目录

Python入门教程

教程简介

本教程专为零基础的学习者精心设计，旨在帮助学习者从零开始系统地掌握Python编程。通过17个循序渐进的实验，我们将带您深入探索Python的基础语法、数据结构、控制流、函数、库使用、文件操作、异常处理和面向对象编程等核心知识。

实验环境准备

软件准备

1. 安装RflySim工具链 v3.07及以上版本
2. 推荐安装Visual Studio Code，详细配置可见：【RflySim安装目录】\1.RflySimIntro\2.AdvExps_PythonConfig

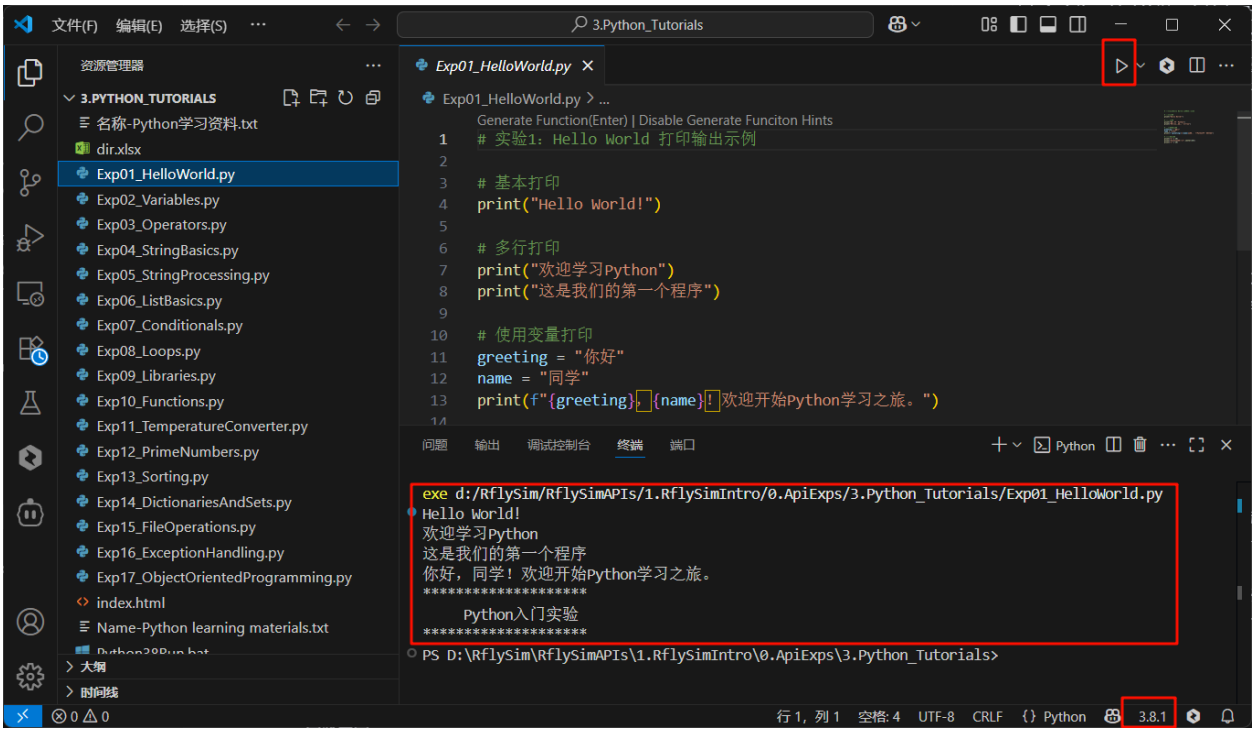
运行方式

本教程提供两种代码运行方式：

方式一：VSCode直接运行

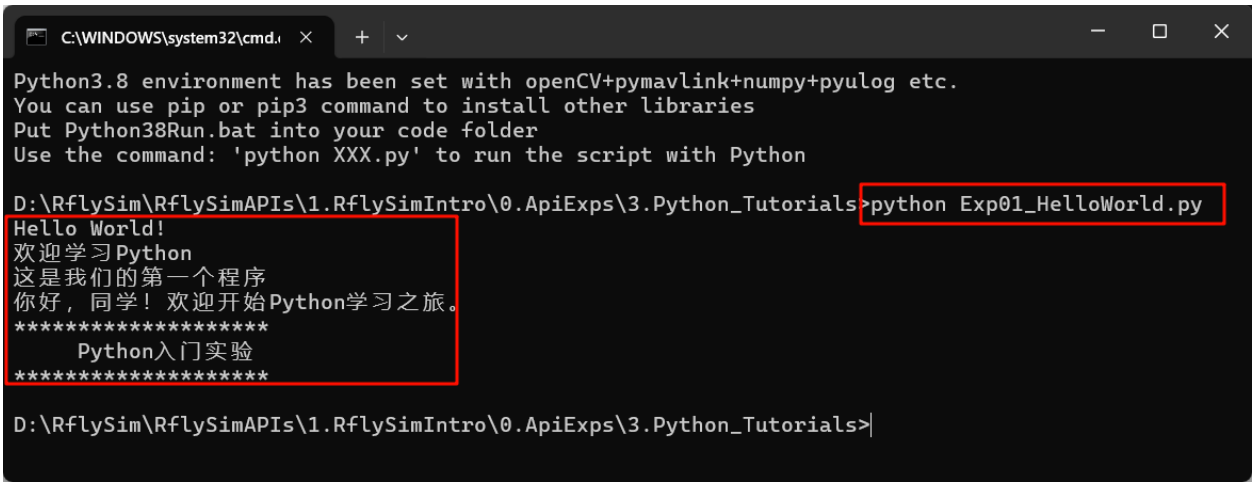
1. 打开VSCode
2. 打开本项目文件夹

3. 右键选择要运行的Python文件
4. 点击”Run Python File in Terminal”



方式二：批处理文件运行

1. 双击`Python38Run.bat`
2. 在弹出的命令行窗口中
3. 输入 `python Exp**_***.py`（将**替换为实验编号）



实验目录

实验1：Hello World打印输出

实验目标

1. 了解Python程序的基本结构
2. 学会使用print()函数
3. 掌握基本的代码编写和运行方法

实验内容

```python
# 基本打印
print("Hello World!")
# 多行打印
print("欢迎学习Python")
print("这是我们的第一个程序")
# 使用变量打印
greeting = "你好"
name = "同学"
print(f"{greeting}, {name}! 欢迎开始Python学习之旅。")
```

代码分析

代码行	功能	输出结果
`print("Hello World!")`	最基本的打印输出	`Hello World!`
`print("欢迎学习Python")`	打印一行文本	`欢迎学习Python`
`print(f"{greeting}, {name}! 欢迎开始Python学习之旅。")`	使用f-string格式化输出	`你好, 同学! 欢迎开始Python学习之旅。`

知识点

知识点	详细说明	示例
print()函数	用于向控制台输出内容	`print("Hello")`
字符串输出	可以直接打印字符串	`print("Python")`
格式化字符串	使用f-string动态插入变量	`print(f"我今年{age}岁")`
变量使用	存储和使用数据	`name = "小明"`
代码编写规范	使用缩进和有意义的变量名	-

实验2：认识变量

实验目标

1. 理解变量的概念
2. 学习不同类型的变量
3. 掌握变量的基本操作

实验内容

```python
# 整数变量
age = 15
print(f"年龄: {age}")

# 浮点数变量
height = 1.75
print(f"身高: {height}米")

# 字符串变量
name = "小明"
print(f"姓名: {name}")

# 类型转换
str_age = "20"
```

```
int_age = int(str_age)
print(f"字符串转整数: {int_age + 5}")
```

代码分析

代码行	功能	输出结果	变量类型
`age = 15`	定义整数变量	`年龄: 15`	int
`height = 1.75`	定义浮点数变量	`身高: 1.75米`	float
`name = "小明"`	定义字符串变量	`姓名: 小明`	str
`int_age = int(str_age)`	字符串转整数	`字符串转整数: 25`	int

知识点

知识点	详细说明	示例
变量命名规则	字母、数字、下划线，区分大小写	`user_name = "张三"`
基本数据类型	整数、浮点数、字符串、布尔值	`x = 10; y = 3.14; name = "Tom"; is_student = True`
类型转换方法	使用int(), float(), str(), bool()	`int("20")`, `float(10)`, `str(100)`
变量赋值	使用=号赋值	`x = 5`
类型检查	使用type()函数	`type(x)`

实验3：运算符与表达式

实验目标

1. 学习算术运算符
2. 理解比较运算符
3. 掌握逻辑运算符
4. 了解运算符优先级

实验内容

```
# 算术运算符
a, b = 10, 3
print(f"加法: {a + b}")   # 13
print(f"减法: {a - b}")   # 7
print(f"乘法: {a * b}")   # 30
print(f"除法: {a / b}")   # 3.333
print(f"整除: {a // b}")  # 3
print(f"取余: {a % b}")   # 1
print(f"幂运算: {a ** b}")  # 1000

# 比较运算符
x, y = 5, 7
print(f"等于: {x == y}")     # False
```

```
print(f"不等于: {x != y}")   # True
print(f"大于: {x > y}")      # False
print(f"小于: {x < y}")      # True
```

代码分析

代码行	功能	输出结果	运算符类型
`a + b`	加法运算	`13`	算术运算符
`a - b`	减法运算	`7`	算术运算符
`a * b`	乘法运算	`30`	算术运算符
`a / b`	除法运算	`3.333`	算术运算符
`a // b`	整除运算	`3`	算术运算符
`a % b`	取余运算	`1`	算术运算符
`a ** b`	幂运算	`1000`	算术运算符
`x == y`	相等比较	`False`	比较运算符
`x != y`	不等比较	`True`	比较运算符

知识点

知识点	详细说明	示例
算术运算符	基本数学运算	`+, -, *, /, //, %, **`
比较运算符	用于比较两个值	`==, !=, >, <, >=, <=`
逻辑运算符	用于组合条件	`and, or, not`
运算符优先级	决定运算顺序	`(a + b) * c`
短路运算	提前终止逻辑运算	`False and print("不会执行")`

实验4：初始字符串

实验目标

1. 了解字符串的基本概念
2. 学习字符串的创建方法
3. 掌握基本的字符串操作

实验内容

```
# 字符串的定义
name1 = '小明'  # 单引号
name2 = "小红"  # 双引号
description = '''这是一个
多行字符串的例子'''

# 字符串索引
text = "Python学习"
print(f"第一个字符: {text[0]}")      # P
print(f"最后一个字符: {text[-1]}")   # 习
```

```python
# 字符串切片
print(f"前两个字符：{text[0:2]}")    # Py
print(f"后两个字符：{text[-2:]}")    # 学习
# 字符串方法
greeting = "  Hello, World!  "
print(f"去除空格：'{greeting.strip()}'")   # "Hello, World!"
print(f"大写：{greeting.upper()}")         # "  HELLO, WORLD!  "
print(f"小写：{greeting.lower()}")         # "  hello, world!  "
```

代码分析

代码行	功能	输出结果	字符串操作
`text[0]`	获取第一个字符	`P`	索引
`text[-1]`	获取最后一个字符	`习`	索引
`text[0:2]`	获取前两个字符	`Py`	切片
`text[-2:]`	获取后两个字符	`学习`	切片
`greeting.strip()`	去除两端空白	`"Hello, World!"`	方法
`greeting.upper()`	转换为大写	`"  HELLO, WORLD!  "`	方法
`greeting.lower()`	转换为小写	`"  hello, world!  "`	方法

知识点

知识点	详细说明	示例
字符串定义	单引号、双引号、三引号	`'hello'`, `"world"`, `'''多行'''`
字符串索引	从0开始的正向和反向索引	`text[0]`, `text[-1]`
字符串切片	提取子串的方法	`text[1:4]`, `text[:3]`
字符串方法	内置的字符串处理函数	`strip()`, `upper()`, `lower()`
字符串不可变性	字符串创建后不能修改	-
字符串格式化	动态插入变量	`f"我是{name}"`

实验5：字符串处理

实验目标

1. 深入学习字符串处理方法
2. 掌握字符串的高级操作
3. 理解字符串处理的实际应用

实验内容

```python
# 字符串分割
text = "Python,Java,C++,JavaScript"
languages = text.split(",")
print(f"分割后的语言列表：{languages}")

# 字符串替换
sentence = "I love Python programming"
```

```
new_sentence = sentence.replace("Python", "Java")
print(f"替换后的句子: {new_sentence}")
# 字符串查找
search_text = "Hello, Python is awesome!"
index = search_text.find("Python")
print(f"'Python'的位置: {index}")
```

代码分析

代码行	功能	输出结果	字符串操作
`text.split(",")`	按逗号分割字符串	`['Python', 'Java', 'C++', 'JavaScript']`	分割
`sentence.replace("Python", "Java")`	替换字符串中的内容	`"I love Java programming"`	替换
`search_text.find("Python")`	查找子串位置	`7`	查找

知识点

知识点	详细说明	示例
split()方法	将字符串分割为列表	`"a,b,c".split(",")`
replace()方法	替换字符串中的子串	`"hello".replace("l", "x")`
find()方法	查找子串的起始位置	`"python".find("th")`

实验6：初始列表

实验目标

1. 了解列表的基本概念
2. 学习列表的创建和基本操作
3. 掌握列表的常用方法

实验内容

```
# 创建列表
fruits = ["apple", "banana", "cherry"]
numbers = [1, 2, 3, 4, 5]
mixed_list = [1, "hello", True, 3.14]

# 访问列表元素
print(f"第一个水果: {fruits[0]}")
print(f"最后一个数字: {numbers[-1]}")

# 列表切片
print(f"前两个水果: {fruits[:2]}")
print(f"后两个数字: {numbers[-2:]}")

# 列表修改
fruits.append("orange")
fruits.insert(1, "grape")
```

```
print(f"添加元素后的水果列表: {fruits}")
```

代码分析

代码行	功能	输出结果	列表操作
`fruits[0]`	访问第一个元素	`"apple"`	索引
`fruits.append("orange")`	在列表末尾添加元素	`["apple", "banana", "cherry", "orange"]`	添加
`fruits.insert(1, "grape")`	在指定位置插入元素	`["apple", "grape", "banana", "cherry"]`	插入

知识点

知识点	详细说明	示例
列表创建	使用方括号创建列表	`my_list = [1, 2, 3]`
列表索引	从0开始的正向和反向索引	`my_list[0]`, `my_list[-1]`
列表切片	提取部分列表	`my_list[1:3]`, `my_list[:2]`

实验7：条件判断与选择

实验目标

1. 理解条件语句的基本概念
2. 掌握if、elif、else的使用
3. 学习复杂条件的编写

实验内容

```
# 简单的if语句
age = 18
if age >= 18:
    print("你已经成年")

# if-else语句
score = 75
if score >= 60:
    print("及格")
else:
    print("未及格")

# if-elif-else语句
grade = 85
if grade >= 90:
    print("优秀")
elif grade >= 80:
    print("良好")
```

```
elif grade >= 60:
    print("及格")
else:
    print("不及格")
```

代码分析

代码行	功能	输出结果	条件类型
`if age >= 18:`	简单条件判断	你已经成年	基本判断
`if score >= 60:`	if-else判断	及格	二分支
`if grade >= 90:`	if-elif-else多分支	良好	多分支

知识点

知识点	详细说明	示例
if语句	基本条件判断	`if condition:`
else语句	条件不满足时执行	`else:`
elif语句	多条件判断	`elif condition:`

实验8：循环（for、while）

实验目标

1. 理解循环的基本概念
2. 掌握for循环和while循环
3. 学习循环控制语句

实验内容

```
# for循环遍历列表
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(f"我喜欢吃{fruit}")

# for循环使用range()
for i in range(5):
    print(f"当前数字: {i}")

# while循环
count = 0
while count < 5:
    print(f"计数: {count}")
    count += 1

# break语句
for num in range(10):
    if num == 5:
        print("找到5，停止循环")
```

```
        break
    print(num)
```

代码分析

代码行	功能	输出示例	循环类型
`for fruit in fruits:`	遍历列表	`我喜欢吃apple`	for循环
`for i in range(5):`	使用range()生成数字	`当前数字: 0`	for循环
`while count < 5:`	条件循环	`计数: 0`	while循环
`if num == 5: break`	提前终止循环	`找到5，停止循环`	循环控制

知识点

知识点	详细说明	示例
for循环	遍历序列（列表、字符串等）	`for item in sequence:`
while循环	满足条件时重复执行	`while condition:`
range()函数	生成数字序列	`range(start, stop, step)`

实验9：函数

实验目标

1. 理解函数的基本概念
2. 学习函数的定义和调用
3. 掌握函数参数和返回值的使用

实验内容

```
# 最简单的函数
def greet():
    print("你好，欢迎学习Python!")

# 带参数的函数
def welcome(name):
    print(f"欢迎，{name}!")

# 带返回值的函数
def add(a, b):
    return a + b

# 默认参数
def power(base, exponent=2):
    return base ** exponent

# 可变参数
def sum_all(*numbers):
    total = 0
    for num in numbers:
```

```
        total += num
    return total
```

代码分析

代码行	功能	输出结果	函数类型
`def greet():`	无参数无返回值函数	`你好，欢迎学习Python！`	基本函数
`def welcome(name):`	带参数函数	`欢迎，小明！`	带参数
`def add(a, b):`	带返回值函数	`8`	返回值
`def power(base, exponent=2):`	默认参数函数	`4`	默认参数

知识点

知识点	详细说明	示例
函数定义	使用def关键字	`def function_name():`
函数参数	传递数据给函数	`def func(a, b)`
返回值	函数返回计算结果	`return result`
默认参数	参数有默认值	`def func(a=10)`

实验9：Python常用库

实验目标

1. 了解Python常用标准库
2. 学习时间、数学、随机数等库的使用
3. 掌握库的导入和基本函数调用

实验内容

```
# 导入常用库
import time
import math
import random
import datetime

# 时间操作示例
current_timestamp = time.time()
formatted_time = time.strftime("%Y-%m-%d %H:%M:%S")

# 数学运算示例
sqrt_value = math.sqrt(16)
random_number = random.randint(0, 100)

# 日期计算示例
now = datetime.datetime.now()
```

```
future_date = now + datetime.timedelta(days=30)
```

代码分析

代码行	功能	示例输出	库操作
`time.time()`	获取当前时间戳	`1623456789.123`	时间戳
`math.sqrt(16)`	计算平方根	`4.0`	数学运算
`random.randint(0, 100)`	生成随机整数	`42`	随机数
`datetime.datetime.now()`	获取当前日期时间	`2023-06-15 10:30:45`	日期时间

知识点

知识点	详细说明	示例
库导入	import语句的使用	`import time`
时间操作	获取时间戳、格式化	`time.time()`, `time.strftime()`
数学函数	数学计算方法	`math.sqrt()`, `math.ceil()`
随机数生成	随机数和随机选择	`random.randint()`, `random.choice()`
日期计算	日期时间处理	`datetime.timedelta()`

实验10：综合案例1—摄氏温度转华氏温度

实验目标

1. 综合运用前面学习的编程知识
2. 实现温度转换的实际应用
3. 学习用户交互和输入处理

实验内容

```
# 温度转换函数
def celsius_to_fahrenheit(celsius):
    """将摄氏温度转换为华氏温度"""
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit

# 温度转换程序
def temperature_converter():
    """温度转换交互程序"""
    print("欢迎使用温度转换器!")


    while True:
        try:
```

```
            # 获取用户输入
            celsius = float(input("请输入摄氏温度（输入'q'退出）："))

            # 转换并显示结果
            fahrenheit = celsius_to_fahrenheit(celsius)
            print(f"{celsius}°C = {fahrenheit:.2f}°F")

        except ValueError:
            # 处理非数字输入
            user_input = input("输入无效。是否退出？(y/n): ")
            if user_input.lower() == 'y':
                break
```

代码分析

代码行	功能	示例输出	知识点
`celsius_to_fahrenheit()`	温度转换计算	`0°C = 32.00°F`	函数定义
`float(input())`	获取用户输入	`25.5`	类型转换
`try-except`	处理输入异常	`输入无效`	异常处理

知识点

知识点	详细说明	示例
函数定义	创建温度转换函数	`def celsius_to_fahrenheit(celsius):`
数学计算	温度转换公式	`(celsius * 9/5) + 32`
用户输入	使用input()获取输入	`float(input())`

实验11：综合案例2—100以内质数求解

实验目标

1. 学习质数判断算法
2. 综合运用循环和条件判断
3. 掌握列表操作和函数设计

实验内容

```
# 质数判断函数
def is_prime(n):
    """判断一个数是否为质数"""
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

# 查找100以内的质数
def find_primes(max_number):
    """找出指定范围内的所有质数"""
    primes = []
```

```
    for num in range(2, max_number + 1):
        if is_prime(num):
            primes.append(num)
    return primes
```

代码分析

代码行	功能	示例输出	知识点
`is_prime(n)`	判断质数	`True/False`	数学算法
`find_primes(max_number)`	查找指定范围质数	`[2, 3, 5, 7, ...]`	列表生成
`range(2, int(n**0.5) + 1)`	优化质数判断	-	算法优化

知识点

知识点	详细说明	示例
质数定义	只能被1和自身整除的数	`2, 3, 5, 7`
循环判断	遍历并检查数字	`for num in range(2, max_number)`
数学算法	质数判断优化	`int(n**0.5)`

实验12：综合案例3—排序

实验目标

1. 学习常见排序算法
2. 理解排序的基本原理
3. 掌握Python中的排序方法

实验内容

```
# 冒泡排序算法
def bubble_sort(arr):
    """冒泡排序：每次比较相邻元素，将最大元素"冒泡"到末尾"""
    n = len(arr)
    for i in range(n):
        # 标记是否发生交换，优化算法
        swapped = False

        for j in range(0, n - i - 1):
            # 如果前一个元素大于后一个元素，交换
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True

        # 如果没有发生交换，说明已经排序完成
        if not swapped:
```

```
            break

    return arr
# 选择排序算法
def selection_sort(arr):
    """选择排序：每次选择最小元素放到前面"""
    n = len(arr)
    for i in range(n):
        # 假设当前位置是最小值
        min_idx = i

        # 在未排序区间找最小值
        for j in range(i + 1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j

        # 交换最小值到正确位置
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

    return arr
```

代码分析

代码行	功能	示例输出	排序算法
`bubble_sort()`	冒泡排序	`[11, 12, 22, 25, 34, 64, 90]`	交换排序
`selection_sort()`	选择排序	`[11, 12, 22, 25, 34, 64, 90]`	选择排序
`arr[j], arr[j + 1] = arr[j + 1], arr[j]`	元素交换	-	交换技巧

知识点

知识点	详细说明	示例
冒泡排序	重复比较相邻元素	每轮将最大元素移到末尾
选择排序	每次选择最小元素	将最小元素放到前面
列表操作	元素交换、复制	`arr[i], arr[j] = arr[j], arr[i]`

实验14：字典和集合

实验目标

1. 了解字典的基本概念和操作
2. 学习集合的特性和常用方法
3. 掌握数据去重和集合运算

实验内容

```
# 字典操作
student = {
    "name": "小明",
    "age": 18,
    "subjects": ["数学", "英语", "编程"]
}
```

```
# 集合操作
fruits1 = {"apple", "banana", "cherry"}
fruits2 = {"banana", "date", "elderberry"}
# 集合运算
common_fruits = fruits1.intersection(fruits2)
all_fruits = fruits1.union(fruits2)
```

代码分析

代码行	功能	示例输出	操作类型
`student["name"]`	访问字典元素	`"小明"`	字典访问
`fruits1.intersection(fruits2)`	集合交集	`{"banana"}`	集合运算
`set(numbers)`	数据去重	`{1, 2, 3, 4, 5}`	去重

知识点

知识点	详细说明	示例
字典创建	使用大括号或dict()创建	`{"key": value}`
字典方法	keys(), values(), items()	`student.keys()`
集合特性	无序、唯一	`{1, 2, 3}`
集合运算	交集、并集、差集	`set1.union(set2)`

实验15：文件操作

实验目标

1. 学习文件的读写操作
2. 掌握文本和CSV文件处理
3. 了解文件操作的异常处理

实验内容

```
# 写入文本文件
with open("sample.txt", "w", encoding="utf-8") as file:
    file.write("Python文件操作示例")

# 读取文本文件
with open("sample.txt", "r", encoding="utf-8") as file:
    content = file.read()

# CSV文件操作
import csv
students = [
    ["姓名", "年龄", "班级"],
    ["小明", 18, "高三1班"]
]
with open("students.csv", "w", newline="") as file:
    csv.writer(file).writerows(students)
```

代码分析

代码行	功能	示例输出	文件操作
`open("file.txt", "w")`	打开文件写入	-	文件写入
`file.read()`	读取全部内容	`"文本内容"`	文件读取
`csv.writer(file)`	CSV文件写入	-	CSV操作

知识点

知识点	详细说明	示例
文件模式	读写模式（r, w, a）	`open(file, mode)`
with语句	自动关闭文件	`with open(file) as f:`
CSV处理	读写表格数据	`csv.reader()`, `csv.writer()`
文件编码	指定文件编码	`encoding="utf-8"`

实验16：异常处理

实验目标

1. 理解异常的基本概念
2. 学习try-except语句
3. 掌握常见异常处理方法

实验内容

```
try:
    x = int(input("请输入数字: "))
    result = 10 / x
except ValueError:
    print("请输入有效数字")
except ZeroDivisionError:
    print("不能除以零")

# 自定义异常
class AgeError(Exception):
    def __init__(self, age):
        self.message = f"年龄{age}不valid"
        super().__init__(self.message)

def check_age(age):
    if age < 0 or age > 120:
        raise AgeError(age)
```

代码分析

代码行	功能	示例输出	异常类型
`try-except`	捕获异常	`"请输入有效数字"`	异常处理
`raise AgeError`	抛出自定义异常	`"年龄不valid"`	自定义异常
`finally`	无论是否异常都执行	-	资源清理

知识点

知识点	详细说明	示例
异常类型	常见异常类型	`ValueError`, `ZeroDivisionError`
异常捕获	多异常处理	`except (Error1, Error2):`
自定义异常	继承Exception类	`class MyError(Exception):`

实验17：面向对象编程基础

实验目标

1. 理解类和对象的概念
2. 学习类的定义和使用
3. 掌握继承和方法重写

实验内容

```
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age
        self.scores = {}

    def add_score(self, subject, score):
        self.scores[subject] = score

    def get_average_score(self):
        return sum(self.scores.values()) / len(self.scores)

# 继承
class SportStudent(Student):
    def __init__(self, name, age, sport):
        super().__init__(name, age)
        self.sport = sport
```

代码分析

代码行	功能	示例输出	对象操作
`__init__()`	构造方法	-	对象初始化
`self.name`	实例变量	`"小明"`	对象属性

`super().__init__()` 调用父类构造方法 - 继承

知识点

知识点	详细说明	示例
类定义	class关键字	`class ClassName:`
构造方法	对象初始化	`def __init__(self):`
实例方法	对象行为	`def method(self):`
继承	类的继承	`class Child(Parent):`

学习建议

1. 认真阅读每个实验的说明
2. 逐行运行代码，理解每行代码的作用
3. 尝试修改代码，观察不同的运行结果
4. 完成每个实验后的拓展练习