计算思维通识教育

Computational Thinking

第2章 计算设备处理信息-使用编程语言

主讲人：曹轶臻

联系方式：caoyizhen@cuc.edu.cn

学习程序设计的目标，绝不是把每个学习者都培养成专业的程序员，
而是帮助每个人建立系统化的逻辑思维方式。

CONTENTS

计算机与网络空间安全学院
School of Computer and Cyber Sciences

计算思维通识教育
Computational Thinking

04

编写一段Python程序

文件与目录 4-1

Python实例 4-2

Python

计算机与网络空间安全学院
School of Computer and Cyber Sciences

4-1 文件与目录

文件File是磁盘上的命名位置，用于存储相关信息。它们用于将数据永久存储在非易失性存储器（例如硬盘）中。

当我们想要读取或写入文件时，我们需要先打开它。

完成后，需要将其关闭，以便释放与文件绑定的资源。

在 Python 中，文件操作按以下顺序进行：

1. Open a file
2. Read or write (perform operation)
3. Close the file

请参考： https://www.runoob.com/python3/python3-file-methods.html

4-1 文件与目录

1.打开文件

Python 有一个内置的 open() 函数来打开文件。此函数返回一个文件对象，可用于相应地读取或修改文件。

我们可以在打开文件时指定模式mode

Mode	Description
r	Opens a file for reading. (default)
t	Opens in text mode. (default)
b	Opens in binary mode.
w	Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
a	Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.
+	open for updating (reading and writing)

```
>>> f = open('test.txt') # 在当前目录中打开文件test.txt
>>> f = open('C:/Python38/README.txt') # 指定完整路径
```

open() 函数常用形式是接收两个参数：
文件名(filename)和模式(mode)

```
f1 = open('test.txt') # 相当于第二个参数是r或rt
f2 = open('test.txt', 'w') # 打开文件以文本模式写入
f3 = open('img.bmp', 'r+b') # 打开文件以二进制模式读写

# 打开文件以文本模式写入，可指定编码类型
f4 = open("test.txt", mode='w', encoding='utf-8')
```

4-1 文件与目录

2.读文件

要在 Python 中读取文件，我们必须以读取**r**模式打开文件

有多种方法可用于读文件

read() : This function reads the entire file and returns a string
readline() : This function reads lines from that file and returns as a string.
readlines() : This function returns a list where each element is single line of that file.


```
f = open('test.txt', 'r')

#读test.txt文件完整内容，以字符串形式赋值给变量text
text = f.read()
print(text)

f.close()
```

```
Jazz
Pop
Rock
Metal
Reggae
Blues
Electronic
Hip-hop
Classic
Disco
```

4-1 文件与目录

2.读文件

read() : This function reads the entire file and returns a string
readline() : This function reads lines from that file and returns as a string.
readlines() : This function returns a list where each element is single line of that file.


```
f = open('test.txt', 'r')
print(f.readline().rstrip())
print(f.readline().rstrip())
print('=======')

while True:
    line = f.readline()  # 只读取一行
    if not line:  # 判断是否读到文件末尾
        break
    else:
        print(line.rstrip())  # rstrip()删除字符串末尾的换行符

f.close()
```

```
Jazz
Pop
=======
Rock
Metal
Reggae
Blues
Electronic
Hip-hop
Classic
Disco
```

4-1 文件与目录

2.读文件

read() : This function reads the entire file and returns a string
readline() : This function reads lines from that file and returns as a string.
readlines() : This function returns a list where each element is single line of that file.


```
f = open('test.txt', 'r')
lines = f.readlines()

i = 0
for line in lines:
    s = line.rstrip().upper() # 消除换行符，改大写
    if i % 2 == 0:
        print(s + '  ', end='')
    else:
        print(s)
    i += 1
f.close()
```

```
JAZZ  POP
ROCK  METAL
REGGAE  BLUES
ELECTRONIC  HIP-HOP
CLASSIC  DISCO
```

4-1 文件与目录

3.写文件

使用**write()**方法完成写入字符串或字节序列（用于二进制文件）
此方法返回写入文件的字符数。

如果当前目录不存在userinput.txt，此句将在当前目录中新创建一个文件

```
#Open the file for reading and
#appending at the end of the file
text_file = open('userinput.txt','a+')

for i in range (1, 5):
    line = input("Enter data: ")+'\n'
    text_file.write(line)

#positioning at the beginning of file
text_file.seek(0)
print(text_file.read())

#don't forget to close the file
text_file.close()
```

如果打开模式改为w+会怎么样?

我们需要**小心w模式**，因为如果文件已经存在，它将覆盖到文件中。因此，所有先前的数据都将被删除（truncate）。

我们必须自己包含换行符(\n)以区分不同的行

seek() 函数用于设置文件指针的位置，而 tell() 函数用于返回当前文件指针的位置。

4-1 文件与目录

4.关闭文件

当我们对文件执行完操作后，我们需要正确关闭文件，释放与文件绑定的资源。

```
f = open("test.txt", encoding = 'utf-8')
# perform file operations
f.close()
```

这种方法并不完全安全。如果在我们对文件执行某些操作时**发生异常**，程序会退出而不关闭文件。

更安全的方法是使用try...finally块

```
try:
    f = open("test.txt", encoding = 'utf-8')
    # perform file operations
finally:
    f.close()
```

即使异常导致程序停止，也能保证文件正确关闭

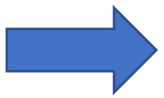
关闭文件的最佳方法是使用with语句。这可确保在with退出语句内的块时关闭文件。我们不需要显式调用该close()方法。它是在内部完成的。

```
with open("test.txt", encoding = 'utf-8') as f:
    # perform file operations
```

4-1 文件与目录

4.关闭文件

```
text_file = open('userinput.txt','a+')

for i in range (1, 5):
    print("Please enter data: ")
    line = input()+'\n'
    text_file.write(line)

text_file.seek(0)
print(text_file.read())

#don't forget to close the file
text_file.close()
```

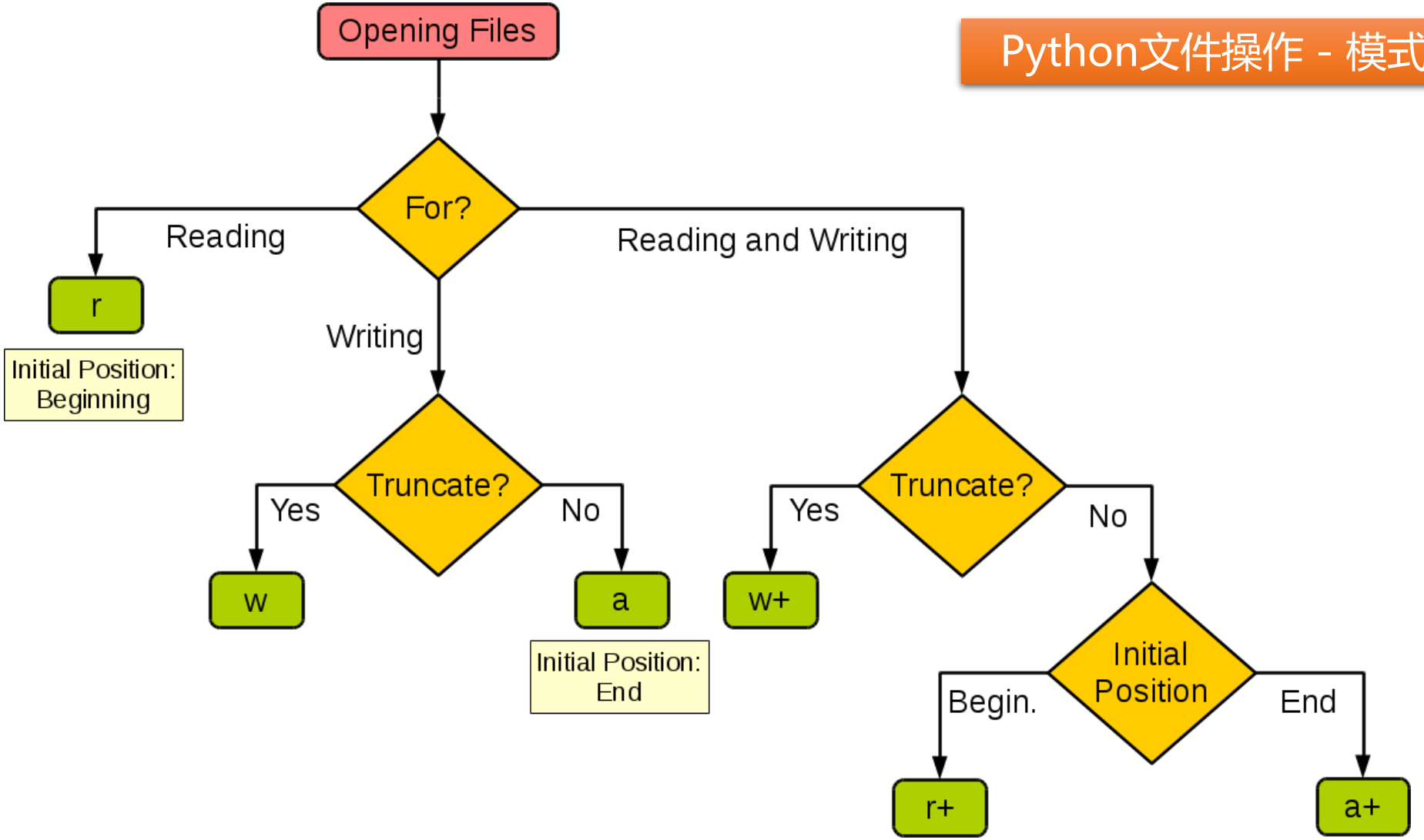


The **with** statement closes the file for you without you telling it to.

```
with open('userinput.txt', 'a+') as text_file:
    for i in range(1, 5):
        print("Please enter data: ")
        line = input() + '\n'
        text_file.write(line)

    text_file.seek(0)
    print(text_file.read())
```

4-1 文件与目录

Python文件操作 - 模式小结



图源：https://stackoverflow.com/questions/6648493/how-to-open-a-file-for-both-reading-and-writing

4-1 文件与目录

操作目录的常用方法

目录（directory）或文件夹（folder）是文件和子目录的集合。

Python 的 os 模块提供了许多处理目录（以及文件）的方法。

```
import os

# Get Current Directory
print(os.getcwd())

# Changing Directory
os.chdir('c:\\users')
print(os.getcwd())

# List Directories and Files
print(os.listdir())
print(os.listdir('d:\\'))
```

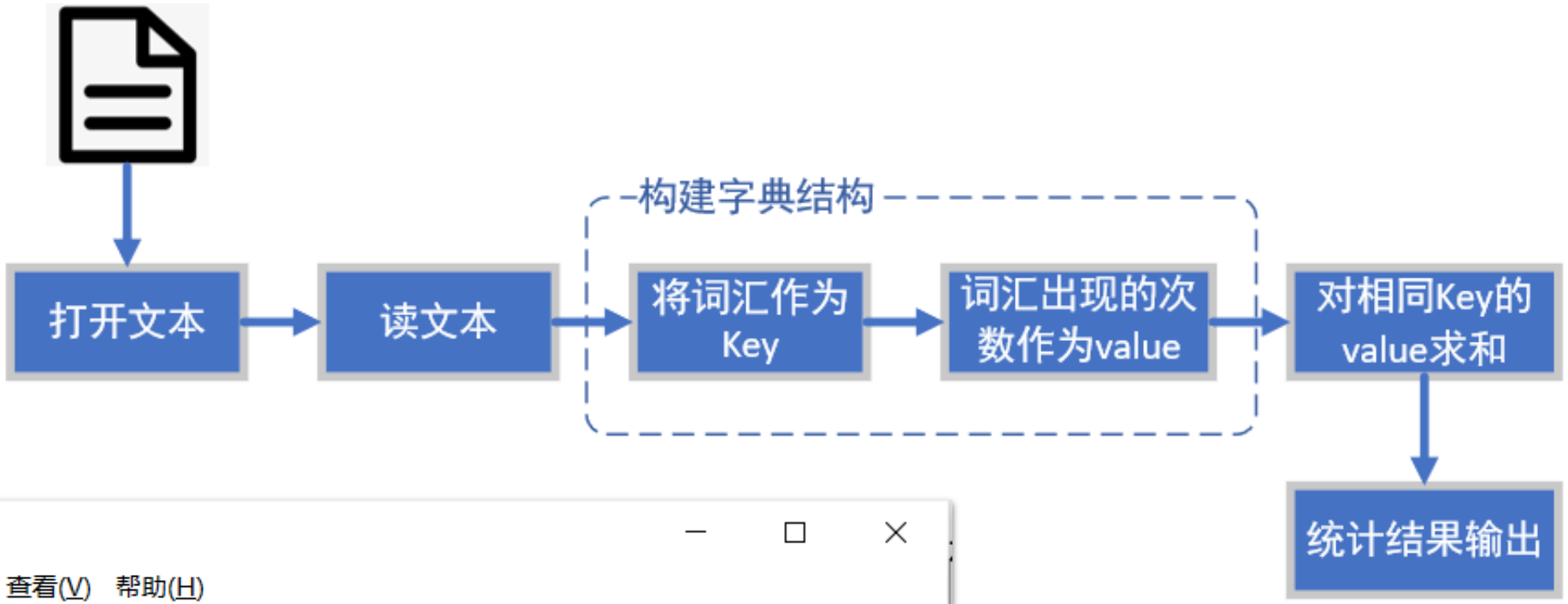
```
import os

# Making a New Directory
os.mkdir('test')

# Renaming a Directory or a File
os.rename('test', 'new_one')
os.rename('1.txt', 'new.txt')

# Removing Directory or File
os.remove('Old.txt')  # delete file
os.rmdir('new folder')  # remove directory
```

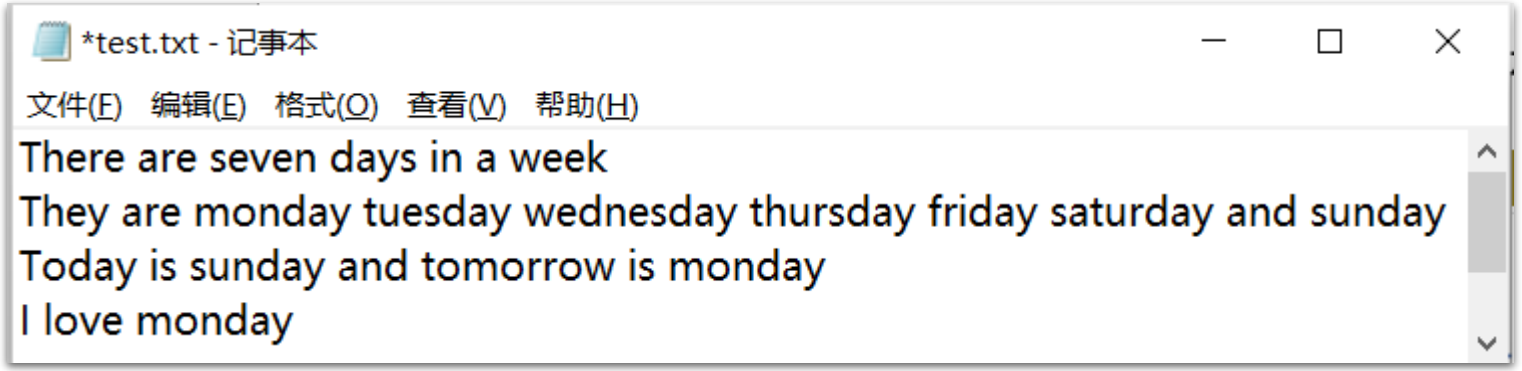
目录及文件的操作详见：https://docs.python.org/3/library/os.html#os-file-dir

4-2 Python实例（1.词频统计）

问题描述 统计文本内词汇出现的次数（词频统计问题），在做文本分析时经常需要

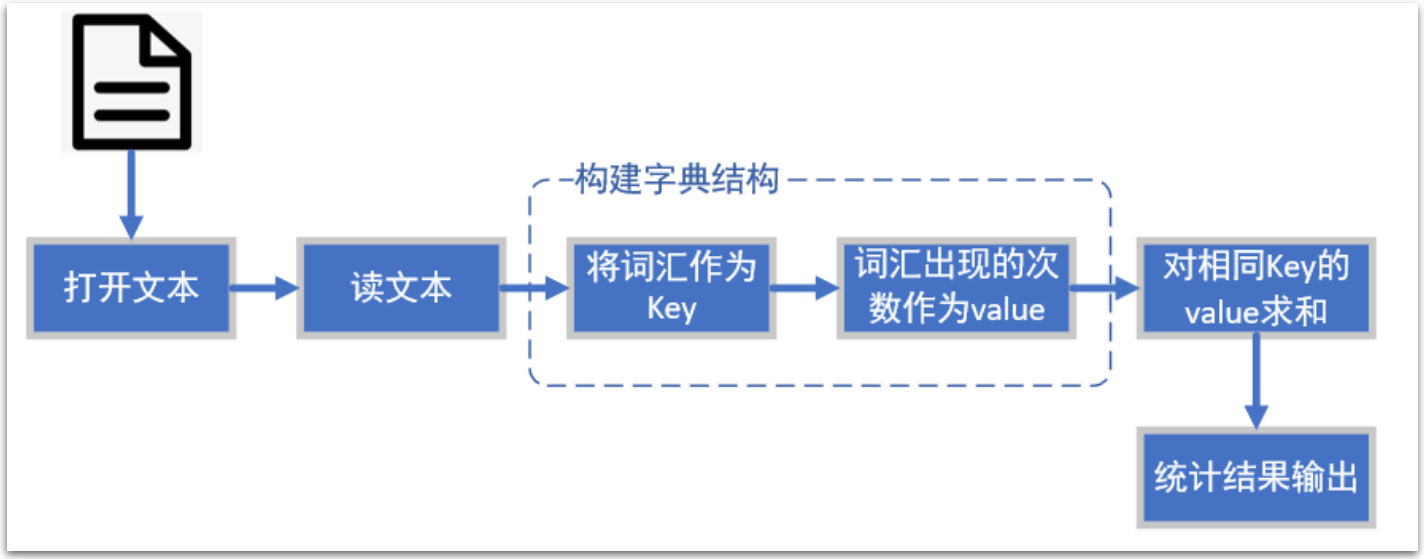
实现思路





4-2 Python实例（1.词频统计）

实现方案




```
with open('test.txt') as f:
    fulltext = f.read()

text_list = fulltext.split()
word_dict = {}
for word in text_list:
    if word in word_dict:
        word_dict[word] += 1
    else:
        word_dict[word] = 1

for word in word_dict:
    print(f'{word}:{word_dict[word]}')
```

4-2 Python实例（1.词频统计）

代码重构 – 使用模块和函数

```
There are seven days in a week
They are monday tuesday wednesday thursday friday saturday and sunday
Today is sunday and tomorrow is monday
I love monday
```

word_count.py

```
def count(text):
    text_list = text.split()
    dict1 = {}
    for w in text_list:
        if w in dict1:
            dict1[w] += 1
        else:
            dict1[w] = 1
    return dict1
```

main.py

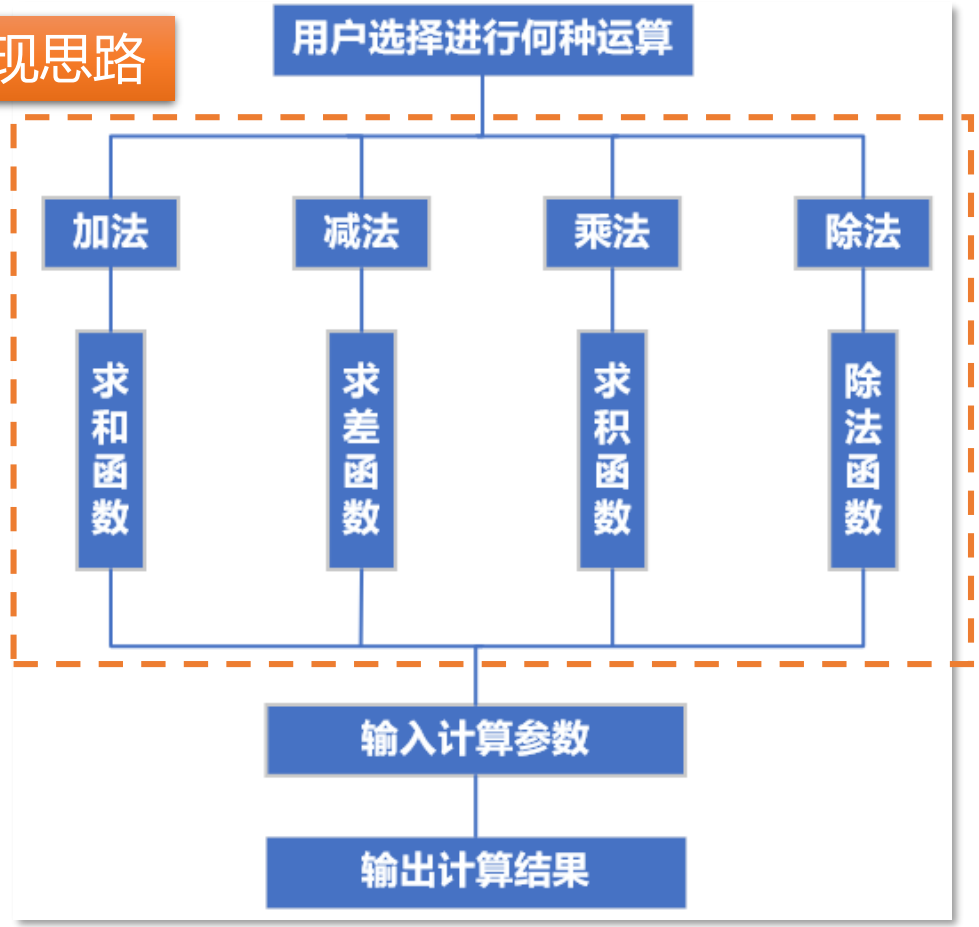
```
import wordcount

with open('test.txt') as f:
    fulltext = f.read()
    word_dict = wordcount.count(fulltext)

for word in word_dict:
    print(f'{word}:{word_dict[word]}')
```

```
There:1
are:2
seven:1
days:1
in:1
a:1
week:1
They:1
monday:3
tuesday:1
wednesday:1
thursday:1
friday:1
saturday:1
and:2
sunday:2
Today:1
is:2
tomorrow:1
I:1
love:1
```

4-2 Python实例（2.设计简单计算器）

问题描述 设计简单的计算器程序，实现加减乘除运算，要求用函数的方法实现基本功能

实现思路



MyCalculator.py

```
# Program make a simple calculator
# This function adds two numbers
def add(x, y):
    return x + y

# This function subtracts two numbers
def subtract(x, y):
    return x – y

# This function multiplies two numbers
def multiply(x, y):
    return x * y

# This function divides two numbers
def divide(x, y):
    return x / y
```

求和函数定义

求差函数定义

求积函数定义

求积函数定义

Python实例（2.设计简单计算器）

main.py

```
from MyCalculator import *

while True:
    # take input from the user
    choice = input('Enter choice(1/2/3/4):')
    # check if choice is one of the four options
    if choice in ('1', '2', '3', '4'):
        num1 = float(input('Enter first number:'))
        num2 = float(input('Enter second number:'))
        if choice == '1':
            print(num1, '+', num2, '=', add(num1, num2))
        elif choice == '2':
            print(num1, '-', num2, '=' , subtract(num1, num2))
        elif choice == '3':
            print(num1, '*', num2, '=', multiply(num1, num2))
        elif choice == '4':
            print(num1, '/', num2, '=' , divide(num1, num2))
        # check if user wants another calculation
        # break the while loop if answer is no
        next_calculation = input("Let's do next calculation? (yes/no): ")
        if next_calculation == "no":
            break
    else:
        print("Invalid Input")
```

确保程序的多轮执行

等待用户输入

如果用户输入的是“1”、“2”、“3”、“4”四个字符中的任何一个

系统接受用户输入两个数值num1和num2（转化为浮点型）

内嵌分支语句，实现不同函数调用

根据用户选择，决定是否进行下一轮运算，如果用户输入“no”，程序将退出。

如果用户输入的不是“1”、“2”、“3”、“4”四个字符中的任何一个，系统输出错误信息

本次课程结束，请继续加油！

Computational Thinking

计算机与网络空间安全学院
School of Computer and Cyber Sciences