Chapter 5 Python Programming

Python is a high-level scripting language that combines interpretability, compiler, interactivity and object-oriented. Created by Guido van Rossum and first released in 1991. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

The contents of this chapter are sketched in Figure 5-1.

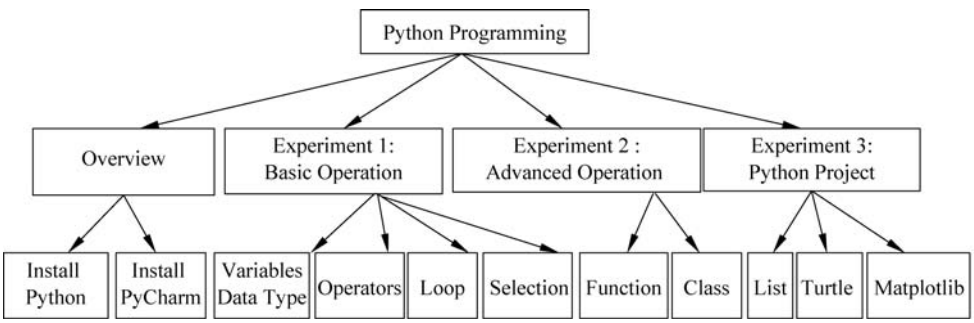

Figure 5-1 The relationship chart of this chapter

The structure of this chapter is shown in Figure 5-1, which is mainly composed of four parts.

(1) OverView mainly explains the installation process of Python and PyCharm. We can download the installation package of Python interpreter from the official website, and configure the python environment.

(2) Experiment 1 introduces the basic knowledge of Python programming. We need to master the basic simple variables and complex variables. Sequence structure, loop structure and conditional sentence need to be mastered. In this section, we can write some simple Python programs.

(3) Experiment 2 introduces the advanced features (function, class) of Python. These object-oriented features help us better solve complex problems and make the code concise.

(4) Experiment 3 introduces additional Python packages. These packages significantly improve our programming efficiency.

5.1 Building Python Environment

In this section, we will learn how to build a development environment for Python programming.

5.1.1 Install Python

In this section, we will learn how to download the Python installation package from the Python website. https://www.python.org is the official website of Python. We can download the installation package from here. Figure 5-2 is the official website of Python. Because the development environment of Windows is the majority, this section mainly explains how to download the Python installation package of Windows version. When we move the mouse to download, we can choose the corresponding Python version of the computer environment. As shown in Figure 5-2, we choose the Windows version.

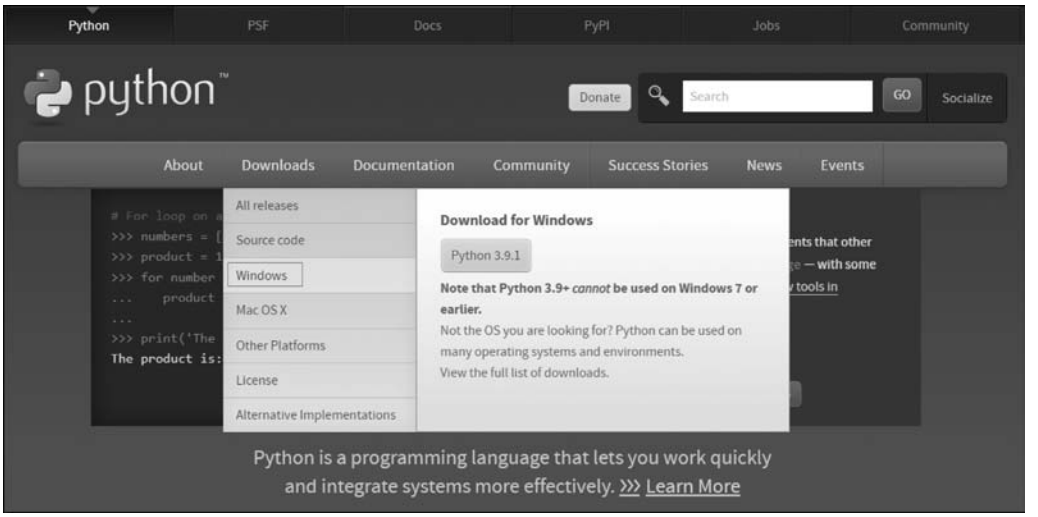

Figure 5-2 Python official website

There are many versions of Python here. We can download a stable version above 3.0. In Figure 5-3, number 1 is the 64 bit version of Windows and number 2 is the 32 bit version of Windows. "Windows x86-64 executable installer" is the easiest version to install, so we choose this version to install.

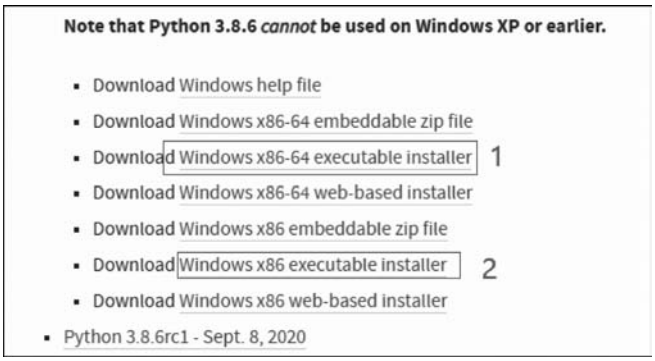

Figure 5-3 Select version of Python installation package

Figure 5-4 shows the installation package we downloaded. Double click the installation package to start the installation.

Figure 5-4 Python installation package

Figure 5-5 shows the initial installation interface. If you are installing Python for the first time, the interface will be different. Let's choose the option in the red box. Your interface will have a "Add Python 3.8 to PATH" option, check it. After checking this option, we don't need to set Python environment variables. Setting Python environment variables is a very troublesome thing, we must check it.

Figure 5-5 Install Python

As shown in Figure 5-6, we enter the "Optional Features" interface. We can choose what we need to install. Here I recommend checking all packages. Among them, "pip" installation package is very important.

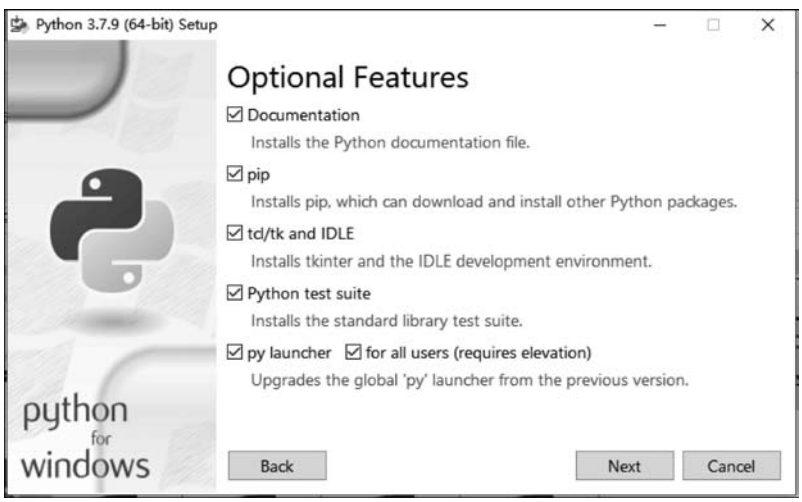

Figure 5-6 Optional Features of Python

In the last interface of installation shown in Figure 5-7, we can choose the path we want to install. This is very important for later study. Then click the "Install" button to install it automatically.

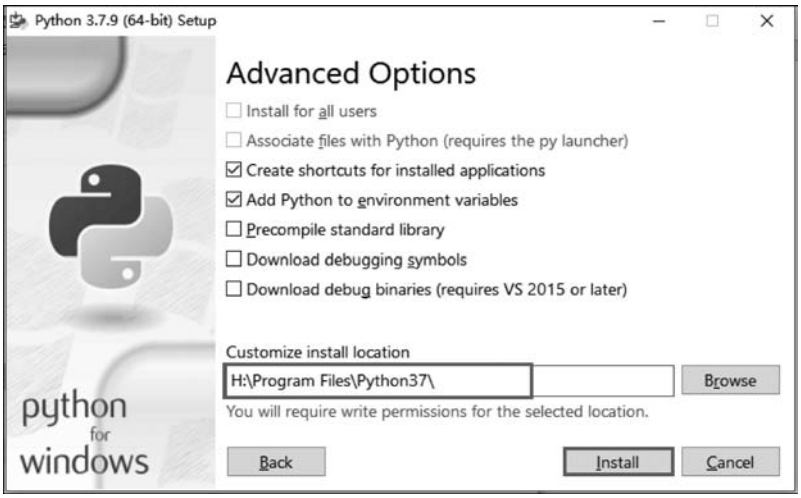

Figure 5-7 Advanced Options of Python

The Windows console can be opened to see if the Python installation is successful. Use the "Windows + R" button to open the console shown in Figure 5-8.

Figure 5-8 Open the terminal

Input "python" in the console. If the display is the same as Figure 5-9, it means that we have successfully installed.

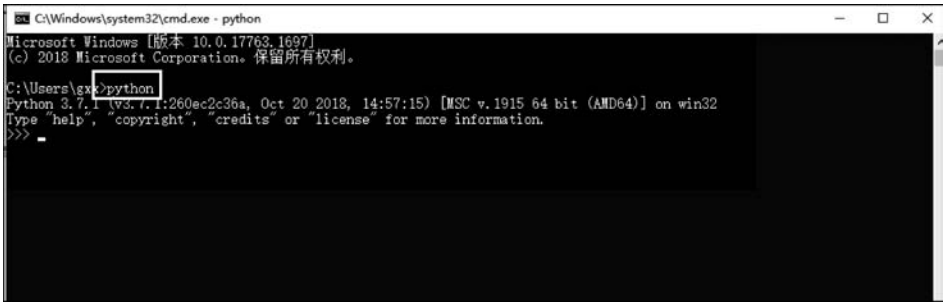

Figure 5-9 Check whether Python is installed successfully

As shown in Figure 5-10, let's write the first Python program. Enter "print ("Hello World")" in the console to output" Hello World ".

Figure 5-10 The first Python program

Press "Ctrl + Z" to exit the console shown in Figure 5-11. Now the Python compilation environment has been configured successfully.

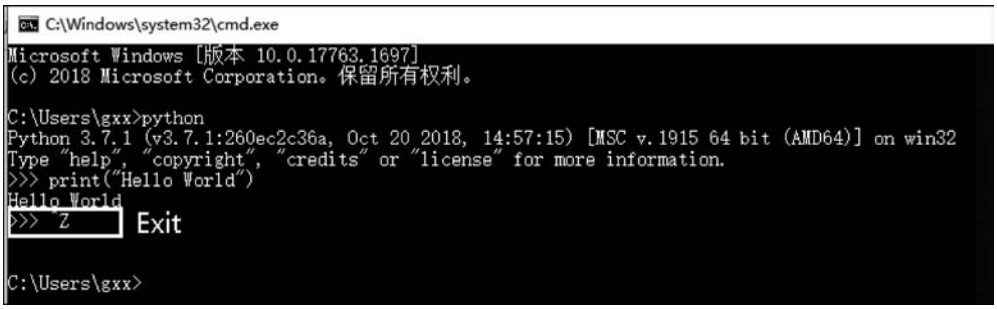

Figure 5-11 Exit the console

5.1.2 Install PyCharm

After configuring the Python environment, we can choose the integrated development environment for Python programming. Common Python integration environments include PyCharm, vscode and sublime. Here we take "PyCharm" as an example to introduce how to install "PyCharm". The download address of PyCharm is https://www.jetbrains.com/pycharm/download/#section=windows.

As shown in Figure 5-12, this is the download address of PyCharm. We choose the "Windows" option. There are two versions, "Professional" and "Community". The professional edition is more powerful, but it costs money. The community version is free, but it has fewer features. For the current study, we only need to download a community version.

After we download the PyCharm installation package, double-click the installation package, and the interface in Figure 5-13 will appear. We click the "Next" button.

We need to choose where to install. You can install according to the default location, or you can customize the location (shown in Figure 5-14).

Meaning of serial number as shown in Figure 5-15:

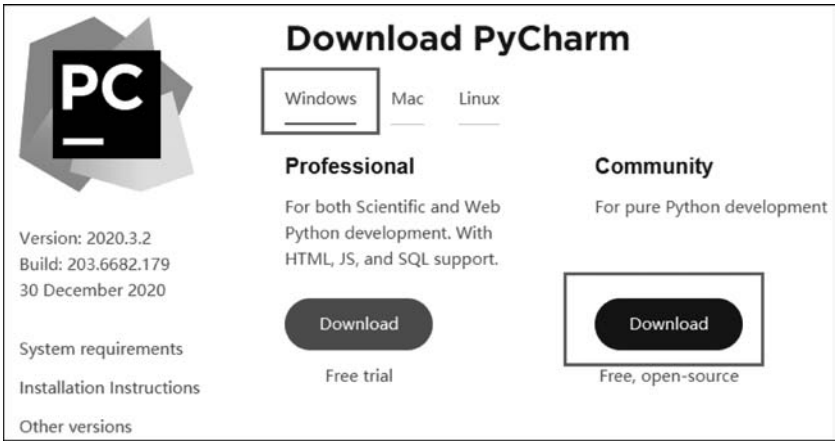

Figure 5-12 Download PyCharm

Figure 5-13 Install PyCharm

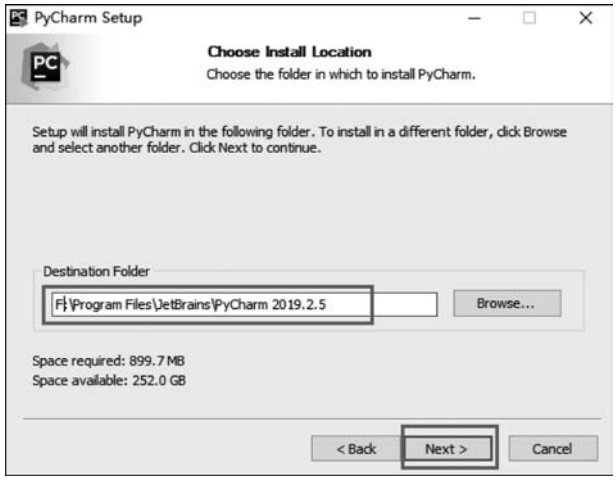

Figure 5-14 Choose install location

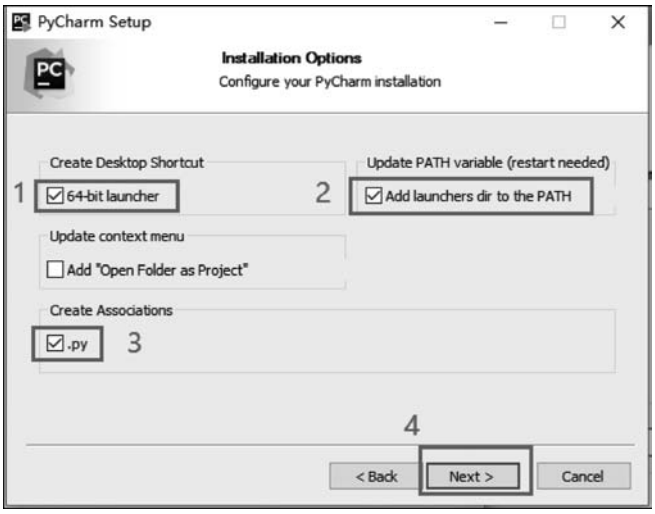

Figure 5-15 Installation Options

(1) Here we choose 64-bit. If your computer is 32-bit, you need to download the 32-bit version of PyCharm from the official website.

(2) Then we add the environment variable of PyCharm.

(3) After checking serial number three, all Python files are opened with PyCharm by default.

(4) Click the "Next" button to select the next step.

In Figure 5-16, click the "Install" button and the PC will start installing PyCharm.

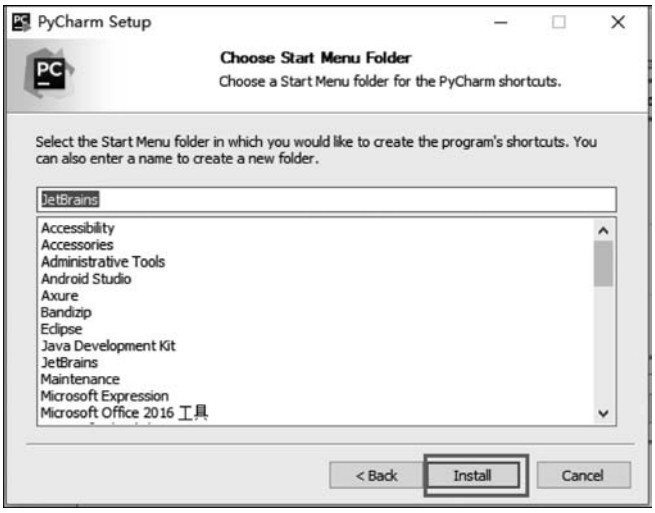

Figure 5-16 Click "Install" button

When the installation is finished, we need to create a new project. Click the "File"→"New Project" to create a new project as shown in Figure 5-17.

We need to choose the location of the new project and the corresponding interpreter. The

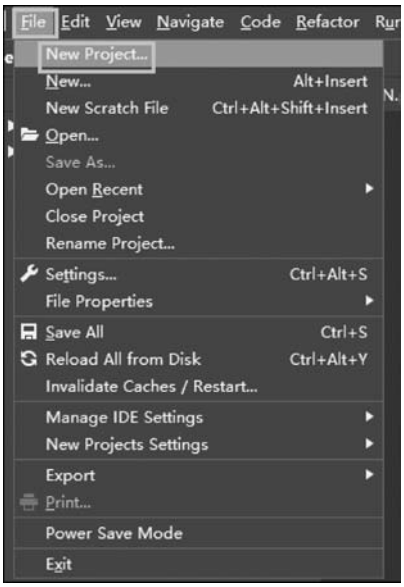

Figure 5-17 Create New Project

interpreter is where we install Python. Here, PyCharm will choose an explanation by default shown in Figure 5-18.

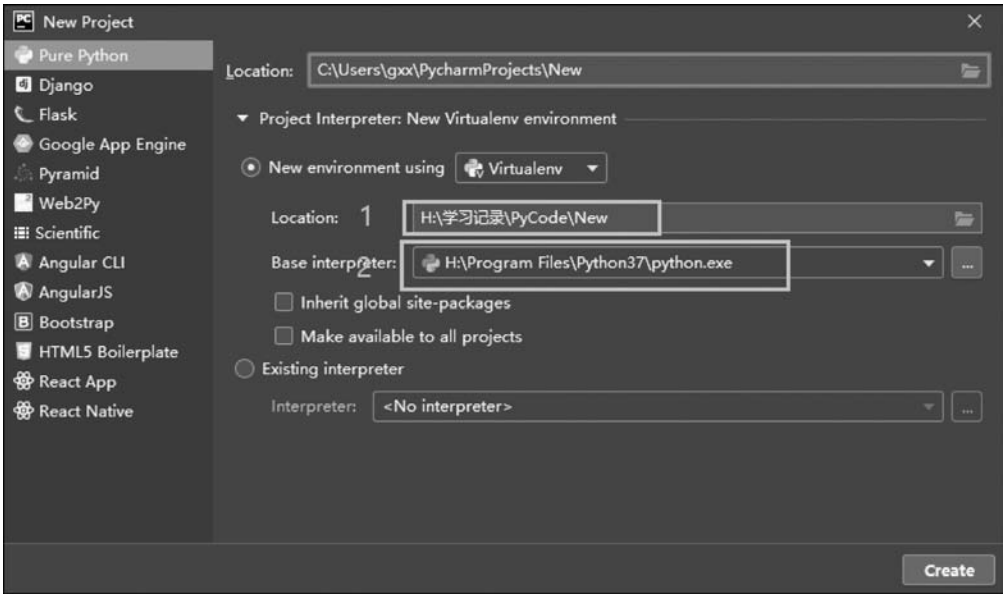

Figure 5-18 Setting of New Project

After we create a new project, right-click the folder of the new project, and the options in Figure 5-19 will appear. Here, we select "New" to create a new Python file, and then click the "Python File" option.

Let's give the python file a name as shown in Figure 5-20.

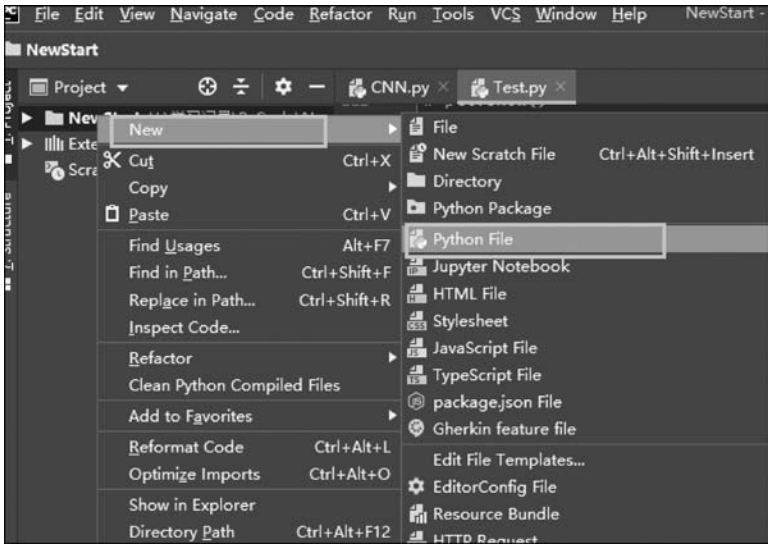

Figure 5-19 Create New Python File

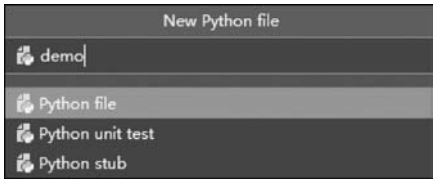

Figure 5-20 New Python file

At this point, we have created the Python file. We can copy the code we have written here. Here is a simple "Hello World" program as shown in Figure 5-21.

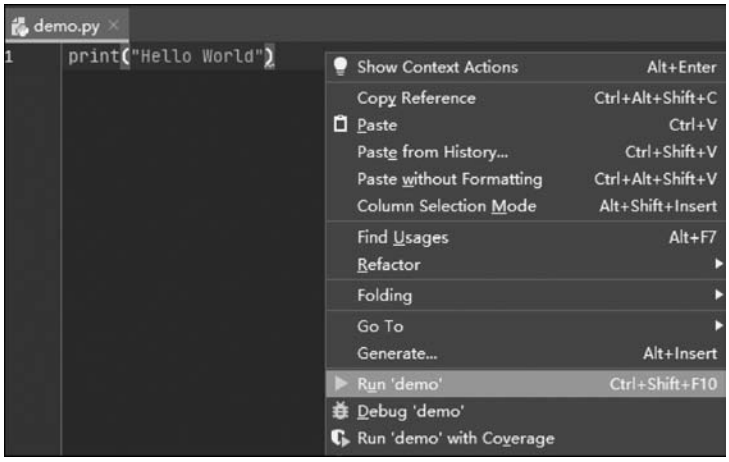

Figure 5-21 Run Python Program

The output of the demo code is "Hello World" as shown in Figure 5-22. That's how PyCharm is installed. There may be some differences during the installation, but they are

basically the same.

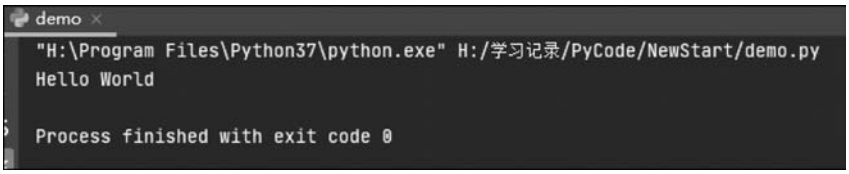

Figure 5-22 Output of demo

5.2 Experiment 1: Basic Operation

5.2.1 Experiment purpose

The purpose of this experiment is to be familiar with the python compiling environment and write basic Python programs. When there is a bug in the Python program, we can debug the program. The requirements of this experiment are as follows:

(1) We should master the use of various variables and operators.

(2) We should master the understanding and use of simple data type and complex data type (list, tuple, dictionary).

(3) We should master the usage of conditional sentence, loop structure and switch structure.

5.2.2 Experiment contents

1. Experiment 1.1: Run "Love"

(1) **Title Description**: Input the codes below and run it. The output of the codes is a sign of heart as shown in Figure 5-23.

(2) **Tips**:

- Create a Python file.
- Copy the code into a python file and run it.

(3) **Input**: None.

(4) **Source Code**:

```
① #Experiment 1.1 Run 'Love'
② print('\n'.join([''.join([(' *' [(x-y)% len(' *')]
③ if((x*0.05)**2+(y*0.1)**2-1)**3-(x*0.05)**2*(y*0.1)**3<=0 else ' ')
  for x in range(-30,30)]) for y in range(30,-30,-1)]))
```

(5) **Output**: as shown in Figure 5-23.

2. Experiment 1.2: Print "A"

(1) **Title Description**: Use " * " to print out a capital letter A as shown in Figure 5-24.

(2) **Tips**:

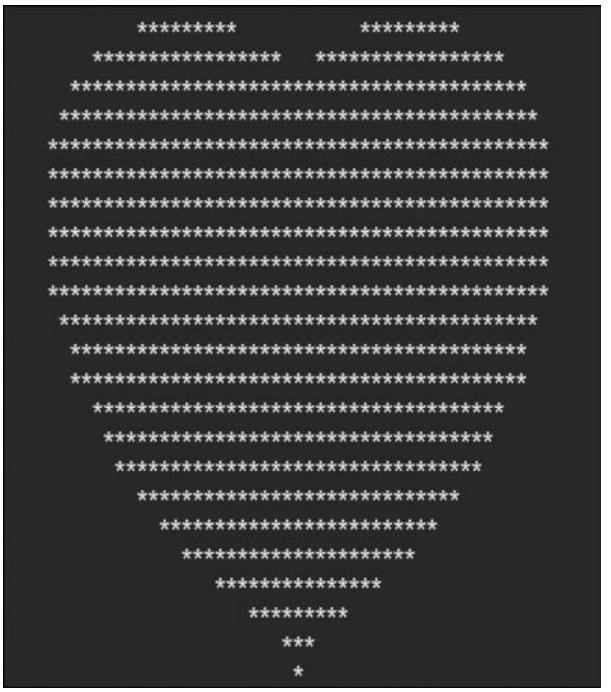

Figure 5-23 Run "Love"

- We can use the print function to print characters.
- When printing, we should add some blanks to make "A" look beautiful.

(3) **Input: None.**

(4) **Source Code:**

```
1. # Experiment 1.2 Print "A"
2. print("    A")
3. print("   AAA")
4. print("  A   A")
5. print(" AAAAAAA")
6. print("A       A")
```

Figure 5-24 Print "A"

(5) **Output:** as shown in Figure 5-24.

3. Experiment 1.3: Input and output of personal information

(1) **Title Description:** This experiment requires the input of your personal information (student number, name, sex, age, major and class), and the output of personal information, input need prompt information as shown in Figure 5-25.

(2) **Tips:**

- This experiment needs to master the "input" and "print" functions.
- As the input is a string, we also need to master int, str and other mandatory type conversion.

(3) **Input:** Your personal information.

(4) **Source Code:**

```
1.  #Experiment 1.3  Your personal information
2.  print("Please input your personal information :")
3.  id=int(input("Input your ID :"))
4.  name=input("Input your name :")
5.  gender=input("Input your gender :")
6.  age=input("Input your age :")
7.  print("Id:",id,end=' ')
8.  print("Name:",name,end=' ')
9.  print("Gender:",gender,end=' ')
10. print("Age:",age,end=' ')
```

(5) **Output**: as shown in Figure 5-25.

```
Please input your personal information
Input your ID :001
Input your name :Mike
Input your gender :male
Input your age :23
Id: 1 Name: Mike Gender: male Age: 23
Process finished with exit code 0
```

Figure 5-25 The output of input and output of personal information

4. Experiment 1.4: Integer operation

(1) **Title Description**: This experiment requires the input of two integers A and B, output their sum, difference, product, quotient, the remainder of A and B, the B power of A as shown in Figure 5-26.

(2) **Tips**: This experiment requires mastering addition, subtraction, multiplication, division and other operations, as well as "for", "loop" statements.

(3) **Input**: "12", "15".

(4) **Source Code**:

```
1.  #Experiment 1.4 Integer operation
2.  print("Please input integer A and B")
3.  A = input()
4.  B = input()
5.  A = int(A)  #The type of input is string
6.  B = int(B)  #we should adjust it to integer
7.  print("A+B=", A + B)
8.  print("A-B=", A - B)
9.  print("A*B=", A * B)
10. print("A/B=", A /B)
11. print("A%B=", A % B)
12. print("A^B=", pow(A, B))
```

```
Please input integer A and B
12
15
A+B= 27
A-B= -3
A*B= 180
A/B= 0.8
A%B= 12
A^B= 15407021574586368
```

Figure 5-26 The output of integer operation

(5) **Output**: as shown in Figure 5-26.

5. Experiment 1.5: The rank of score

(1) **Title Description**: This experiment requires inputing a student's score, the output of the student's score level. Above 90 is A-level, 80-90 is B-level, 70-80 is C-level, 60-70 is D-level, and below 60 is F-level.

(2) **Tips**:

- This experiment needs to use print and input functions.
- For the input student score, we need to use conditional judgment statements.
- The fraction you enter may be a decimal, so we use float to force the type conversion.

(3) **Input**: "99".

(4) **Source Code**:

```
1.  #Experiment 1.5 The rank of score
2.  print("Please input your score:")
3.  score = input()
4.  score = float(score)
5.  if score >= 90:
6.      print("A")
7.  elif score >= 80:
8.      print("B")
9.  elif score >= 70:
10.     print("C")
11. elif score >= 60:
12.     print("D")
13. else:
14.     print("F")
```

```
Please input your score:
99
A
```

Figure 5-27 The output of the rank of score

(5) **Output**: as shown in Figure 5-27.

6. Experiment 1.6: The sum of odd and even numbers from 1 to 100

(1) **Title Description**: In this experiment, we calculate the sum of all odd and even numbers from 1 to 100 is obtained respectively.

(2) **Tips**: The key of this experiment is to master the usage of range function and learn to use for loop structure.

(3) **Input**: None.

(4) **Source Code**:

```
1.  #Experiment 1.6 The sum of odd and even numbers from 1 to 100
2.  oddSum = 0
3.  evenSum = 0
4.  for i in range(1, 101, 2):
5.      oddSum += i
6.  for i in range(0, 101, 2):
7.      evenSum += i
8.
9.  print("The sum of odd numbers from 1 to 100 is :", oddSum)
10. print("The sum of even numbers from 1 to 100 is :", evenSum)
```

(5) **Output**: as shown in Figure 5-28.

```
The sum of odd numbers from 1 to 100 is : 2500
The sum of even numbers from 1 to 100 is : 2550
```

Figure 5-28 The output of the sum of odd and even numbers from 1 to 100

7. Experiment 1.7: Multiplication table

(1) **Title Description**: This experiment requires printing out a multiplication table.

(2) **Tips**: The key of this experiment is the understanding of range function and the flexible use of print function.

(3) **Input**: None.

(4) **Source Code**:

```
1.  #Experiment 1.7 Multiplication table
2.  for i in range(1, 10):
3.      for j in range(1, i + 1):
4.          print("%d * %d = %d" % (i, j, i * j), end='   ')
5.      print()
```

(5) **Output**: as shown in Figure 5-29.

```
1*1=1
2*1=2  2*2=4
3*1=3  3*2=6  3*3=9
4*1=4  4*2=8  4*3=12  4*4=16
5*1=5  5*2=10  5*3=15  5*4=20  5*5=25
6*1=6  6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1=7  7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1=8  8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1=9  9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

Figure 5-29 The output of multiplication table

8. Experiment 1.8: Chicken and rabbit in the same cage

(1) **Title Description**: There are many chickens and rabbits in a cage. There are 23 heads and 94 feet in the cage. How many chickens and rabbits are there in the cage?

(2) **Tips**: This experiment is actually a problem of solving equations. Since the amount of calculation in this experiment is not particularly large, we can use the exhaustive method.

(3) **Input**: The number of heads and feet in the cage.

(4) **Source Code**:

```
1.  #Experiment 1.8 Chicken and rabbit in the same cage
2.  Heads = int(input("Input the number of heads in the cage:"))
3.  Feet = int(input("Input the number of feet in the cage:"))
4.  for x in range(1, Heads):
5.      y = Heads - x
6.      if 2 * x + 4 * y == Feet:
7.          print("The number of Chicken is :", x)
8.          print("The number of Rabbit  is :", y)
```

(5) **Output**: as shown in Figure 5-30.

```
Input the number of heads in the cage:120
Input the number of feet in the cage:350
The number of Chicken is : 65
The number of Rabbit  is : 55
```

Figure 5-30 The output of chicken and rabbit in the same cage

9. Experiment 1.9: Temperature conversion program

(1) **Title Description**: Enter a temperature in the console and convert it to another standard temperature. If the input temperature is in Centigrade, it is converted to Fahrenheit. If you input Fahrenheit, convert it into Centigrade. If the input content does not conform to the two temperature formats, then output "error".

(2) **Tips**:

- This experiment uses conditional statements, list data types, input and output and other knowledge points.
- Equation (1) gives the temperature conversion method. In equation (1), C is the temperature in centigrade and F is the temperature in Fahrenheit.

$$1.8 * C + 32 = F \tag{1}$$

(3) **Input**: "25c", "25C","80f","39".

(4) **Source Code**:

```
1.  #Experiment 1.9 Temperature conversion program
2.  val = input("please input the temperature:")
3.  if (val[-1] in ['c','C']):
4.      result = 1.8 * float(val[0:-1]) + 32
5.      print("it is:" + str(result) + "F")
6.  elif (val[-1] in ['f','F']):
7.      result = 5 /9.0 * (float(val[0:-1]) - 32)
8.      print("it is: " + str(result) + "C")
9.  else:
10.     print("error")
```

(5) **Output**: as shown in Figure 5-31.

```
please input the temperature25c
it is:77.0F
```

Figure 5-31 The output of temperature conversion program

5.2.3 Self test practice

1. Practice 1: The rank of weight

(1) **Title Description**: Table 5-1 gives the weight criteria for adult men (Unofficial). Input the weight of an adult male and output the corresponding weight level.

Table 5-1 The adult male weight standard

Weight range (kg)	Rank
Weight≤50	Thin
50<Weight≤75	Normal
75<Weight	Fat

(2) **Tips**: This experiment can refer to the Experiment 1.4. The key of this experiment is the use of "if" sentence.

(3) **Input**: The weight of an adult male.

(4) **Output**: The rank of weight.

2. Practice 2: Sum of fractions

(1) **Title Description**: In this practice, we need to calculate the result of formula (2).

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{100} \tag{2}$$

(2) **Tips**: This practice can refer to the Experiment 1.5. This practice examines the use of cyclic structures.

(3) **Input**: None.

(4) **Output**: The result of formula (2).

3. Practice 3: The surface area and volume of a sphere

(1) **Title Description**: Formula (3) is the surface area formula of the ball. Formula (4) is the volume formula of the ball. Input the radius of a ball, output the surface area and volume of the ball.

$$S = 4\pi R^2 \tag{3}$$

$$V = \frac{4}{3}\pi R^3 \tag{4}$$

(2) **Tips**: This experiment can refer to experiment 1.3. Here we can use loop structure or POW function to calculate cubic power.

(3) **Input**: The radius of the ball.

(4) **Output**: The surface area and volume of the ball.

4. Practice 4: Print right triangle

(1) **Title Description**: In this practice, we can use " * " to print out a right triangle. Both right angles of a triangle are five. There are less " * " at the top and more " * " at the bottom of the right triangle.

(2) **Tips**:

- We need to use a double loop to print out the right triangle in the figure below.
- The outer loop controls the number of layers of the triangle.
- The inner loop is used to print the number of " * " in each layer.

Figure 5-32 The output of print right triangle

(3) **Input**: None.

(4) **Output**: as shown in Figure 5-32.

5.3 Experiment 2: Advanced Operation

5.3.1 Experiment purpose

Through the study of Experiment 1, we have mastered the basic ability of Python programming. In Experiment 2, we will learn more advanced features of Python to deepen our understanding of Python programming. Python is an object-oriented language, whose basic characteristics are inheritance, encapsulation and polymorphism. Specifically, we should master the knowledge of functions, classes and objects.

(1) **Function**: The Python language includes many built-in functions, such as print, max, etc., and users can customize functions. A function is a block of code that can be called repeatedly. Functions are organized, reusable code segments that implement single, or associated functions. Using functions, you can effectively organize your code and improve the reuse of code. The structure of the function is as follows:

```
def functionName(arg1, arg2,…):
    functionContents
    return [expression]
```

The necessary components of a function are keywords (def, return), name of function, content of function and expression. After we define a function, we can call it with the function name.

(2) **Class**: Class binds data to functionality. To create a new class is to create a new object type, thus creating a new instance of that type. Class instances have a variety of properties to maintain their own state. Class instances also support methods (defined in a class) to change their state.

Compared with other programming languages, Python adds classes to the language with very little new syntax and semantics. It is the combination of class mechanism in C + + and Modula-3. Python classes provide all the standard features of object-oriented programming: class inheritance mechanism allows multiple base classes, derived classes can override any method of its base class, and a method can call methods with the same name in the base class. Objects can contain any number and type of data. Like modules, classes have Python's natural dynamic properties: they are created at run time and can be modified after creation. The structure of the class is as follows:

```
class className():
    def functionName1(self, arg1, arg2):
    def functionName2(self, arg1, arg2):
obj = className()
```

The necessary components of a class include keywords (class), variables and functions. In fact, a class is common feature abstracted from a series of things with the same characteristics.

5.3.2 Experiment contents

1. Experiment 2.1: Guessing game

(1) **Title Description**: This experiment is a guessing game. The computer randomly generates an integer from 1 to 100. You have three chances to guess. If you guess correctly within three times, the computer will output "Congratulation!". If the number you guess is smaller than the random number, the computer will output "Smaller!". If the number you guess is larger than the random number, the computer will output "Larger!".

(2) **Tips**:

- Use the input function to get your guess.
- Use the "while" loop to continuously get the number you want to guess.
- Compare the size of the number you guess with the size of the random number by using the comparison operator.
- Define an integer variable to calculate the number of times you guess.

(3) **Input**: An integer.

(4) **Source Code**:

```
1.  #Experiment 2.1 Guessing game
2.  from random import randint
3.  def GuessingGame():                  #Define a guessing game function
4.      x = randint(1, 100)              #Generating a random number
5.      i = 1                            #i is used to indicate the number of guesses
6.      x = 55
7.      number = int(input("Input an integer from 0 to 100 :"))
8.      while (True):
9.          if (x > number):             #It's larger than you guessed
10.             print("Larger!")
11.         elif (x < number):           #It's smaller than you guessed
12.             print("Smaller!")
13.         else:                        #You guessed right!
14.             print("Congratulations!")
15.             break
16.         i = i + 1
17.         if (i > 3):                  #You only have three chances
18.             print("You have guessed too many times. Try again!!!")
19.             break
20.             #Input the number you want to guess
21.         number = int(input("Input an integer from 0 to 100 :"))
22. GuessingGame()
```

(5) **Output**: as shown in Figure 5-33 and Figure 5-34.

2. Experiment 2.2: Prime

(1) **Title Description**: This experiment is a mathematical problem. Input a range of numbers. Output all prime numbers in this range. Prime number refers to the natural number which has no other factors except 1 and itself.

```
Input an integer from 0 to 100 :22
Larger!
Input an integer from 0 to 100 :36
Larger!
Input an integer from 0 to 100 :59
Smaller!
You have guessed too many times. Try again!!!
```

Figure 5-33 You have guessed too many times. Try again!!!

```
Input an integer from 0 to 100 :76
Smaller!
Input an integer from 0 to 100 :28
Larger!
Input an integer from 0 to 100 :55
Congratulations!
```

Figure 5-34 You guessed right!

(2) **Tips**:

- Use the input function to get the range of primes you want.
- Loop through all the numbers to see if they are prime. If it is a prime, it will be output. Here's a trick. Since the multiple of two is definitely not a prime number, we just need to find the prime number in the odd number. When the range of numbers is relatively large, the amount of calculation can be greatly reduced by half.
- For a number, as long as it can be divided by a number less than it, then it is not a prime number. Use a loop structure to divide by integers less than it in turn. Here's another trick, just traverse to its square root. You can do it yourself.

(3) **Input**: A range of integers.

(4) **Source Code**:

```
1.  #Experiment 2.2 Prime
2.  def Prime(num_range):   #Define a prime function
3.      for i in range(3, num_range + 1, 2):  #Traverse all even numbers
4.          for j in range(2, i):         #Judge whether the number is prime or not
5.              if (i % j == 0):
6.                  break
7.          else:
8.              print(i, end=' ')            #If it is a prime, it will be output
9.  num_range = int(input("Please input a numerical range :"))
10. print('2', end=' ')
11. Prime(num_range)
```

(5) **Output**: as shown in Figure 5-35.

```
Please input a numerical range :100
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Figure 5-35 All prime numbers from 1 to 100

3. Experiment 2.3: Leap Year

(1) **Title Description**: This experiment is to determine whether a year is a leap year. If you enter a leap year, exit the program. If you are not entering a leap year, you can enter a new year until you are entering a leap year. Ordinary leap year: the Gregorian calendar year is a multiple of 4, not a multiple of 100, which is an ordinary leap year. Century leap year: Gregorian calendar year is the whole hundred, must be a multiple of 400 is the century leap year.

(2) **Tips**:

- First, you need to write a leap year judgment function. This year must be divisible by 4, but not by 100. When this year can be divided by 400, it is also a leap year.
- Next, you need to write a circular structure to constantly determine whether the year you enter is a leap year.
- When the input is leap year, jump out of the cycle, end the program, otherwise continue to judge the input year.

(3) **Input**: Year.

(4) **Source Code**:

```
1.  #Experiment 2.3 Leap Year
2.  def isLeap(year):                #Define a leap year function
3.      flag = False
4.      if (((year % 4 == 0) & (year % 100 != 0)) | (year % 400 == 0)):
5.          flag = True
6.      return flag
7.
8.  while (True):                    #Judge whether this year is a leap year
9.      year = int(input("Please input a year :"))
10.     if (isLeap(year)):
11.         print("%d is leap year" % year)
12.         break
13.     else:
14.         print("%d is not leap year" % year)
```

(5) **Output**: as shown in Figure 5-36.

4. Experiment 2.4: Rectangle class

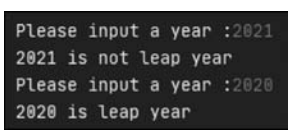

Figure 5-36 Judge if it's a leap year

(1) **Title Description**: This experiment requires the definition of a Rectangle class. This Rectangle class needs to include variables such as the length, width, and name of the rectangle. It also needs to define functions for area, side length, and rectangle name. Through this experiment, we should master the basic usage of class.

(2) **Tips**:

- First, we need to define the basic variables and function names needed by the rectangle class.
- Next, we implement the area function and perimeter function according to the specific rectangular area and perimeter formula.
- Input the length and width of the rectangle. Then instantiate the rectangle class and pass in these two parameters.

(3) **Input**: Year.

(4) **Source Code**:

```
1.  #Experiment 2.4 Rectangle class
```

```
class Rectangle():                  #  Define a rectangle class
    def __init__(self, name, Length, Width):
        self.name = name            #The Name of the rectangle
        self.Length = Length        #The Length of the rectangle
        self.Width = Width          #The Width of the rectangle

    def RectName(self):             #Output the Name of the rectangle
        print('The name of rectangle is :',self.name)
        return self.name

    def area(self):                 #Output the area of the rectangle
        area = self.Length * self.Width
        print('The area of rectangle is :%d' % area)
        return area

    def Perimeter(self):            #Output the perimeter of the rectangle
        perimeter = 2 * self.Length * self.Width
        print('The perimeter of rectangle is :%d' % perimeter)
        return perimeter

    def PrintRectangle(self):
        for i in range(self.Length):
            for j in range(self.Width):
                print('*', end=' ')
            print()
Name = input("Please input the Name of the rectangle :")
Width = int(input("Please input the Width of the rectangle :"))
Length = int(input("Please input the Length of the rectangle :"))
R = Rectangle(Name, Length, Width)  #Instantiation of rectangle class
R.RectName()
R.area()
R.Perimeter()
R.PrintRectangle()
```

(5) **Output**: as shown in Figure 5-37.

```
Please input the Name of the rectangle :001
Please input the Width of the rectangle :2
Please input the Length of the rectangle :3
The name of rectangle is :  001
The area of rectangle is : 6
The perimeter of rectangle is : 12
* *
* *
* *
```

Figure 5-37 The output of rectangle class

5. Experiment 2.5: Point class

(1) **Title Description**: This experiment requires the definition of a Point class. This Point class consists of three points. At the same time, this point class has the function of distance

between two points, the function of whether three lines can form a triangle, the function of triangle perimeter and the function of triangle area.

(2) **Tips**:

- First, we need to define a framework for the underlying classes.
- Next, we need to find the distance between any two points. Through these three distances, we can find out the perimeter, area and type of the triangle.
- Here we give the area formula of triangle.

$$S = \sqrt{P(P-a)(P-b)(P-c)} \tag{5}$$

Formula (5) is Helen's formula, where S is the area of the triangle, P is half of the circumference, and a, b and c are the distances of the three sides respectively

(3) **Input**: None.

(4) **Source Code**:

```
1.  #Experiment 2.5 Point class
2.  class Point():                    #  Define a point class
3.      def __init__(self, p1, p2, p3):
4.          self.p1 = p1              #Define three points
5.          self.p2 = p2
6.          self.p3 = p3
7.
8.      def distance(self):
9.  self.d12 = pow((pow((self.p1[0]-self.p2[0]),2)+pow((self.p1[1]-self.
    p2[1]),2)),0.5)
10. self.d13 = pow((pow((self.p1[0]-self.p3[0]),2)+pow((self.p1[1]-self.
    p3[1]),2)),0.5)
11. self.d23 = pow((pow((self.p2[0]-self.p3[0]),2)+pow((self.p2[1]-self.
    p3[1]),2)),0.5)
12.         print('The distance between P1 and P2 is %d' % self.d12)
13.         print('The distance between P1 and P3 is %d' % self.d13)
14.         print('The distance between P2 and P3 is %d' % self.d23)
15.
16.     def angle(self):              #Judge the type of triangle
17.          an1 = self.d23 * self.d23 + self.d13 * self.d13 - self.d12 *
             self.d12
18.          an2 = self.d23 * self.d23 + self.d12 * self.d12 - self.d13 *
             self.d13
19.          an3 = self.d13 * self.d13 + self.d12 * self.d12 - self.d23 *
             self.d23
20.         if(an1<0 or an2<0 or an3<0):
21.             print("Acute triangle")
22.         elif(an1==0 or an2==0 or an3==0):
23.             print("Right triangle")
24.         else:
25.             print("Obtuse triangle")
26.
27.     def area(self):               #Output the area of the triangle
28.         P=self.perimeter/2
```

```
        area=pow(P*(P-self.d12)*(P-self.d13)*(P-self.d23),0.5)
        print("The area of the triangle is :",area)
        return area
    def Perimeter(self):        #Output the perimeter of the triangle
        self.perimeter = self.d12+self.d13+self.d23
        print('The perimeter of triangle is :%d' % self.perimeter)
        return self.perimeter
T = Point([0,0], [3,0], [0,4])    #Instantiation of triangle class
T.distance()
T.Perimeter()
T.area()
T.angle()
```

(5) **Output**: as shown in Figure 5-38.

The output of point class is the distance of each edge, and judge whether three edges can form a triangle. If you can form a triangle, then output the area and perimeter of the triangle.

```
The distance between P1 and P2 is 3
The distance between P1 and P3 is 4
The distance between P2 and P3 is 5
The perimeter of triangle is : 12
The area of the triangle is : 6.0
Right triangle
```

Figure 5-38 The output of triangle class

5.3.3 Self test practice

1. Practice 1: Fibonacci sequence

(1) **Title Description**: This experiment requires the use of functions to find Fibonacci sequence. Fibonacci series, also known as the golden section series, is also known as the "Rabbit Series" because the mathematician Leonardoda Fibonacci introduced the rabbit breeding as an example.

(2) **Tips**:

- First, we need to build a function structure.
- Next, according to Fibonacci recurrence formula, write Fibonacci function. There are two ways to do it. The first one is realized by loop structure, which is a little complicated but fast. The second is to use recursion. Recursive structure code is simple, but the calculation speed is not fast enough.

$$F(n) = F(n-1) + F(n-2) \tag{6}$$
$$\text{s.t.} \quad F(0) = 1$$
$$F(1) = 1$$
$$n \geqslant 2$$

(3) **Input**: An integer.

(4) **Output**: A series of Fibonacci numbers.

2. Practice 2: Circle class

(1) **Title Description**: Similar to experiment 2.4 and experiment 2.5, this experiment requires the definition of a Circle class. The Circle class includes the center coordinate, radius, area function and perimeter function. It's better to try to print out the circle.

(2) **Tips**:

- First, we can define the structure of the circle class.
- Next, we calculate various parameters according to the formula of circle. The distance between the center of the circle and the origin can be realized by Euclidean distance formula. The formula of circumference and area of a circle is as follows:

$$S = \pi R^2 \tag{7}$$

$$P = 2\pi R \tag{8}$$

Where S is the area of the circle and R is the radius of the circle. The P is the circumference of the circle.

(3) **Input**: The radius and center coordinates of a circle.

(4) **Output**: The circumference, area and the distance from the center to the origin of a circle.

3. Practice 3: Date class

(1) **Title Description**: This experiment requires the implementation of a Date class. The Date class can output today's date, add a number of days at the same time, and then output the corresponding date. After adding the number of days, the date may go to the next year, and leap years need to be considered.

(2) **Tips**:

- First, input a date with year, month and day.
- Next, since the date entered may be a leap year, we need to write a judgment function for the leap year.
- The number of days added to the input date may exceed 365, so the year needs to be added by one.
- The days of each month are different, so we can use a list to record the days of each month.
- The method of this experiment is to find out the days that have passed in this year, and then add the new days. If the sum of days is more than one year, the year will be increased by one, and the days will be decreased, and the corresponding date will be calculated.

(3) **Input**: Input a date with year, month and day "2021.1.15", "2".

(4) **Output**: The date after adding the number of days "2021.1.17".

5.4 Experiment 3: Python Project

5.4.1 Experiment purpose

Through the study of Experiment 1 and Experiment 2, we have been able to skillfully master Python programming. In this experiment, we can integrate the previous knowledge to solve more complex problems. In the study of this experiment, we need to think more, and then use the

knowledge flexibly.

(1) **Data structure**: In this experiment, we mainly introduce Python data structure combined with the previous knowledge points. In Python, tuples and strings are immutable, while lists are imutable. In this experiment, we will take the list as an example to explore more complex operation of the list. Here we introduce the commonly used functions in the set.

① list. append ( ) adds an element to the end of the list.

② list. count (x) Calculate the number of elements x in the list.

③ list. index (x) Returns the index of the first element in the list with the value X. If there is no element, an error is returned.

Through some simple examples to understand these functions as shown in Figure 5-39.

```
1.  #Example
2.  List = [1, 2, 3, 4, 5]
3.  print(List)
4.  List.append(1)
5.  print(List)
6.  print(List.count(1))
7.  print(List.index(2))
```

(2) **Matplotlib**: Matplotlib is the drawing library of Python. It can be used with NumPy and provides an effective open source alternative to MATLAB. It can also be used with graphical toolkits such as pyqt and wxpthon. Matplotlib drawing is very simple and convenient. In this experiment, we can draw some simple pictures to learn shown in Figure 5-40.

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 1]
2
1
```

Figure 5-39 Some examples of list function

```
1.  import matplotlib.pyplot as plt
2.  X=[1,2,3,4,5,6,7,8,9,10,11,12]
3.  Y=[31,28,31,30,31,30,31,31,30,31,30,31]
4.  plt.bar(X,Y)
5.  plt.title("Days per month in 2021")
6.  plt.show()
```

(3) **Turtle**: Turtle first came from the logo language. It is specially used for children to learn programming. Through programming, it simulates a turtle crawling on the drawing board to draw patterns. Later, many high-level languages have transplanted turtle drawing. Python has added the turtle library to its internal library since 2.6. Because it is an internal library, you can import the turtle library by using the import turtle statement. The main steps of drawing are as follows: setting the sketchpad, setting the brush, controlling the turtle movement, drawing graphics, color filling and so on.

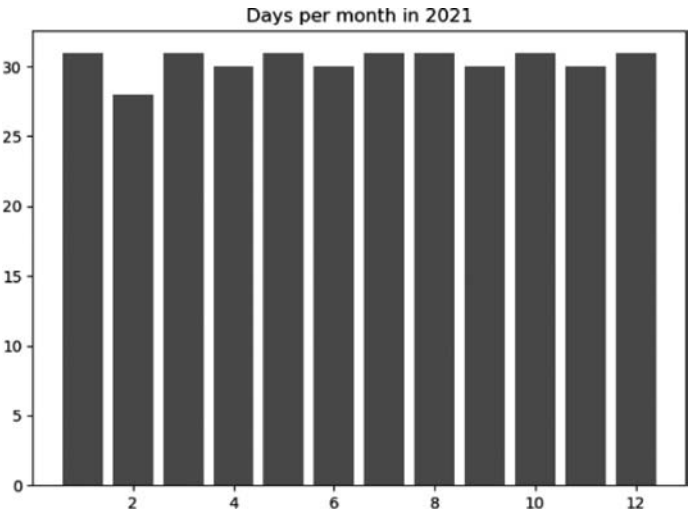

Figure 5-40 The picture for days per month in 2021

5.4.2 Experiment contents

1. Experiment 3.1: The number of times each word appears in a sentence

(1) **Title Description**: Give a passage, count the number of each word in the passage, and visualize the result with Matplotlib library.

(2) **Tips**:

- Use raw_ Input function input string S.
- Split the string s into a List with split function.
- Use a loop to duplicate List to generate a The_List without duplicate elements.

① Set the initial value of The_List as an empty list;

② Detect all element items in the List;

③ The operator not in is used to detect whether the element item is in The_List, and the append function is used to add the element item not in The_List to The_List.

- Use list List_count to count the number of occurrences of each element.

① Set the initial value of List_count as an empty list;

② Count function is used to count the times of all element items The_List;

③ Use the append function to add the count result to the List_count.

(3) **Input**: "I am a good student. Xiaohong is also a good student."

(4) **Source Code**:

```
1.  #Experiment3.1  The number of times each word appears in a sentence
2.  import matplotlib.pyplot as plt
3.  Sentence="I am a good student. Xiaohong is also a good student."
4.  def CountList():
5.      The_List = []
6.      List = Sentence.split(' ')  #Split the list
```

```
    print(List)
    for i in List:
        if i not in The_List:      #If i does not exist, add it to The_list
            The_List.append(i)
    List_count = []                #List_count is used to count

    for i in The_List:
        List_count.append(List.count(i))

    for i in range(len(The_List)):
        print(The_List[i], end=' ')
        print(List_count[i])
    return The_List,List_count

def Picture(The_List,List_count):
    plt.bar(The_List,List_count)
    plt.title("The number of times each word appears in a sentence")
    plt.show()

The_List,List_count=CountList()
Picture(The_List,List_count)
```

(5) **Output**: as shown in Figure 5-41 and Figure 5-42.

```
I 1
am 1
a 2
good 2
student. 2
Xiaohong 1
also 1
is 1
```

Figure 5-41 The output of the number of times each word appears in a sentence

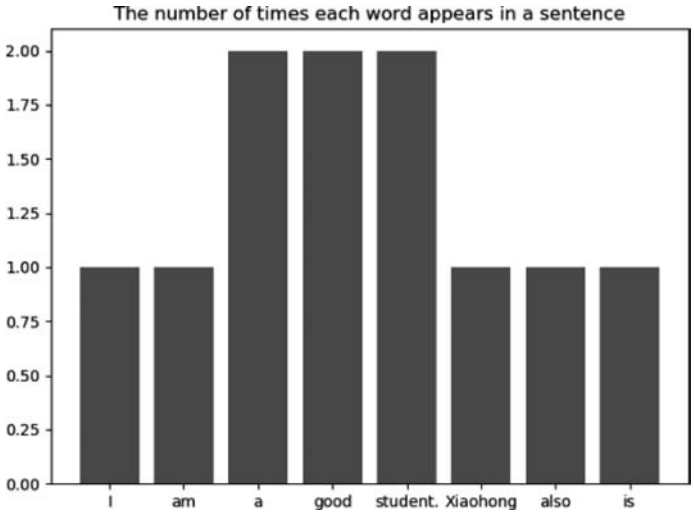

Figure 5-42 The visual output of the number of times each word appears in a sentence

2. Experiment 3.2: Draw a square

(1) **Title Description**: In this experiment, we need to draw a square with the Turtle library. The side length of the square is 60 pixels, the inside of the square is red, and the border of the square is blue.

(2) **Tips**:

- First, we use the turtle.reset Function to clear the screen.
- Next, set the properties of the brush. The width of the brush is 10 pixels, the color of the brush is blue, and the filling color of the brush is red.
  ① Use turtle.pensize sets the width of the brush.
  ② Use turtle.pencolor sets the color of the brush.
  ③ Use turtle.fillcolor sets the inner fill color of the square.
- Let the brush move and draw a square.
  ① Use turtle.forward Function to move the brush forward.
  ② Use turtle.left Let the brush turn left.

(3) **Input**: None.

(4) **Source Code**:

```
1.  #Experiment3.2  Draw a square
2.  import turtle
3.  turtle.reset()
4.  Width=60
5.  turtle.pencolor("blue")          #Set color of brush
6.  turtle.pensize(10)               #Set width of brush
7.  turtle.begin_fill()
8.  turtle.fillcolor("red")          #Set color of square
9.  turtle.left(90)                  #Rotate 90 degrees to the left
10. turtle.forward(Width)
11. turtle.left(90)
12. turtle.forward(Width)
13. turtle.left(90)
14. turtle.forward(Width)
15. turtle.left(90)
16. turtle.forward(Width)
17. turtle.end_fill()
18. turtle.done()
```

(5) **Output**: as shown in Figure 5-43.

3. Experiment 3.3: Draw a cube

(1) **Title Description**: Similar to Experiment 3.2, this experiment requires drawing a positive cube. The front of the cube is red, the top of the cube is blue, and the right side of the cube is black. The side length of a positive cube is 100 pixels and the border width is 10 pixels.

Figure 5-43 The output of square

(2) **Tips**:

- According to Experiment 3.2, we need to draw a square on the front.
- Next, we need to adjust the position of the brush. Draw two parallelograms at the top.
  ① Use turtle.penup function to lift the brush.
  ② Use turtle.goto function to move the brush to the specified position.
  ③ Use turtle.pendown function starts to draw a line.
- Draw the same parallelogram on the right.

(3) **Input**: None.

(4) **Source Code**:

```
1.  #Experiment 3.3 Draw a cube
2.  import turtle
3.  Width = 100
4.  turtle.goto(0, 0)                #First draw the square on the front
5.  turtle.pendown()
6.  turtle.pencolor('black')
7.  turtle.begin_fill()
8.  turtle.fillcolor('red')
9.  for i in range(4):
10.     turtle.forward(Width)
11.     turtle.left(90)
12. turtle.end_fill()
13.
14. turtle.penup()                   #Draw the square at the top
15. turtle.goto(0, 0 + Width)
16. turtle.pendown()
17. turtle.fillcolor('blue')
18. turtle.begin_fill()
19. turtle.left(45)
20. turtle.forward(int(Width * 0.6))
21. turtle.right(45)
22. turtle.forward(Width)
23. turtle.left(360 - 135)
24. turtle.forward(int(Width * 0.6))
25. turtle.end_fill()
26.
27. turtle.fillcolor('black')        #Draw the square on the right
28. turtle.begin_fill()
29. turtle.left(45)
30. turtle.penup()
31. turtle.goto(0 + Width, 0)
32. turtle.pendown()
33. turtle.left(135)
34. turtle.forward(int(Width * 0.6))
35. turtle.left(45)
36. turtle.forward(Width)
37. turtle.left(45)
38. turtle.forward(Width)
```

```
39. turtle.right(90)                    #方向还原,向左
40. turtle.penup()
41. turtle.end_fill()
42. turtle.done()
```

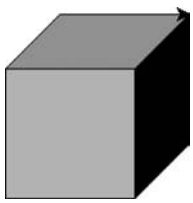

Figure 5-44 The output of cube

(5) **Output**: as shown in Figure 5-44.

4. Experiment 3.4: Student information management system

(1) **Title Description**: After the above study, we have mastered Python Programming skillfully. In this experiment, we want to implement a student information management system. The system can add, delete, modify and select student information. The system can count the age of students. So the functions of this system are as follows:

① Add student information.

② Delete student information.

③ Modify student information.

④ Show all student information.

⑤ Select information.

⑥ Count the number of students of each age.

(2) **Tips**:

- First, we can define a student class, which includes all students' information and various functions.
- For each student's information, we can use a dictionary to store it, and then use a list to synthesize all the students' information.
- Add student information. We can use dictionary data type to store a student's information. Then use the append function to add the student's information to the list.
- Delete student information. We can use the input function to enter the student number that we want to delete. Delete all the information of the student by the number of the student.
- Modify student information. Similar to the delete function, after entering the student number, we can modify the student information according to the index.
- Show all student information. Use the print function and loop structure to print out all the student information.
- Select information. In the query sub function, users should have a variety of options to query students' information. We can provide four kinds of query methods, according to the number, name, gender and number of students.
- Count the number of students of each age. In this sub function, we need to use Matplotlib package to count the age distribution of students.

(3) **Input**: None.

(4) Source code and output:

```
1.  #Experiment 3.4  Student information management system
```

```
2.  import matplotlib.pyplot as plt
3.  class Student():    #  Student information management system
4.      def __init__(self, stuInfo):
5.          self.stuInfo = stuInfo
6.      def Mymain(self):
7.          while True:
8.              self.printMenu()        #print Menu
9.              key = int(input('Please input the number corresponding to the
    function:'))
10.             if key == 1:
11.                 self.addInfo()      #Add student information
12.             elif key == 2:
13.                 self.delInfo()      #Delete student information
14.             elif key == 3:
15.                 self.modifystuInfo()   #Modify student information
16.             elif key == 4:
17.                 self.showstuInfo()  #Show all student information
18.             elif key == 5:
19.                 self.selectInfo()   #Select information
20.             elif key == 6:
21.                 self.ageStatistics()
                                        #Count the number of students of each age
22.             elif key == 7:          #Exit the system
23.                 quitConfirm = input('Do you want to quit?(Yes or No):')
24.                 if quitConfirm == 'Yes':
25.                     break           #Exit the system
26.                 else:
27.                     print('Input error, please input again !')
28.
29.     #print Menu
30.     def printMenu(self):
31.         print('=' * 30)
32.         print('Student information management system')
33.         print('1.Add student information')
34.         print('2.Delete student information')
35.         print('3.Modify student information')
36.         print('4.Show all student information')
37.         print('5.Select student information')
38.         print('6.Count the number of students of each age')
39.         print('7.Exit the system')
40.         print('=' * 30)
```

The menu function can print out the menu, and the user can select the corresponding function according to the menu prompt. The main function uses the loop structure to listen to the user's operation until the user chooses to exit. Above code are the source code of main function and menu function respectively.

```
41. #1. Add student information
42. def addInfo(self):
43.     newname = input('Input the name of the new student:')
```

```
44.     newsex = input('Input the sex of the new student:')
45.     newage = input('Input the age of the new student:')
46.     newInfo = {}
47.     newInfo['name'] = newname
48.     newInfo['sex'] = newsex
49.     newInfo['age'] = newage
50.     self.stuInfo.append(newInfo)
```

As shown in Figure 5-45, we can select our functions according to the menu prompts. We choose to add the student information function, and then input the name, gender and age of the new student. Figure 5-46 is the updated student information. As can be seen from the figure, we have added new student information.

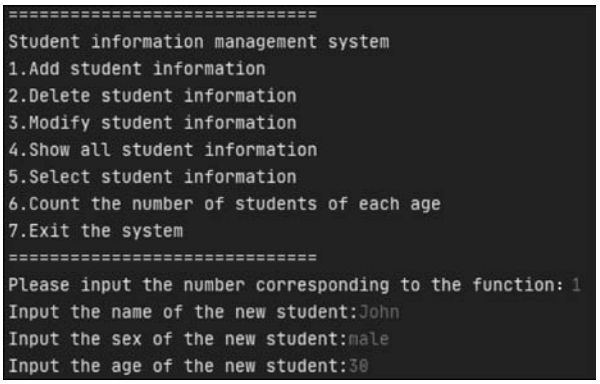

Figure 5-45 The operation of adding student information

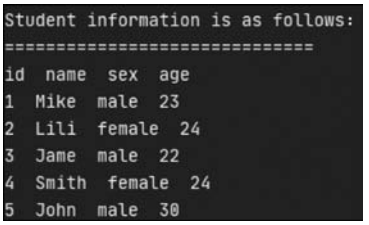

Figure 5-46 The new student information

```
51. #2. Delete student information
52. def delInfo(self):
53.     delNum = int(input('Input ID of student:')) - 1
54.     print("Delete information for student id is ",delNum+1)
55.     del self.stuInfo[delNum]
```

As shown in Figure 5-47, we choose the function of deleting student information. We need to give the student's id, and then delete the student 5. The updated student information is shown in Figure 5-48.

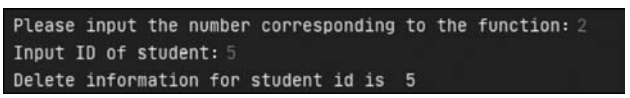

Figure 5-47 The operation of deleting student information

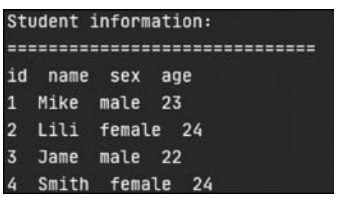

Figure 5-48 The updated student information

```
56. #3. Modify student information
57. def modifystuInfo(self):
58.     stuId = int(input('Input ID of student:')) - 1
59.     newname = input('Input name of student:')
```

```
60.     newsex = input('Input sex of student:')
61.     newage = input('Input age of student:')
62.     self.stuInfo[stuId]['name'] = newname
63.     self.stuInfo[stuId]['sex'] = newsex
64.     self.stuInfo[stuId]['age'] = newage
```

The functions of modifying student information are shown in Figure 5-49. We can select a student number and modify the student's personal information. In Figure 5-50, the updated student information is shown.

```
Please input the number corresponding to the function: 3
Input ID of student: 2
Input name of student:Lili
Input sex of student:female
Input age of student:18
```

Figure 5-49 The operation of deleting student information

```
Student information:
============================
id  name  sex  age
1  Mike  male  23
2  Lili  female  18
3  Jame  male  22
4  Smith  female  24
```

Figure 5-50 The updated student information

```
65. #4. Show all student information
66. def showstuInfo(self):
67.     print('=' * 30)
68.     print('Student information:')
69.     print('=' * 30)
70.     print("id  name  sex  age")
71.     i = 1
72.     for tempInfo in self.stuInfo:
73.         print('%d  %s  %s  %s' % (i, tempInfo['name'], tempInfo['sex'],
    tempInfo['age']))
74.         i += 1
75.
76. def selectStu(self, key , information):
77.     for i in range(len(self.stuInfo)):
78.         if (self.stuInfo[i][key] == information):
79.             print(i, end=' ')
80.             print(self.stuInfo[i]['name'], end=' ')
81.             print(self.stuInfo[i]['sex'], end=' ')
82.             print(self.stuInfo[i]['age'], end=' ')
83.             print()
84.
85. #5. Select student information
86. def selectInfo(self):
87.     print("  0.Id")
88.     print("  1.Name")
89.     print("  2.Sex")
90.     print("  3.Age")
91.     key = input('Select method:')
92.     if key=='0':
93.         id = input('Input id of student:')
94.     elif key=='1':
95.         name = input('Input name of student:')
```

```
96.         self.selectStu('name',name)
97.     elif key=='2':
98.         sex=input('Input sex of student:')
99.         self.selectStu('sex',sex)
100.    elif key=='3':
101.        age=input('Input age of student:')
102.        self.selectStu('age',age)
103.    else:
104.        print("Input error !")
```

Figure 5-50 shows the information of all students. We have used this function many times before. Lines 76-104 shows the function of selecting student information. We can select id, name, sex and age to select the information of students. Figure 5-51 shows the specific operation of selecting student information. We select the name to query the student information. Input the name of the student, we can find all the information of the student.

```
Please input the number corresponding to the function: 5
  0.Id
  1.Name
  2.Sex
  3.Age
Select method: 1
Input name of student:Lili
1 Lili female 18
```

Figure 5-51 Select student information

```
105. #6. Count the number of students of each age
106. def ageStatistics(self):
107.    Age=[]
108.    for i in range(len(self.stuInfo)):
109.        Age.append(self.stuInfo[i]['age'])
110.    print(Age)
111.    plt.xlabel("Age")
112.    plt.ylabel("Number")
113.    plt.hist(Age)
114.    plt.title "Count the number of students of each age")
115.    plt.show()
```

Lines 105-115 are the functions of student age statistics. As shown in Figure 5-52, there are four students in system, of which there are at most two aged 24.

```
116. stuInfo = [{'name':'Mike','sex':'male','age':'23'},
117.           {'name':'Lili','sex':'female','age':'24'},
118.          {'name':'Jame','sex':'male','age':'22'},
119.          {'name':'Smith','sex':'female','age':'24'},]
120. S=Student(stuInfo)
121. S.Mymain()
```

Lines 116-121 is an example of the student management system. When the student class is initialized, the information of four students is stored in advance.

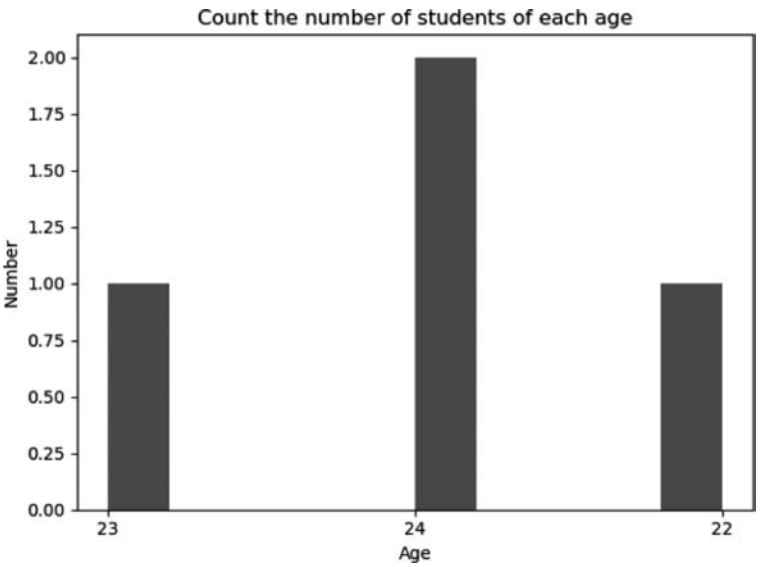

Figure 5-52 Count the number of students of each age with Matplotlib

5.4.3 Self test practice

1. Practice 1: Olympic rings

(1) **Title Description**: This experiment requires drawing the five Olympic rings. Olympic rings was conceived and designed by Pierre de Coubertin in 1913. It was determined by the Olympic Charter. It is also known as the Olympic rings. It is the most widely recognized symbol of the Olympic Games in the world. It consists of five Olympic rings, which are blue, yellow, black, green and red. The rings are connected from left to right, with blue, black and red rings on the top and yellow and green rings on the bottom. The whole shape is a small regular trapezoid at the bottom.

(2) **Tips**:

- First, we need to use turtle. goto function to move the brush to the specified position.
- Next, use the turtle. circle function to draw a circle. This function needs to give the radius of the circle. At the same time, we need to give the thickness of the brush.

(3) **Input**: None.

(4) **Output**: as shown in Figure 5-53.

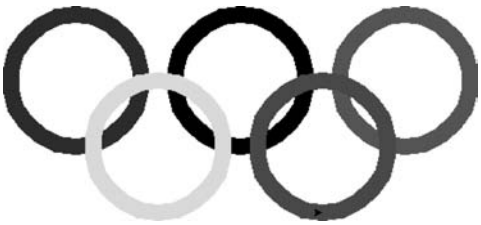

Figure 5-53 Olympic rings

2. Practice 2: Comparison of weather in different cities

(1) **Title Description**: Table 5-2 shows the average weather of Beijing, Shanghai and Guangzhou in 12 months. The standard temperature is centigrade. We will use Matplotlib package to draw a line chart to observe the changes of weather and temperature and the comparison between three different cities.

Table 5-2 Monthly weather in Beijing, Shanghai and Guangzhou

	Beijing	Shanghai	Guangzhou
January	-2℃	8℃	20℃
February	2℃	15℃	21℃
March	8℃	18℃	23℃
April	16℃	20℃	26℃
May	22℃	25℃	28℃
June	25℃	28℃	29℃
July	29℃	31℃	32℃
August	27℃	30℃	28℃
September	18℃	25℃	27℃
October	9℃	16℃	25℃
November	1℃	11℃	22℃
December	-5℃	5℃	20℃

(2) **Tips**:

① First, we use three lists to store the monthly temperature of three cities.

② Next, we can use the plot function to draw a line graph.

(3) **Input**: Monthly temperature in Beijing, Shanghai and Guangzhou.

(4) **Output**: as shown in Figure 5-54.

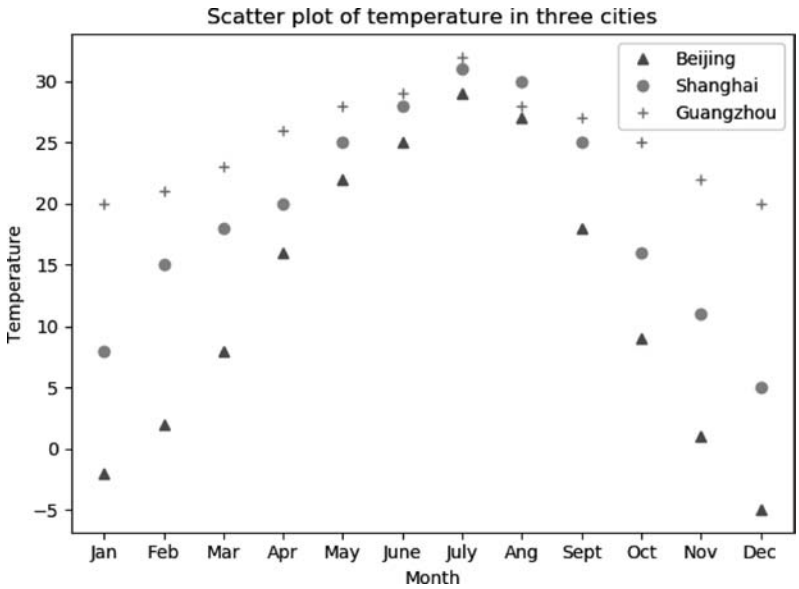

Figure 5-54 Scatter plot of temperature in three cities