr.close() # 读入完毕，关闭文件
```

```
if __name__ == '__main__':  
    print_syntax_format()
```

运行结果:

读者名: Charlie , 年龄: 18 , 箴言: 人生苦短, 我用 Python!

读者名: ||Charlie||年龄: ||18||箴言: ||人生苦短, 我用 Python!

85 88 89

沧海月明珠有泪

蓝田日暖玉生烟

2、格式化输出

占位符: 在一个字符中占据着一个位置, 在输出时将这个位置替换成具体的内容。而占位符并不是随意地替任何内容占位, 它有着严格的规则。即, 每一个占位符只能替一种特定的类型占位。常见的占位符:

```
print('我叫%s, 身高%s' % (name,height))  ** 传入的值为元组
```

✧ %s : str(), 字符串

✧ %r: 非转移功能的字符串

✧ %c: 单个字符

✧ %d: 十进制整数, 一般会写成%nd, 其中 n 代表输出的总长度

✧ %i: 十进制整数

✧ %b: 二进制整数

✧ %o: 八进制整数

✧ %x : 十六进制整数

✧ %f 或 %F: 浮点型, 一般会写成%m.nf, 其中 m 代表总长度, n 代表小数点后几位

✧ %e 或 %E: 指数 (基地写为 e 或 E)

✧ %g 或 %G: 指数 (e 或 E) 或浮点数 (根据显示长度)

指定长度:

✧ %5d 右对齐, 不足左边补空格

✧ %-5d -代表左对齐, 不足右边默认补空格

✧ %05d 右对齐, 不足左边补 0

浮点数:

✧ %f 默认是输出 6 位有效数据, 会进行四舍五入

✧ 指定小数点位数的输出 %8f——保留小数点后 8 位

✧ '%10.8f' 10 代表整个浮点数的长度, 包括小数, 只有当字符串的长度大于 4 位才起作用

format, 特性: 字符串的 format 方法

✧ 顺序填坑: {}

```
print('姓名是:{}, 年龄是:{}'.format('Tom',20))
```

✧ 下标填坑

```
print('姓名是:{0}, 年龄是:{1}'.format('Tom',20))
```

✧ 变量填坑

```
print('姓名是:{name}, 年龄是:{age}'.format(name='Tom',age=20))
```

{:n} 指定输出长度 = n

字符串 {:5}--左对齐

数值 {:5}--右对齐

使用 >< 可以避免字符串 / 数值对齐方法不一致

✧ > 右对齐

✧ < 左对齐

✧ 中间对齐 ^ 不足的长度用*表示，则

```
print('姓名是:{0:*^11}\n 年龄是:{1:*^11}'.format('Tom',20))
```

代码演示说明：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:format_output.py
@time:2021/09/03 10:18
@IDE:PyCharm
@copyright:数学与统计学院
"""

import datetime as dt
import os
import math

def format_output_demo(msg):
    """
    代码演示格式化输出，根据参数 msg 的数据类型，执行不同的内容代码
    :param msg: 参数信息：整数、浮点数、字符串、列表元素或字典
    :return:
    """

    if isinstance(msg, int):
        print('这是一个整数: %d' % msg) # %d 位十进制, %o 是八进制, %x 是十六进制
        print('长度为 8, 实际长度不足, 前面补空格: %8d' % msg) # %nd 位十进制, n 表示显示长度
        print('—代表左对齐, 不足右边默认补空格: %-8d%s' % (msg, ".")) # %nd 位十进制, n 表示显示长度
        print("对应八进制为: %o, 十六进制: %x" % (msg, msg))
        print("下标填坑: {0:<8} | {1:>10} | {2:^15} | {0:<5}".format(msg, msg ** 2, msg ** 3, msg)) # 下标填坑
    elif isinstance(msg, float):
        # %10.2f——10 代表浮点数长度，包括小数点位数，且保留 2 位小数
        print("这是一个浮点数: %10.2f" % msg)
        print("这是一个浮点数，科学计数法: %.5e" % (math.sqrt(msg) / 10**8))
        num = 12.5
        # str.rjust(n)的作用是将字符串靠右对齐，参数 n 表示长度。center 表示居中对齐，rjust 表示右对齐
        print("手动拼接格式化: ", str(num).ljust(8), str(num**2).center(12), str(num**3).rjust(12), sep="|")
    elif isinstance(msg, str):
        # %-10.2s——左对齐，占位 10，截取两位字符串
        print('这是一个字符串: %s' % msg)
        print("输出字符串第一个字符: %c" % msg[0])
        print('这是一个字符串（非转义功能）: %r' % msg)
```

```
elif isinstance(msg, dict):
    print("传入参数为字典结构，内容如下：")
    # 指定占位符宽度，且是居中对齐，<左对齐，>右对齐，^居中对齐
    # print("{:~10}".format("name"), "{:~10}".format("age"), "{:~12}".format("height"))
    titlestr = ["name", "age", "height"] # 列表变量
    # print("{0:*~10} {1:*~10} {2:*~12}".format(*titlestr)) # 不足的长度用*表示
    print("{0:~10} {1:~10} {2:~12}".format(*titlestr)) # 补齐字符：默认空格
    for m in msg:
        print(" %-10s%-10d%-10.2f" % (msg[m]["name"], msg[m]["age"], msg[m]["height"]))

def file_print(msg):
    """
    文件信息简单的输入与输出
    :param msg: 文件内容
    :return:
    """
    fp = open(r'../file/mr.txt', 'a+') # 打开文件，进行追加
    print(msg, file=fp)
    print("文件所包含的字节数：%d" % fp.tell()) # 估计是读取有多少个字节
    fp.seek(os.SEEK_SET) # 重新定义文件位置指针，表示文件开头
    print("文件中的内容为：")
    print(fp.read()) # 读取并打印文件的内容
    fp.close()

if __name__ == '__main__':
    a_int, b_float = 100, 15.54328 # 定义一个整数和一个浮点数
    str_msg = "go big \t or go home" # 定义一个字符串
    format_output_demo(a_int) # 传参一个整数
    print('-' * 60)
    format_output_demo(a_int * b_float) # 传参一个浮点数
    print('-' * 60)
    format_output_demo(str_msg) # 传参一个字符串
    print('-' * 60)
    str_msg2 = "\u751f\u5316\u5371\u673a"
    str_msg3 = "\u4e2d\u56fd"
    format_output_demo(str_msg2 + str_msg3) # 传参一个 unicode 形式字符串
    print('-' * 60)
    dict_msg = {"1": {'name': '风清扬', 'age': 95, 'height': 186.0},
                "2": {'name': '令狐冲', 'age': 23, 'height': 188.5},
                "3": {'name': '任我行', 'age': 65, 'height': 180.7}}
    format_output_demo(dict_msg)
    print('-' * 60)
    print('当前年份：' + str(dt.datetime.now().year))
    print('当前日期时间：' + dt.datetime.now().strftime('%y-%m-%d %H:%M:%S'))
    print('-' * 60)
```

```
file_msg = input('您想在文件中存储什么内容：')
file_print(file_msg)
```

运行结果：

这是一个整数：100

长度为 8，实际长度不足，前面补空格：100

-代表左对齐，不足右边默认补空格：100 .

对应八进制为：144，十六进制：64

下标填坑：100 | 10000 | 1000000 | 100

这是一个浮点数：1554.33

这是一个浮点数，科学计数法：3.94250e-07

手动拼接格式化：|12.5 | 156.25 | 1953.125

这是一个字符串：go big or go home

输出字符串第一个字符：g

这是一个字符串（非转义功能）：'go big \t or go home'

这是一个字符串：生化危机中国

输出字符串第一个字符：生

这是一个字符串（非转义功能）：'生化危机中国'

传入参数为字典结构，内容如下：

name	age	height
风清扬	95	186.00
令狐冲	23	188.50
任我行	65	180.70

当前年份：2021

当前日期时间：21-09-03 10:53:22

您想在文件中存储什么内容：大江东去，浪淘尽，千古风流人物。

文件所包含的字节数：34

文件中的内容为：

大江东去，浪淘尽，千古风流人物。

1.6 python 输入

input 函数用于向用户生成一条提示，然后获取用户输入的内容。由于 input 函数总会将用户输入的内容放入字符串中，因此用户可以输入任何内容，input 函数总是返回一个字符串。

案例说明：

```
# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:input_format.py
@time:2021/09/03 10:55
@IDE:PyCharm
@copyright:数学与统计学院
```

```
"""

import datetime as dt

def input_test(sex, birth_year, height, weight):
    """
    根据输入的出生年月，计算年龄并判断属相，然后根据年龄，判断是未成年人、青年人、中年人还是老年人
    根据体重和身高计算 BMI 指数，并判断 BMI 指数是否在合理范围，并根据性别推算出最理想体重。
    :param sex: 性别
    :param birth_year: 出生年
    :param height: 身高
    :param weight: 体重
    :return:
    """

    now_year = dt.datetime.now().year # 获取当前时间的年份
    age = now_year - birth_year # 年龄
    zodiacs = ['猴', '鸡', '狗', '猪', '鼠', '牛', '虎', '兔', '龙', '蛇', '马', '羊']
    # zodiacs = ['鼠', '牛', '虎', '兔', '龙', '蛇', '马', '羊', '猴', '鸡', '狗', '猪']
    print("您的年龄是: %d 岁, 属相: %s" % (age, zodiacs[birth_year % 12]))

    # 1、根据年龄判断所属年龄段人群
    if age < 18:
        print("您现在是未成年人 q(≧▽≦q)")
    elif 18 <= age < 66:
        print("您现在是青年人(●~▽~●)")
    elif 66 <= age < 80:
        print("您现在是中年人(๖๑๑๑)/")
    else:
        print("您现在是老年人(๖•_•๖)")

    # 2、根据身高体重计算 BMI 指数，并判断是否健康
    bmi = weight / (height * height) * 100 ** 2 # 计算 BMI 指数
    # 判断身材是否合理
    if bmi < 18.5:
        print("您的 BMI 指数为: %.2f" % bmi) # 输出 BMI 指数
        print("体重过轻 ~@_@~")
    elif bmi >= 18.5 and bmi < 24.9:
        print("您的 BMI 指数为: %.2f" % bmi) # 输出 BMI 指数
        print("正常范围, 注意保持 (-_-)")
    elif bmi >= 24.9 and bmi < 29.9:
        print("您的 BMI 指数为: %.2f" % bmi) # 输出 BMI 指数
        print("体重过重 ~@_@~")
    elif bmi >= 29.9:
        print("您的 BMI 指数为: %.2f" % bmi) # 输出 BMI 指数
        print("肥胖 ~@_@~")
```



```

print("-" * 40)
print("根据美国有关医学统计：BMI 小于 16 的人和 BMI 大于 30 的人死亡率最高。")

# 3、根据性别和身高，计算理想体重
if sex == "女":
    weight = 19 * (height / 100) ** 2
elif sex == "男":
    weight = 22 * (height / 100) ** 2
print("最理想的体重：%.2fKG" % weight)
return None

if __name__ == '__main__':
    birth_year = int(input("1、请输入您的出生年份："))
    sex = input("2、请输入您的性别（男/女）：")
    height = float(input("3、请输入您的身高（CM）：")) # 输入身高
    weight = float(input("4、请输入您的体重（KG）：")) # 输入体重
    print("-" * 50)
    input_test(sex, birth_year, height, weight)

```

运行结果：

- 1、请输入您的出生年份：2000
- 2、请输入您的性别（男/女）：男
- 3、请输入您的身高（CM）：182
- 4、请输入您的体重（KG）：71

您的年龄是：21 岁，属相：龙
 您现在是青年人(●~▼~●)
 您的 BMI 指数为：21.43
 正常范围，注意保持（-_-）

根据美国有关医学统计：BMI 小于 16 的人和 BMI 大于 30 的人死亡率最高。
 最理想的体重：72.87KG

案例：菜单与信息的输出问题：

```

def menu():
    print('\t\t地铁来了')
    print('='*34)
    print('\t\t 主要功能菜单      +')
    print('='*34)
    print('+ 1 切换城市                +')
    print('+ 2 实时位置                  +')
    print('+ 3 换乘引导                    +')
    print('+ 4 首末车时刻                +')
    print('+ 5 站点查询                    +')
    print('+ 6 地铁线路                    +')
    print('+ 7 周边站点                    +')

```

地铁来了

```

=====
+          主要功能菜单          +
=====
+ 1 切换城市                +
+ 2 实时位置                  +
+ 3 换乘引导                    +
+ 4 首末车时刻                +
+ 5 站点查询                    +
+ 6 地铁线路                    +
+ 7 周边站点                    +
+ 8 增加城市                    +
+ 9 退出系统                    +
=====

```

说明：通过数字或向上向下键选择菜单

版本编号：2019-07 2

```

print('+ 8 增加城市          +')
print('+ 9 退出系统          +')
print('='*34)
print('说明：通过数字或向上向下键选择菜单')
print('='*34)
print("                      版本编号：2019-07 2")

```

```

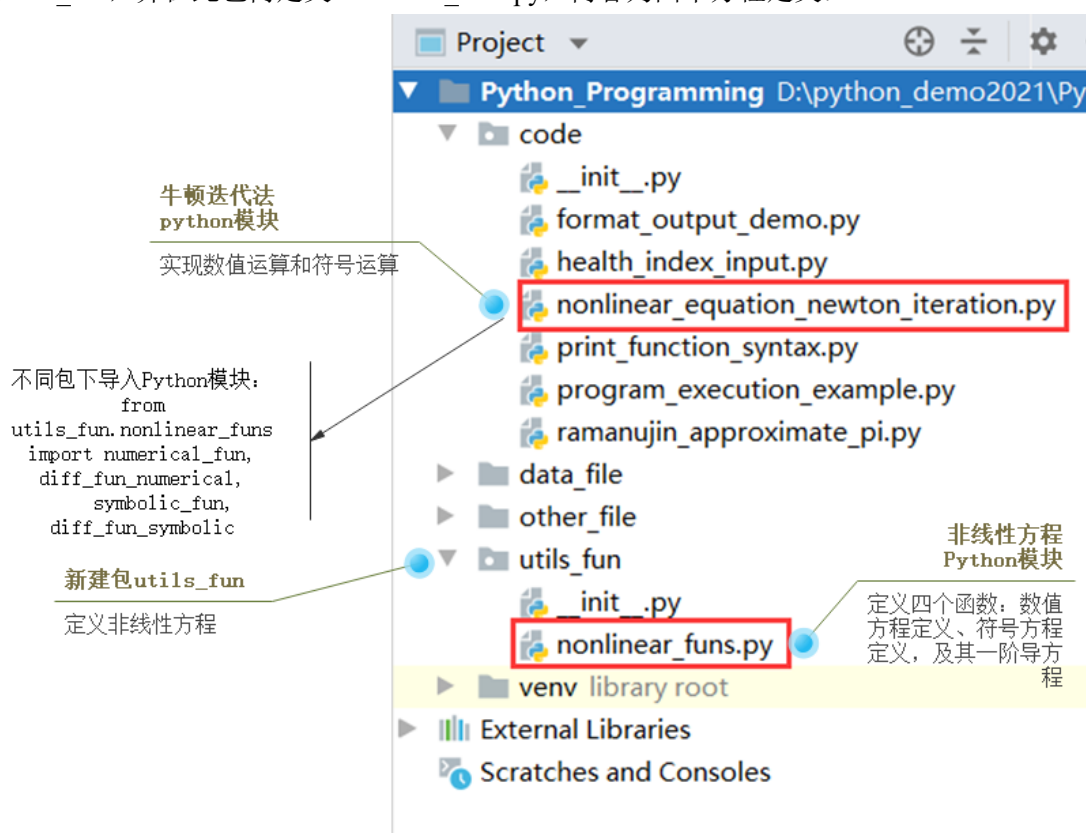
if __name__ == '__main__':
    memu()

```

案例：牛顿迭代法，求解方程如下：

$$2e^{-x} \sin x + 2 \cos x - 0.25 = 0$$

1. 新建包 `utils_fun`，并在此包内定义 `nonlinear_funs.py`，内容为四个方程定义：



```

import math
import sympy

```

```

def numerical_fun(x):

```

```

    """

```

```

    定义非线性方程，数值方程

```

```

    :param x: 自变量

```

```

    :return:

```

```

    """

```

```

    return 2 * math.exp(-x) * math.sin(x) + 2 * math.cos(x) - 0.25

```

```

def diff_fun_numerical(x):
    """
    非线性方程的一阶导函数，数值方程
    :param x: 自变量
    :return:
    """
    return 2 * math.exp(-x) * (math.cos(x) - math.sin(x)) - 2 * math.sin(x)

def symbolic_fun():
    """
    定义非线性方程，符号方程
    :return:
    """
    x = sympy.Symbol("x") # 定义符号变量，自由变量
    return 2 * sympy.exp(-x) * sympy.sin(x) + 2 * sympy.cos(x) - 0.25

def diff_fun_symbolic():
    """
    非线性方程的一阶导函数，符号方程
    :return:
    """
    return symbolic_fun().diff()

```

2. 牛顿迭代法求解非线性方程：数值运算和符号运算³两种方式

```

# -*- coding: UTF-8 -*-
"""
@author:Lenovo
@file_name:nonlinear_equations_newton_iteration.py
@time:2021-09-03 14:15
@IDE:PyCharm
@copyright: http://maths.xynu.edu.cn
"""
# 如何从另外一个包中导入所需要的Python 模块
from utils_fun.nonlinear_funs import numerical_fun, diff_fun_numerical, \
    symbolic_fun, diff_fun_symbolic
import sympy

# 符号运算
def symbolic_newton_iteration(x0, eps=1e-10, max_iter=100):
    """
    牛顿迭代法求解非线性方程组的解，采用符号运算解法
    :param x0: 初值
    :param eps: 近似解的精度

```

³ 符号运算函数文档：<https://docs.sympy.org/latest/modules/plotting.html>

```

:param max_iter: 最大迭代次数
:return: 近似解
"""

fh, dfh = symbolic_fun(), diff_fun_symbolic() # 符号方程, 以及一阶导函数
x = fh.free_symbols.pop() # 返回集合 {x, y, z}, {x}
sympy.plot(fh, (x, 0, 2)) # 符号方程图像可视化
print("%5s %20s %22s" % ("Iter", "ApproximateRoot", "ErrorPrecision"))
x_next, iter, tol = x0, 0, 1 # x(k+1) 初始化, iter 迭代变量初始化, 精度初始化
for iter in range(max_iter): # [0, max_iter-1]
    x_before = x_next # 近似解不断更替, 直至收敛为止
    # 牛顿迭代核心公式, 其中 x_next 表示 x(k+1)
    fx = fh.evalf(subs={x: x_before}) # evalf 是符号运算求值函数
    dfx = dfh.evalf(subs={x: x_before}) # 符号运算
    x_next = x_before - fx / dfx
    tol = abs(x_next - x_before) # 两次迭代的近似解的差的绝对值小于给定精度
    # tol = fun(x_next) # 把 x_next 代入方程
    print("%3d %22.15f %25.15e" % (iter + 1, x_next, tol))
    # 所求近似解满足精度要求, 则退出循环
    if abs(tol) <= eps:
        break

if iter < max_iter:
    print("方程在满足精度" + str(eps) + "的要求下, 近似解为: " + str(x_next) +
          ", 精度: " + str(tol))
else:
    print("牛顿迭代法求解数值逼近, 已经达到最大迭代次数, 可能不收敛或精度过高.....")

# 数值运算
def numerical_newton_iteration(x0, eps=1e-10, max_iter=100):
    """
    牛顿迭代法求解非线性方程组的解, 采用数值运算解法
    :param x0: 初值
    :param eps: 近似解的精度
    :param max_iter: 最大迭代次数
    :return: 近似解
    """

    print("%5s %20s %22s" % ("Iter", "ApproximateRoot", "ErrorPrecision"))
    x_next, iter, tol = x0, 0, 1 # x(k+1) 初始化, iter 迭代变量初始化, 精度初始化
    for iter in range(max_iter): # [0, max_iter-1]
        x_before = x_next # 近似解不断更替, 直至收敛为止
        # 牛顿迭代核心公式, 其中 x_next 表示 x(k+1)
        x_next = x_before - numerical_fun(x_before) / diff_fun_numerical(x_before) # 数值运算
        tol = abs(x_next - x_before) # 两次迭代的近似解的差的绝对值小于给定精度
        # tol = fun(x_next) # 把 x_next 代入方程
        print("%3d %22.15f %25.15e" % (iter + 1, x_next, tol))
        # 所求近似解满足精度要求, 则退出循环

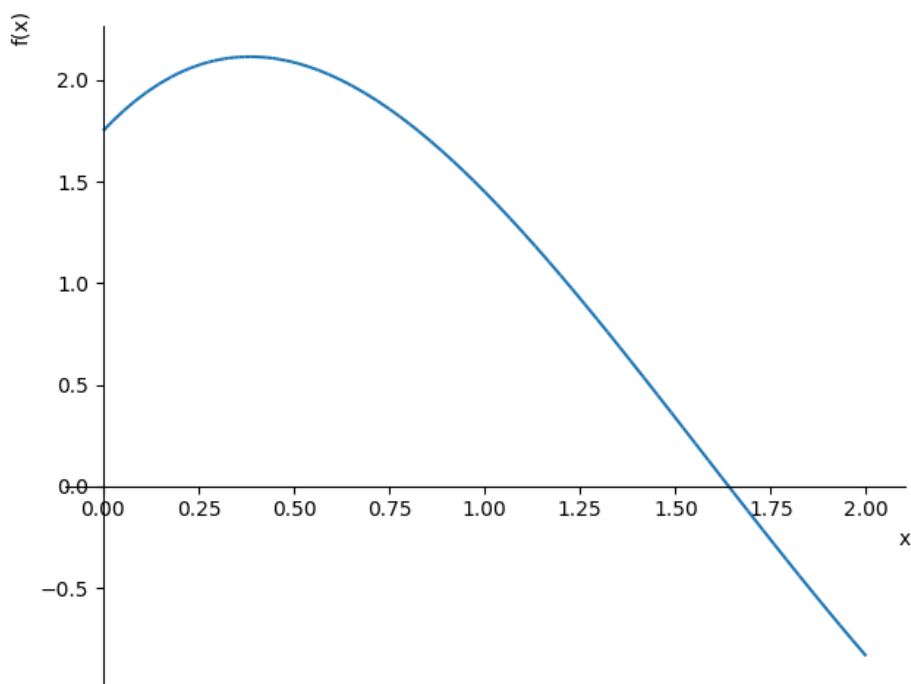
```

```
        if abs(tol) <= eps:
            break

    if iter < max_iter:
        print("方程在满足精度" + str(eps) + "的要求下, 近似解为: " + str(x_next) +
              ", 精度: " + str(tol))
    else:
        print("牛顿迭代法求解数值逼近, 已经达到最大迭代次数, 可能不收敛或精度过高.....")

if __name__ == '__main__':
    x0 = float(input("1. 请输入初值 x0: "))
    eps = float(input("2. 请输入近似解的精度 eps: "))
    max_iter = int(input("3. 请输入最大迭代次数 max_iter: "))
    numerical_newton_iteration(x0=x0, eps=eps, max_iter=max_iter) # 关键字参数
    print("=" * 60)
    symbolic_newton_iteration(x0, eps, max_iter) # 位置参数, 不可调换参数位置
```

执行结果示例:



1. 请输入初值 x0: 1
2. 请输入近似解的精度 eps: 1e-15
3. 请输入最大迭代次数 max_iter: 100

Iter	ApproximateRoot	ErrorPrecision
1	1.761198166196976	7.611981661969760e-01
2	1.638207816540044	1.229903496569322e-01
3	1.639474666447672	1.266849907628709e-03
4	1.639474729143502	6.269582919493644e-08
5	1.639474729143502	0.000000000000000e+00

方程在满足精度 $1e-15$ 的要求下，近似解为：1.6394747291435017，精度：0.0

Iter	ApproximateRoot	ErrorPrecision
1	1.761198166196976	7.611981661969760e-01
2	1.638207816540044	1.229903496569322e-01
3	1.639474666447672	1.266849907628709e-03
4	1.639474729143502	6.269582919493644e-08
5	1.639474729143502	0.000000000000000e+00

方程在满足精度 $1e-15$ 的要求下，近似解为：1.63947472914350，精度：0

本章小结：

本章主要介绍 Python 的起源、历史和应用领域，为什么学习 Python？因为 Python 语言在科学计算、数据分析、人工智能等领域的卓越表现，尤其是人工智能领域，且 Python 语言已经成为未来市场和科研的必选语言之一，其高效优雅简洁的设计理念，要求我们“人生苦短，我用 python”。

介绍 Python 语言的编码规范问题，养成一个好的编码习惯，功在千秋。重点讲解了 Python 的安装和使用方法、Python 语言的输入和输出，尤其是输出格式和格式化输出，且以案例驱动的方式详细展现了 Python 语言的特点。输入 input 函数接收控制台的输入，且以字符串的形式返回，故而，针对不同的问题需要转换输入内容的数据类型。

作 业	课下安装 Python 软件，完成视频教学的内容。 完成实验指导书——实验一的内容，书写实验报告。
主要参考资料	
课后自我总结分析	<ol style="list-style-type: none"> 1. 软件的熟练使用 2. Python 代码书写规范、养成良好的习惯、掌握必要的快捷键 3. 简单算法的设计思路，如何把数学问题转化为计算机语言求解。 4. 强调基础和思想的重要性。 5. 把数学问题、Python 算法设计，当成一种自觉习惯，该用输入输出、控制结构就自觉书写对应的 print、if、for 等结构，该定义函数就 def XXX，不必过多思考。算法设计完毕，对算法的优化调试。
备注	