

# 《Python 程序设计》指导书

## 一 Python 语法规则

### 1. 实验要求：

- (1) 可以编辑并运行一个 Python 程序
- (2) 定义一个变量并对其进行赋值、计算等操作
- (3) 在程序中使用输入函数获取初始值，使用输出函数将计算结果输出。

### 2. 实验目的

- (1) 熟悉 Python 程序的运行方式
- (2) 掌握变量的定义和使用方法
- (3) 掌握基本的输入、输出方法


### 3. 实验内容

- (1) 定义一个求和函数，设置两个参数，返回两个数相加后的值，并调用函数计算两个数的和。
- (2) 编写计算一个数的平方的函数运算字符串表达式，并用 `eval()` 函数计算表达式的结果，将结果赋值给变量并输出。
- (3) 输入商品的价格和折扣，并转为浮点型，计算商品折扣后的价格。
- (4) 输入球的半径，求球的表面积和体积。
- (5) 对文章中词频进行统计。

参考代码如下：

1.

```
def add(a, b): # 定义加和函数，有两个参数
    print("成功调用 add() 函数")
    return a + b # 返回两个数相加后的值
sum = add(2, 3) # 向函数传参数 2、3，使用 add() 函数计算后
print("2+3 的和为：", sum) # 打印 sum 的值
```



```
成功调用add()函数
2+3的和为: 5
```

2.

#求 18 的平方

```
x = 18
```

```
#用 eval 函数运算字符串表达式 x**2，其中 x 已赋值为 18
```

```
square_sum = eval("x**2")
```

```
#eval 函数计算的结果赋值给了变量 square_sum，打印该变量
```

```
print("18 的平方为", square_sum, sep = ":")
```

```
18的平方为:324
```

3.

```
#输入商品的价格和折扣，并转为浮点型
```

```
price = float(input("商品原来的价格是："))
```

```
discount = float(input("此商品的折扣为(输入小数)："))
```

```
#求得商品折扣后的价格，运用算术运算符*
```

```
current_price = price * discount
```

```
print("商品现在的价格为：", current_price)
```

```
商品原来的价格是: 100  
此商品的折扣为(输入小数): 0.8  
商品现在的价格为: 80.0
```

4.

```
import math
```

```
r=float(input("请输入半径："))
```

```
sarea=4*math.pi*r*r
```

```
volume=4/3*math.pi*r**3
```

```
print ("球的表面积: %.2f"% sarea)
```

```
print ("球的体积: %.2f" % volume)
```

```
请输入半径: 3  
球的表面积: 113.10  
球的体积: 113.10
```

5.

```
#黛玉葬花节选
```

```
verse = """
```

```
怪奴底事倍伤神？半为怜春半恼春。怜春忽至恼忽去，至又无言去不闻。
```

```
昨宵庭外悲歌发，知是花魂与鸟魂？花魂鸟魂总难留，鸟自无言花自羞；
```

```
愿奴胁下生双翼，随花飞到天尽头。天尽头，何处有香丘？
```

```
未若锦囊收艳骨，一抔净土掩风流；质本洁来还洁去，强于污淖陷渠
```

沟。

尔今死去侬收葬，未卜侬身何日丧？侬今葬花人笑痴，他年葬侬知是谁？

试看春残花渐落，便是红颜老死时。一朝春尽红颜老，花落人亡两不知！

"""

```
template = "虚词：{0:~^5}出现了：{1:~^5}次"
empty_word1 = "为"
result1 = verse.count(empty_word1)
empty_word2 = "以"
result2 = verse.count(empty_word2)
empty_word3 = "何"
result3 = verse.count(empty_word3)
print(template.format(empty_word1,result1))
print(template.format(empty_word2,result2))
print(template.format(empty_word3,result3))
```

```
虚词：--为--出现了：--1--次
虚词：--以--出现了：--0--次
虚词：--何--出现了：--2--次
```

## 二

### 1. 实验要求

- (1) 实现 if 条件语句
- (2) 实现 for 和 while 循环语句
- (3) 理解并使用 break 和 continue 跳转语句

### 2. 实验目的

- (1) 掌握程序的基本结构
- (2) 掌握条件语句和循环语句的使用方法
- (3) 掌握跳转语句的使用方法

### 3. 实验内容

(1) 统计一年中有 31 天的月份，输入 1 到 12 的整数，如果它对应的月份为 31 天则输出 yes，否则输出 no。

(2) 输入身高和体重，计算身体质量指数 BMI，利用条件语句找

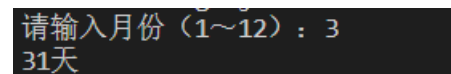
到对应的指标分类并输出类别。

(3) 寻找 100 到 999 之间的水仙花数，判断一个数是否是回文数。

参考代码如下：

1.

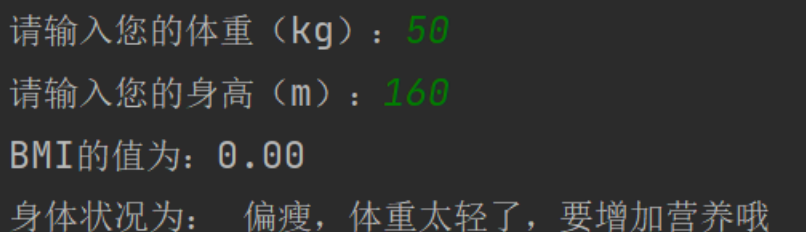
```
month = int(input('请输入月份 (1~12) : '))
if month in (4,6,9,11):
    print('30 天')
else:
    print('31 天')
```

A terminal window showing the input '3' for the month and the output '31天' (31 days).

```
请输入月份 (1~12) : 3
31天
```

2.

```
weight = float(input("请输入您的体重 (kg) : "))
height = float(input("请输入您的身高 (m) : "))
bmi = weight / pow(height,2)
print(f"BMI 的值为: {bmi:.2f}")
if bmi < 18.5:
    level = "偏瘦，体重太轻了，要增加营养哦"
if 18.5 <= bmi < 24:
    level = "正常，您的身体非常健康，太棒啦"
if 24 <= bmi < 28:
    level = "偏胖，规律作息、合理饮食，会变得健康哦"
if bmi >= 28:
    level = "肥胖，保持健康的身体是爱护自己的表现，要运动起来呀"
print("身体状况为: ",level)
```

A terminal window showing the input of weight '50' and height '160', followed by the BMI calculation '0.00' and the output '身体状况为: 偏瘦，体重太轻了，要增加营养哦'.

```
请输入您的体重 (kg) : 50
请输入您的身高 (m) : 160
BMI的值为: 0.00
身体状况为: 偏瘦，体重太轻了，要增加营养哦
```

3.

```
for number in range(100,1000):
```

```

high = number // 100
mid = number // 10 % 10
low = number % 10
if number == high**3 + mid**3 + low**3:
    print(number, end="\t\t")

```

153	370	371	407
-----	-----	-----	-----

```

number = int(input("请输入一个数字："))
origin_number = number
reversed_number = 0
while number > 0:
    reversed_number = reversed_number * 10 + number % 10
    number //= 10
if origin_number == reversed_number:
    print("{}是回文数".format(origin_number))
else:
    print("{}不是回文数".format(origin_number))

```

请输入一个数字：123 123不是回文数	请输入一个数字：12321 12321是回文数
-------------------------	----------------------------

### 三 列表与元组

#### 1. 实验要求

- (1)可以使用不同方法定义列表，并且对列表进行操作
- (2)可以取出列表元素并对元素进行计算
- (3)可以使用不同方法定义元组，并对元组进行操作

#### 2. 实验目的

- (1)掌握列表的定义以及基本操作
- (2)掌握列表元素的使用
- (3)掌握元组的定义以及基本操作

#### 3. 实验内容

- (1)用列表存储矩阵元素，并对矩阵进行加、减、乘等操作
- (2)计算所有可能的运动计划方案消耗的热量并输出，求出最大消耗量和最小消耗量并输出。
- (3)创建一个商品清单，根据输入获取用户的购物资金。打印商品清单，根据输入获取用户选择要购买的商品。退出系统后，打印购买的商品清单。

参考代码如下：

1.

```
import numpy as np
a = np.array([4, 2, 3])
b = np.array([2, 5, 7])
c = a+b
d = a-b
e = a*b
f = a/b
print("加%s" % c)
print("减%s" % d)
print("乘%s" % e)
print("除%s" % f)
```

```
加[ 6  7 10]
减[ 2 -3 -4]
乘[ 8 10 21]
除[2.         0.4        0.42857143]
```

2.

```
run_list = ["0 分钟", "20 分钟", "40 分钟", "60 分钟"]
swim_list = ["0 米", "200 米", "400 米", "600 米"]
calories_list = []
for i in range(len(run_list)):
    for j in range(len(swim_list)):
        calories_list.append(i * 200 + j * 100)
print("卡路里列表：", calories_list)
print(f"运动计划中最多消耗{max(calories_list)}卡路里，最少消耗{min(calories_list)}卡路里")
```

```
卡路里列表： [0, 100, 200, 300, 200, 300, 400, 500, 400, 500, 600, 700, 600, 700, 800, 900]
运动计划中最多消耗900卡路里，最少消耗0卡路里
```

3.

```
products = [("牛奶", 5), ("鸡蛋", 20), ("香蕉", 10), ("杯子", 10)]
shopping_list = []
money = float(input("请输入您的购物资金："))
while True:
    print("*"*30)
    print("商品列表如下：")
```

```

for index,product in enumerate(products):
    print(f"{index+1}. 商品: {product[0]}, 价格:
{product[1]}")
print("*" * 30)
option = input("请输入您要购买的商品(退出请键入 q): ")
if option.isdigit():
    option = int(option)
    if 0 <= option-1 < len(products):
        option_product = products[option - 1]
        if option_product[1] <= money:
            shopping_list.append(option_product)
            money -= option_product[1]
            print("购买成功!")
        else:
            print(f"您的余额不足, 余额为: {money}")
    else:
        print("您选的商品不存在!")
elif option == "q":
    print("-" * 10, "购物清单", "-" * 10)
    for item in shopping_list:
        print(f"已购商品: {item[0]}, 价格: {item[1]}")
    print("您的余额为: ", money)
    break
else:
    print("您的输入不合法!")

```

```

请输入您的购物资金: 100
*****
商品列表如下:
1.商品: 牛奶, 价格: 5
2.商品: 鸡蛋, 价格: 20
3.商品: 香蕉, 价格: 10
4.商品: 杯子, 价格: 10
*****
请输入您要购买的商品(退出请键入q): 2
购买成功!

```

```

*****
商品列表如下:
1.商品: 牛奶, 价格: 5
2.商品: 鸡蛋, 价格: 20
3.商品: 香蕉, 价格: 10
4.商品: 杯子, 价格: 10
*****
请输入您要购买的商品(退出请键入q): q
----- 购物清单 -----
已购商品: 鸡蛋, 价格: 20
您的余额为: 80.0

```

## 四 字典与集合

### 1. 实验要求

- (1) 可以使用不同方法定义字典以及对字典进行操作
- (2) 可以使用不同方法定义集合以及对集合进行操作

### 2. 实验目的

(1) 掌握创建字典的方法以及多种字典的操作，包括访问、添加、删除

(2) 掌握创建集合的方法以及多种集合的操作，包括添加、删除

### 3. 实验内容

(1) 创建一个保存学生信息的字典，包括其姓名、学号和年级，并通过元素的键访问并修改元素的值，向该字典中添加学生信息并删除字典中不再需要的信息。

(2) 创建一个集合，在集合中实现添加元素和删除元素，利用集合判断一个列表中是否存在重复元素。

参考代码如下：

1.

```
student_dict = dict(name="小千", stu_id="202201", grade="大二")
print(student_dict)
print("学号为: ", student_dict["stu_id"])
student_dict["age"] = "20"
print(student_dict)
del student_dict["age"]
print(student_dict)
```

```
{'name': '小千', 'stu_id': '202201', 'grade': '大二'}
学号为: 202201
{'name': '小千', 'stu_id': '202201', 'grade': '大二', 'age': '20'}
{'name': '小千', 'stu_id': '202201', 'grade': '大二'}
```

2.

```
set01 = set("qianfeng")
set02 = set(("小千", "小锋"))
set03 = set(["小千", "小锋"])
set04 = set({"小千": 19, "小锋": 18})
print("set01:", set01)
print("set02:", set02)
```



```

print("set03:", set03)
print("set04:", set04)

language_set = {"汉语", "英语", "法语"}
language_set.add("俄语")
print(language_set)
language_set.discard("英语")
print(language_set)
language_set.remove("法语")
print(language_set)

```

```

set01: {'q', 'i', 'f', 'n', 'e', 'g', 'a'}
set02: {'小锋', '小千'}
set03: {'小锋', '小千'}
set04: {'小锋', '小千'}
{'俄语', '汉语', '法语', '英语'}
{'俄语', '汉语', '法语'}
{'俄语', '汉语'}

```

```

lst=[1, 3, 5, 3, 4, 4, 2, 9, 6, 7]
set_lst=set(lst)
#set 会生成一个元素无序且不重复的可迭代对象，也就是我们常说的去重
if len(set_lst)==len(lst):
    print('列表里的元素互不重复！')
else:
    print('列表里有重复的元素！')

```

**列表里有重复的元素！**

## 实验五 函数与类和对象 学时:4

### 1. 实验要求

- (1)可以定义函数，向函数传递参数并获取返回值。
- (2)掌握变量的使用方法，可以对函数进行递归调用。
- (3)可以使用类以及方法设计人机猜拳游戏

### 2. 实验目的

- (1)掌握定义、使用函数的方法
- (2)掌握变量的使用方法，以及函数的递归调用
- (3)掌握类和方法的定义和使用

### 3. 实验内容

- (1)找出序列类型中的最大值、最小值并返回

(2)对一组数据进行快速排序，排序过程中将需要排序的数据分割成独立的两部分，其中一部分的数据比另一部分的所有数据都小，分别对两部分进行拆分并快速排序，排序过程可以用递归实现，直到数据有序为止。

(3)设计人机猜拳游戏，将游戏过程分解为玩家的动作、机器的动作以及人和机器的互动，分别用类实现。玩家赢则玩家得一分，机器赢则机器得一分。游戏结束后，统计总的猜拳次数，比较玩家和机器的得分，得分高的判为游戏胜利。

参考代码如下：

1.

```
import numpy as np
a = [1, 2, 3, 5, 4, 6, 10, 7]
b = np.array([[1, 2, 3, 5], [4, 6, 2, 6]])
print(np.max(b))  # 返回最大值
print(np.min(b))  # 返回最小值
```

```
6
1
```

2.

```
def partition(array, low, high):
    """
    分解过程
    :param array:整体数据
    :param low:数据的最左端
    :param high:数据的最右端
    :return:基准值的位置
    """
    left = low - 1
    pivot = array[high]
    for right in range(low, high):
        if array[right] <= pivot:
```

```

        left += 1
        array[left], array[right] =
array[right], array[left]
    array[left+1], array[high] = array[high], array[left+1]
    return left+1
def quickSort(array, low, high):
    """
    快速排序函数，无返回值，直接改变列表内容
    :param array:整体数据
    :param low:数据的最左端
    :param high:数据的最右端
    """
    if low < high:
        pivot = partition(array, low, high)
        quickSort(array, low, pivot-1)
        quickSort(array, pivot+1, high)
if __name__ == "__main__":
    list01 = [9, 3, 1, 5, 8, 6, 2, 4]
    quickSort(list01, 0, len(list01)-1)
    print(list01)

```

```
[1, 2, 3, 4, 5, 6, 8, 9]
```

3.

```

from random import *
class Player:
    def __init__(self, score=0):
        self.name = "玩家"
    def player_action(self, option):
        if option == 1:
            print(f"{self.name}出石头")
        elif option == 2:

```

#玩家的姓

#玩家的得分

#方法用于打印

玩家猜拳的动作

```

        print(f"{self.name} 出剪刀")
    elif option == 3:
        print(f"{self.name} 出布")
class Computer:
    def __init__(self, score=0):
        self.cname = "电脑"                #机器的姓名，默认为“电脑”
        self.score = score                  #机器的得分
    def computer_action(self, option):      #方法用于打印机器猜拳的动作
        if option == 1:
            print(f"{self.cname} 出石头")
        elif option == 2:
            print(f"{self.cname} 出剪刀")
        elif option == 3:
            print(f"{self.cname} 出布")
class PlayGame:
    count = 0                               #对战次数
    def __init__(self):
        self.player = Player()             #创建一个玩家实例
        self.computer = Computer()         #创建一个机器实例
    def show_result(self):
        print(f"一共比赛了 {PlayGame.count} 场")
        print(f"{self.player.name} 的最终得分为 {self.player.score}")
        print(f"{self.computer.cname} 的最终得分为 {self.computer.score}")
        if self.player.score > self.computer.score:
            print("恭喜您获得全场比赛的胜利！")
        elif self.player.score == self.computer.score:
            print("全场比赛平局啦！")

```

```

        else:
            print("很遗憾，本场比赛您失败了")
def start_game(self):
    print("-"*15,"欢迎进入猜拳游戏","-"*15)
    print("出拳规则：1. 石头    2. 剪刀    3. 布")
    print("*"*40)
    choose = input("是否开始游戏（y 表示是，n 表示否）？")

    while choose == "y":
        player_option = input("请选择您要出什么拳？")
        if player_option.isdigit():
            player_option = int(player_option)
            self.player.player_action(player_option)
        else:
            print("您的输入有误，请重新输入")
            continue
        computer_option = randint(1,3)
        self.computer.computer_action(computer_option)
        if player_option == computer_option:
            print("平局啦！")
        elif (player_option == 1 and computer_option == 2) \
            or (player_option == 2 and
computer_option == 3) \
            or (player_option == 3 and
computer_option == 1):
            print("玩家胜利啦！")
            self.player.score += 1
        else:
            print("玩家失败了，再接再厉哦")
            self.computer.score += 1
        PlayGame.count += 1
        choose = input("是否进入下一轮（y 表示是，n 表示否）")

```

```

        self.show_result()
if __name__ == "__main__":
    game = PlayGame()
    game.start_game()

```

```

----- 欢迎进入猜拳游戏 -----
出拳规则：1.石头  2.剪刀  3.布
*****
是否开始游戏（y表示是，n表示否）？ y
请选择您要出什么拳？ 1
玩家出石头
电脑出剪刀
玩家胜利啦！
是否进入下一轮（y表示是，n表示否）

```

## 六 面向对象程序设计与函数的高级特性

### 1. 实验要求

(1)可以利用基类以及派生类模拟薪资结算，实现不同薪资计算方法。

(2)可以利用函数设计用户登录系统，并定义系统中的聊天、购物等函数。

### 2. 实验目的

- (1)掌握基类、派生类等类的使用方法
- (2)掌握创建、使用函数的方法

### 3. 实验内容

(1)假设公司有三类员工，将员工定义为基类，三类员工分别继承基类中的属性，并定义自己的特殊属性，利用派生类实现不同的薪资计算方法。

(2)验证用户登陆信息，如果用户的账号和密码正确则可以执行聊天和购物函数，如果不正确则提示用户名或密码错误。

参考代码如下：

```

1.
class Employee:                                #员工类，作为基类
    def __init__(self,name):

```

```

        self.name = name                #定义属性 name
    def get_salary(self):                #定义获取薪资的方法
        pass

class Manager(Employee):                #定义产品经理类，继承 Employee
    类
    def __init__(self, name, salary=15000):
        super().__init__(name)          #继承父类属性
        self.salary = salary            #定义薪资 salary
    def get_salary(self):                #重写父类方法
        return self.salary
    def __str__(self):                  #重写__str__()方法
        return f"{self.name}的薪资是{self.get_salary()}"

class Programmer(Employee):            #定义程序员类，继承 Employee
    类
    def __init__(self, name, base_salary=12000, over_time=0):
        super().__init__(name)          #继承父类属性
        self.base_salary = base_salary  #定义基础工资 base_salary
        self.__over_time = over_time    #定义加班时长
    def get_salary(self):                #重写父类方法
        if self.__over_time < 0:
            raise ValueError("工作时长的输入有误")
        elif self.__over_time > 20:
            self.__over_time = 20        #加班时长不能超过 20 小时，超出
20 小时不计入薪资
        return self.base_salary + 100 * self.__over_time
    def __str__(self):                  #重写__str__()方法
        return f"{self.name}的薪资是{self.get_salary()}"

class SoftTest(Employee):              #定义测试工程师类，继承
Employee 类
    def __init__(self, name, base_salary=8000, bug_num=0):
        super().__init__(name)          # 继承父类属性
        self.base_salary = base_salary  #定义基础工资 base_salary
        self.bug_num = bug_num          #定义发现的错误个数 bug_num
    def get_salary(self):                #重写父类方法

```

```

        return self.base_salary + 5 * self.bug_num
    def __str__(self):
        #重写__str__()方法
        return f"{self.name}的薪资是{self.get_salary()}"
def main():
    #定义计算所有员工工资的函数
    employee_list = [
        Manager("宋江"), Manager("吴用"), Manager("公孙胜", 10000),
        Programmer("花荣"), Programmer("武松", 10000, 10), Programmer("林
        冲", 13000, 30),
        SoftTest("朱武"), SoftTest("蒋敬"), SoftTest("柴进", 9000, 100)
    ]
    for emp in employee_list:
        if isinstance(emp, Programmer):
            print("程序员: ", emp)
        elif isinstance(emp, Manager):
            print("产品经理: ", emp)
        else:
            print("测试工程师: ", emp)
if __name__ == "__main__":
    main()

```

```

产品经理: 宋江的薪资是15000
产品经理: 吴用的薪资是15000
产品经理: 公孙胜的薪资是10000
程序员: 花荣的薪资是12000
程序员: 武松的薪资是11000
程序员: 林冲的薪资是15000
测试工程师: 朱武的薪资是8000
测试工程师: 蒋敬的薪资是8000
测试工程师: 柴进的薪资是9500

```

2.

```

username = "小千"
password = "xiaoqian123"
user_status = False
type = input("请输入登入方式（社交账号或博客账号）: ")
def login(login_type):

```



```

def check(func):
    def wrapper(*args,**kwargs):
        global user_status
        if not user_status:
            if login_type == "社交账号" or login_type == "博客账号":
                user = input("请输入用户名: ")
                pwd = input("请输入密码: ")
                if user == username and pwd == password:
                    user_status = True
                else:
                    print("用户名或者密码错误!")
            else:
                print("此登入方式无法使用!")
        if user_status:
            return func(*args,**kwargs)
        return wrapper
    return check

@login(type)
def chat():
    print("聊天")

@login(type)
def shop():
    print("购物")

if __name__ == "__main__":
    chat()
    shop()

```

请输入登入方式（社交账号或博客账号）：~~社交账号~~

请输入用户名：~~user~~

请输入密码：~~123~~

用户名或者密码错误！

请输入用户名：~~小千~~

请输入密码：~~xiaoqian123~~

购物