Python编程基础

第1章 Python安装

1.Python解释器的安装和使用

- python下载官网
- 学习使用python必装

2.PyCharm的安装和使用

- PyCharm是一款Python的集成开发环境，在Vindows、Mac OS和Linux操作系统中都可以使用，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。
- PyCharm下载官网
- 首次打开PyCharm时，会打开导入窗口设置，一般默认选择OK即可

第2章 Python编程基础

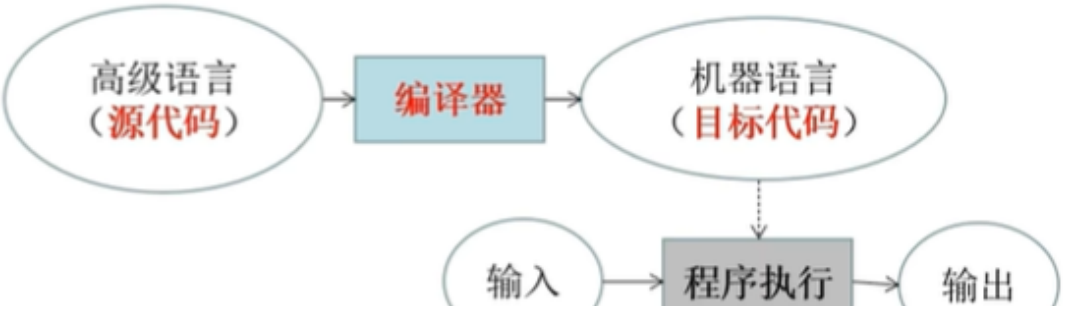
2.1 程序设计语言的分类

2.1.1 语言分类

- 机器语言
  - 是一种**二进制**语言，它直接使用二进制代码表达指令，是计算机硬件可以直接识别和执行的程序设计语言
- 汇编语言
  - 使用**方便助记符**与机器语言中的指令对应
- 高级语言
  - 是接近**自然语言**的一种计算机程序设计语言，Python、Java都是高级语言。

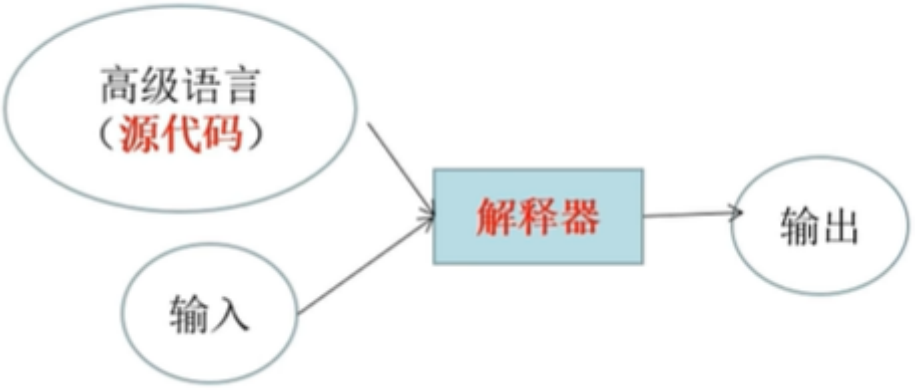
2.1.2 编译与解释

- **编译**是指将源代码转换成目标代码的过程，通常源代码是高级语言代码，目标代码是机器语言代码，执行编译的计算机程序称为编译器(Compiler)
  -



- C++,Java都是编译型的语言

- **解释**是指将源代码逐条转换成目标代码同时逐条运行目标代码的过程，执行解释的计算机程序称为解释器(Interpreter)



- Python,PHP都是解释型语言

2.2 Python语言的简介与开发工具

2.2.1 Python语言简介

- Pythoni语言的发明人一吉多。范罗苏姆（荷兰人）
- Python语言的设计非常优雅、明确、简单
- Pythoni语言具有丰富和强大的库，能够把使用其他语言制作的各种模块(尤其是C/C++)很轻松地联结在一起

2.2.2 Python语言的发展

- Python语言是在1989年诞生的，但是最早的可用版本诞生于1991年，在之后的近20年间又经历了Python2到Python3的演化过程。
- 2000年10月，Python2.0版本发布，开启了Python广泛应用的新时代。
- 2010年，Python2.x系统发布了最后一个版本，主版本号为2.7，用于终结2x系列版本的发展，并且不再进行重大改进。
- 2008年12月，Pytho3.0版本发布，这个版本的解释器内部完全采用面向对象方式实现，在语法层面做了很多重大改进。
- 2016年，所有Python重要的标准库和第三方库都已经在Python3.×版本下进行演进和发展。Python语言版本升级过程宣告结束。

2.2.3 Python语言的特点

- **语法简洁**: Python以简洁明了的语法著称，具有高度可读性，使得代码更易于编写、理解和维护。这种简洁的语法减少了代码量，提高了开发效率。
- **平台无关**: Python是一种解释性语言，可以在多个操作系统上运行，包括Windows、Linux、macOS等，因此具有很好的跨平台性，开发者可以在不同的操作系统上开发和部署Python程序。
- **粘性扩展**: Python具有良好的集成性，可以方便地与其他语言（如C、C++、Java等）进行交互，这使得开发者可以利用其他语言的库和功能来扩展Python的功能。

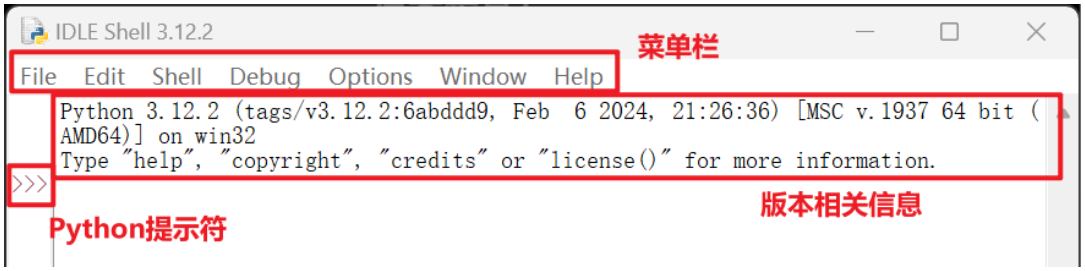
- **开源理念**: Python采用开放源代码的开发模式，这意味着任何人都可以查看、修改和共享Python的源代码。这种开源的特点促进了Python社区的发展和代码质量的提升。
- **通用灵活**: Python是一种通用编程语言，可以应用于各种领域，包括Web开发、数据科学、人工智能、自动化测试等。由于其灵活性，Python被广泛应用于不同的行业和领域。
- **强制可读**: Python强调代码的可读性，因此有一些强制性的编码规范，比如使用缩进来表示代码块，这使得代码更加清晰易懂，有助于降低代码错误的发生率。
- **支持中文**: Python具有良好的中文支持，包括关键字、标识符等，这使得母语为中文的开发者能够更轻松地使用Python进行编程。
- **模式多样**: Python拥有丰富的编程模式，包括面向对象编程、函数式编程、面向过程编程等，开发者可以根据项目需求选择合适的编程模式。
- **类库丰富**: Python拥有庞大且丰富的标准库和第三方库，涵盖了各种功能和领域，例如网络编程、图形用户界面开发、数据处理、机器学习等，这些库极大地丰富了Python的功能和应用范围，使得开发者可以快速构建复杂的应用程序。

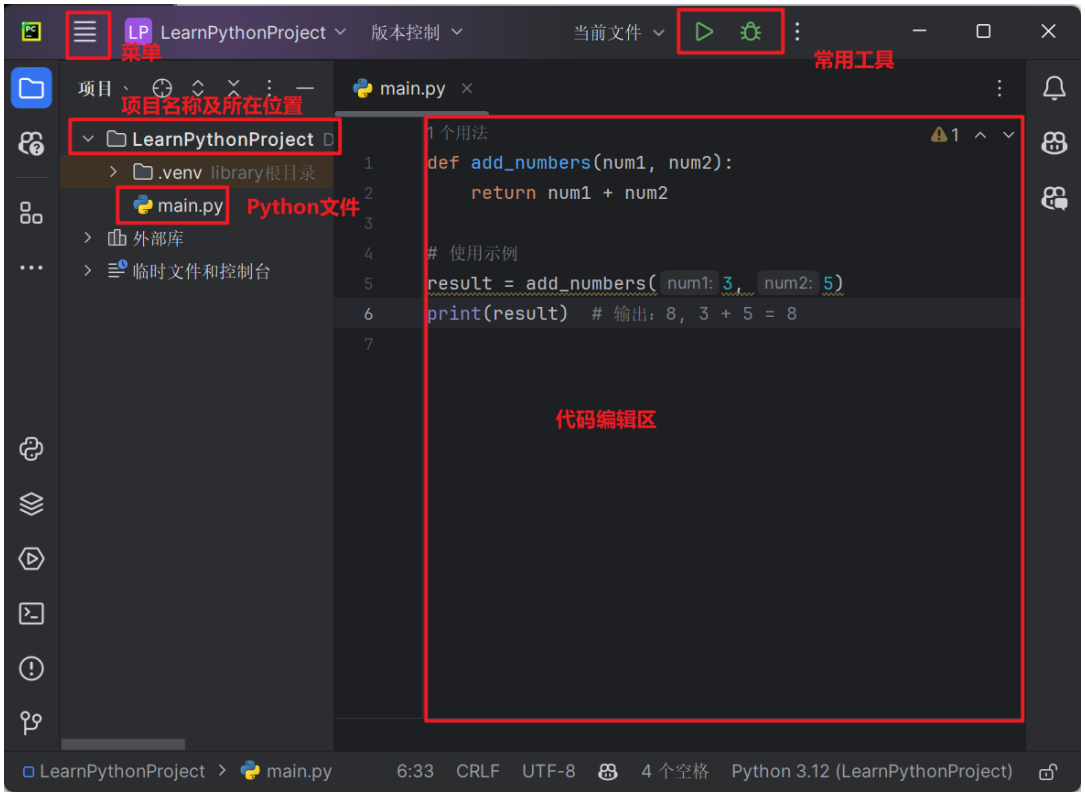
2.2.4 Python的应用领域

- **Web开发**:
    - 在Web开发中，Python的框架如Django和Flask提供了强大的工具和库，使开发人员能够快速构建高效的Web应用程序。这些框架提供了丰富的功能，包括URL路由、模板引擎、数据库集成等，使得开发Web应用变得更加简单和高效。
- **数据分析与科学计算**:
    - Python在数据科学和数值计算领域拥有强大的生态系统，例如NumPy、Pandas和SciPy等库提供了丰富的数据结构和算法，以及统计分析和数据可视化工具，使得数据分析工作更加便捷和高效。
- **人工智能和机器学习**:
    - Python成为了人工智能和机器学习领域的主流语言之一，主要得益于其丰富的机器学习库，如TensorFlow、PyTorch和Scikit-learn等。这些库提供了强大的工具和算法，使开发人员能够构建各种复杂的机器学习模型和深度学习网络。
- **自动化测试和运维**:
    - Python在自动化测试和运维领域也有着广泛的应用，因为其简洁的语法和丰富的库使得编写测试脚本和自动化任务变得更加容易和高效。诸如Selenium和Pytest等库可以用于自动化测试，而像Fabric和Ansible这样的工具则用于自动化部署和运维任务。
- **网络爬虫**:
    - Python被广泛应用于网络爬虫开发，因为其简单易用的语法和丰富的网络库（如Requests和Scrapy）能够帮助开发人员快速编写和部署网络爬虫，从而抓取和分析互联网上的数据。
- **游戏开发**:
    - 虽然Python在游戏开发领域并不是首选语言，但它仍然被用于开发一些游戏的逻辑部分、工具和脚本。一些游戏引擎（如Unity）也支持Python作为脚本语言，使得开发人员可以利用Python的简洁和灵活性来扩展游戏的功能和内容。

Python的灵活性和丰富的生态系统使得它成为了一个多用途的编程语言，可以应用于各种不同的领域，并且在各个领域中都有着显著的影响力。

2.2.5 Python的开发工具

- Python自带的集成开发学习环境IDLE(Integrated Development Learning Environment)
  - 

- 第三方开发工具PyCharm
  - 

2.3 IOP编程方式

- **IPO (Input,Process,Output)**
- 

2.4 基本的输出函数print

2.4.1 基本语法

语法结构：

```
print（输出内容）
```

print（）函数完整的语法格式：

```
#  sep参数用来设置输出数据之间的分隔符,默认是空格
#  end参数用来设置输出数据之后的结束符,默认是换行符
print(value,...,sep='',end='\n',file=None)
```

2.4.2 简单输出示例

```
a = 100  # 变量a赋值为100
b = 200  # 变量b赋值为200
print(99)  # 输出99
print(a)  # 输出变量a的值
print(a + b)  # 输出变量a和b的运算结果
```

2.4.3 输出字符串示例,用不同的符号输出字符串

```
# 输出字符串示例,用不同的符号输出字符串
print("Hello World!")  # 输出字符串,用双引号
print('Hello World!')  # 输出字符串,用单引号
print('''Hello World!''')  # 输出字符串,用三个单引号
print("""Hello World!""")  # 输出字符串,用三个双引号
```

2.4.4 不换行一次输出多个数据

```
# 一次输出多个数据,用逗号隔开,输出结果每个数据之间用空格隔开
print(a, b, 99, "Hello World!")
```

2.4.5 print函数与ASCII码

2.4.5.1 chr()函数

```
i = 100  # 变量i赋值为100
#chr()函数原型
chr(i)  #  返回一个整数i对应的Unicode字符
```

2.4.5.2 ord()函数

```
c = chr(65)  # 变量c赋值为字符A
#ord()函数原型
ord(c)  # 返回一个字符c对应的Unicode编码
```

2.4.5.3 通过ASCII码输出字符

```
# 通过ASCII码输出字符,用chr()函数
print(chr(65))  # 输出字符A
print(chr(97))  # 输出字符a
print(chr(91))  # 输出字符[
#  中文编码范围是0x4E00-0x9FA5
print(chr(0x4E00))  # 输出中文字符一
print(chr(0x9FA5))  # 输出中文字符龥
```

2.4.5.4 通过字符输出ASCII码

```
#  通过字符输出ASCII码,用ord()函数
print(ord('A'))  # 输出字符A的ASCII码
print(ord('杨'))  # 输出字符杨的ASCII码
```

2.4.6 输出到文件中

```
#  将数据输出到文件,用file参数,用内置函数open()打开文件
fp = open("test.txt", "w")  # 打开文件
print("Hello World!", file=fp)  # 输出到文件
fp.close()  # 关闭文件
```

2.4.7 使用print()函数输出复杂的数据

```
#  使用print()函数输出复杂的数据,用sep参数和end参数
#  函数原型:print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
print("Hello", "World!","hhh", sep="***", end="\n")  # 输出
Hello***World!***hhh

#  多行print函数,输出到同一行
print("Hello", end="-->")  # 输出Hello-->
print("World!")  # 输出World!
```

2.5 基本的输入函数input

2.5.1 基本语法

语法结构：

```
x=input('提示文字')
```

注意事项：无论输入的数据是什么，x的数据类型都是**字符串**类型

2.5.2 input函数的使用

```
#  input函数的使用
name = input("请输入你的名字：")
print("你好，" + name)

#  输入数字并转换为整数
age = input("请输入你的年龄：")
print("你的年龄是：" + age)  # 输出年龄
age = int(age)  # 将输入的字符串转换为整数
print("你的年龄是：" + str(age))  # 输出年龄
```

2.6 Python中的注释

- 注释：
  - 程序员在代码中对代码功能`解释说明`的`标注性文字`
  - 可以提高代码的`可读性`
  - 注释的内容将`被Python解释器忽略`，不被计算机执行
  - 注释分类：`单行注释`、`多行注释`和`中文文档声明注释`

2.6.1 单行注释

```
#  单行注释练习
#  单行注释有两种方式,一种是在代码后面加#,另一种是单独一行写注释
#  举例:
#  要求从键盘输入年份,要求是4位的年份,举例:2019
year1 = input("请输入年份:")

year2 = input("请输入年份:")  #  输入年份
```

2.6.2 多行注释

```
year2 = input("请输入年份:")  #  输入年份

#  多行注释练习
#  多行注本质上是字符串,可以用三个单引号或者三个双引号包裹
#  举例:
'''
要求从键盘输入年份,要求是4位的年份,举例:2019
'''
year1 = input("请输入年份:")

"""
要求从键盘输入年份,要求是4位的年份,举例:2019
"""

year2 = input("请输入年份:")  #  输入年份
```

2.6.3 中文文档声明注释

```
#  中文文档声明注释练习
#  中文文档声明注释是在必须文件的开头声明文件的编码格式
#  举例:
#  -*- coding: utf-8 -*-
```

2.7 Python中的缩进

- 代码缩进
  - 是指`每行语句开始前的空白区域`
  - 用来表示Python程序间的`包含`和`层次关系`

- 类定义、函数定义、流程控制语句以及异常处理语句等`行尾的冒号`和`下一行的缩进`表示`一个代码块的开始`，而缩进结束，则`表示一个代码块的结束`
- 通常情况下采用`4个空格`作为一个缩进量

```
#  代码缩进
'''
代码缩进是Python语言的一个重要特点，它用来表示代码之间的层次关系。
在Python中，代码块的开始和结束是通过代码缩进来确定的。
下面是一个简单的例子：
'''
if True:
    print('True')
else:
    print('False')
```

2.8 章节习题

2.8.1 选择题

1)Python是一种(B)类型的编程语言
A.机器语言
B.解释
C.编译
D.汇编语言

2)Python语句print(“中国，你好")的输出是（C）
A.(“中国，你好")
B.“中国，你好”
C.中国，你好
D.运行结果出错

3)以下不是IPO模式的是（B）
A.input
B.program
C.process
D.output

4)Python语言通过（C）来体现语句之间的逻辑关系
A.{}
B.()
C.缩进
D,自动识别逻辑

5)Python解释器在语法上不支持（D）编程方式
A,面向过程
B,面向对象
C.语句
D,自然语言

2.8.2 实战题

实战一：输出“人生苦短，我用Python”需求：使用print()函数将“人生苦短，我用Python”输出到文本文件text.txt中

```
fp = open("text.txt",'w')
print('人生苦短，我用Python',file=fp)
fp.close()
```

实战二：输出个人自我介绍
需求：使用input(函数从键盘输入姓名、年龄，座右铭，并使用print（）函数输出到控制台

```
name = input("请输入你的名字：")
age = input("请输入你的年龄：")
motto = input("请输入你的座右铭：")
print("我叫" + name + "，今年" + age + "岁，我的座右铭是" + motto)
```

第3章