程式設計概論
Programming 101
– function (函式)

授課老師：邱淑怡

Outline

- Function: how to define our function
- Global variables and local variables
- How to define and use main() function
- 補充資料
  - Recursive function
  - lambda function

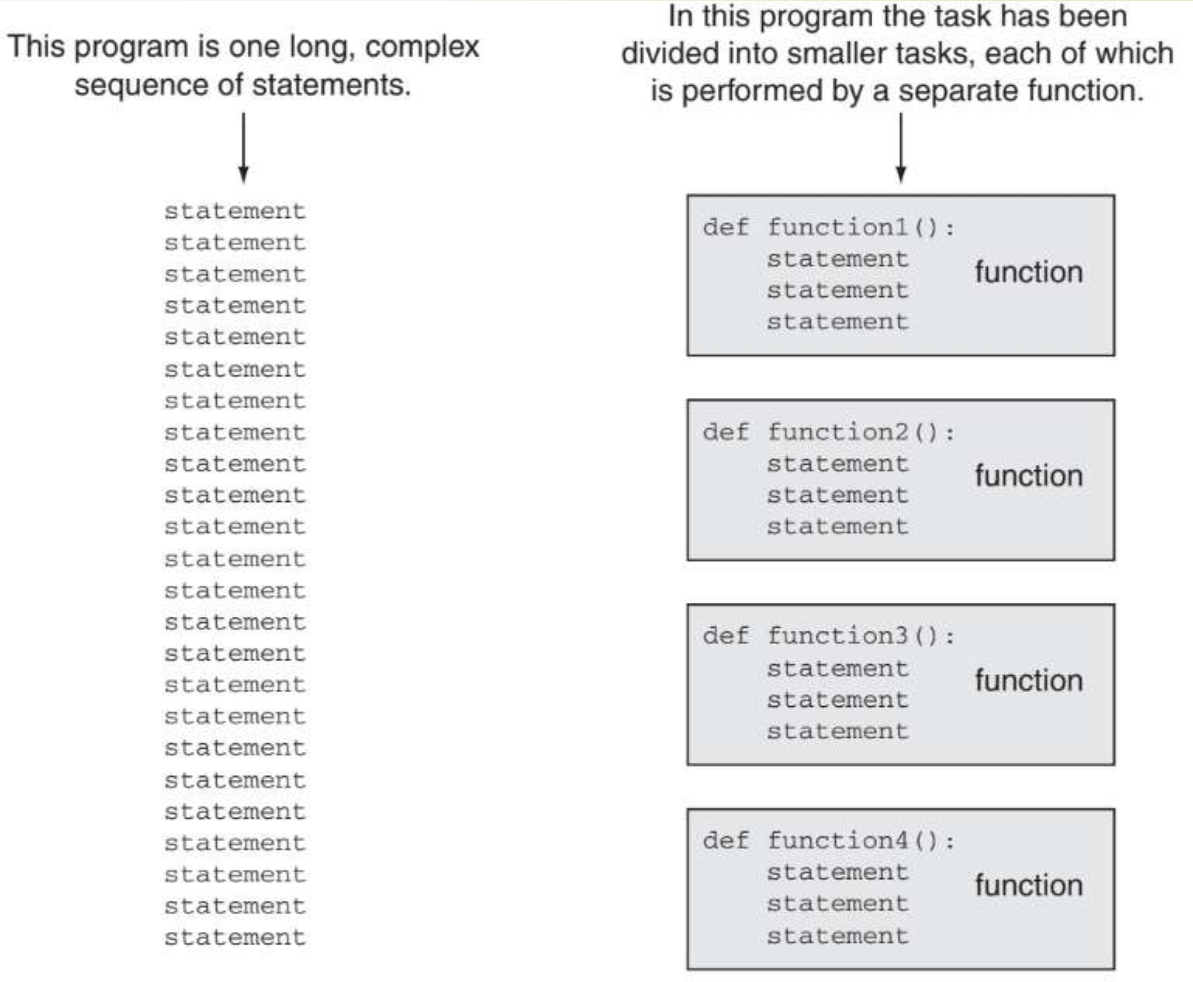
Function

- function is a small block in a program. It's like a machine that can decide its usage and its input and output by yourself.
- Example: A vending machine is like a function. The input is coins and item selection ; the output is the item you selected.

Benefits of modularizing a program with function

1. Reusability
2. Legibility (易讀性)
3. Easy for debugging
4. Consistency
5. Modularize program: modularize is a concept while programming.

Modularize program



Functions (自訂函數)

- A simple function
- Value-returning function

How to define our function?

- Two steps:
  1. Define the function
  2. Call this function

```
# define function
def function_name(x):
        statements
          ….

# call function
function_name(x1)
```

Practice

Situation 1 :

1. Write a function(CtoF1). It can convert °C to °F (F=C*1.8+32), print °F
2. Call CtoF1 function deliver °C as a parameter and print °F in this function

Situation 2 :

1. Write a function(CtoF2) It can convert °C to °F (F=C*1.8+32)
2. Call CtoF2 function deliver °C as a parameter, then °F as the return value.
3. print °F

Define our function

Define function

```
def mypowfun(x, y):
    r=1
    while y>0:
        r*=x    #r=r*x
        y-=1    #y=y-1
    return r
```

x,y are parameters (參數)

r is return value

Call function

```
Res= mypowfun(3,5)
```

Function example: default value

Define function

```
def power(x, y=2):
    r=1
    while y>0:
        r=r*x
        y=y-1
    return r
```

put the parameter without default value in front

Call function

```
Res= power(3,5)
```

=

```
Res= power(y=5,x=3)
```

Res =power(3)

parameters with default value

```
def function_name(param1, param2=value2, param3=value3, ...):
```

cannot write that

```
def function_name(param1=value1, param2, param3):
```

Step1

```
# define function
def greet(name="John", message='Hi'):
        return "%s: %s!"%(name,message)
```

Step2

```
# call function
greeting = greet('Hello')
print(greeting)
greeting = greet(message='Hello')
print(greeting)
greeting = greet()
print(greeting)
```

Practice

- Pass the list as a parameter to the function product_msg(), and then list all members' product launch letters

```
def product_msg(users):
        str1="Dear"
        str2="Our company will hold a product launch event in Taipei on 2022/12/30"
        str3="Sincerely, General Manager"
        for person in users:
                msg=....
                print(msg)
members=["小明", "小花", "小白"]
product_msg(members)
```

How to define main() and call main()

Calculating the sale price of an item seems like it would be a simple task

- Design a program that calculates the sale price of an item in a retail business.
  1. To do that, the program(function) would need to get the item's regular price from the user.
     - DISCOUNT_PERCENTAGE = 0.20 #set constant
     - def get_regular_price()
  2. Then, you calculate the discount and subtract it from the regular price.
     - def discount(price)
     - def main() and call main()

program

```
DISCOUNT_PERCENTAGE = 0.20
# The main function.
def main():
  reg_price = get_regular_price()
  sale_price = reg_price - discount(reg_price)
  print('The sale price is NT',"%.2f"%(sale_price), sep='')

def get_regular_price():
  price = float(input("Enter the item's regular price: "))
  return price

def discount(price):
  return price * DISCOUNT_PERCENTAGE
main()
```

Example

1.Write a program that lets the user perform arithmetic operations on two numbers.

Your program must be menu driven, allowing the user to select the operation (+, -, *, or /) and input the numbers.

2.Functions:

(1)Function showChoice: This function shows the options to the user and explains how to enter data.

(2)Function add: This function accepts two number as arguments and returns sum.

(3)Function subtract: This function accepts two number as arguments and returns their difference.

(4)Function multiply: This function accepts two number as arguments and returns product.

(5)Function divide: This function accepts two number as arguments and returns quotient.

https://www.cs.nccu.edu.tw/~sichiu/10902_ppt/menu.py

Global variable vs. local variable

What's the problem?

- Use function(myfun()function) to modify a global variable(x1)

```
x1=10
def myfun():
        x2=15
        x1=x2+10
        print(x1, x2)

myfun()
print(x1)
```

https://pythontutor.com/

Global variable vs. local variable

1. define global variables and local variables
2. Read global variables in the function
3. Modify global variables in the function

1. Global variable and local variable define

- local variable: in the function
- global variable: outside function

```
x1=10
def myfun():
        x2=20
        print(x1, x2)
print(x1,x2)
```

2.Read the value of global variable/local variable in the function

```
x1 =10    #global var.
def myfun():
        x2= x1 +5
        print(x1, x2)

myfun()
print(x1)
```

≠

```
x1 =10    #global var.
def myfun1():
        x1=20       #local var
        x2=x1+5
        print(x1, x2)

myfun1()
print(x1)
```

3. Modify the value of global variables in the function


```
x1=10
def fun():
    x1 = 35   # local var.
    print("local var:",x1)

fun()
print("global var",x1)
```

modify

```
x1=10
def fun1():
        global x1
        x1=35   #modify global var
        print(x1)

fun1()
Print(x1)
```

Exercise 5

Question 1

Define constant
STATE_RATE=
COUNTY_RATE=

- A retail company must file a monthly sales tax report listing the total sales for the month, and the amount of state and county sales tax collected.
- The state sales tax rate is 4 percent and the county sales tax rate is 2 percent. → state_tax() and county_tax()
- Write a program that asks the user to enter the total sales for the month. → get_sales()
- From this figure, the application should calculate and display the following:
  - The amount of county sales tax
  - The amount of state sales tax
  - The total sales tax (county plus state)

Define main()
Call main()

Question 2

- Hal owns a business named Make Your Own Music, which sells guitars, drums, banjos, synthesizers, and many other musical instruments.
- Hal's sales staff works strictly on commission. At the end of the month, each salesperson's commission is calculated according to this table (next page).
- To calculate a salesperson's monthly pay, the formula:
  - pay=sales X commission_rate – advanced_pay

Question 2---the table

Sales This Month	Commission Rate
Less than $10,000	10%
$10,000–14,999	12%
$15,000–17,999	14%
$18,000–21,999	16%
$22,000 or more	18%

- A salesperson with $16,000 in monthly sales will earn a 14 percent commission ($2,240).
- Another salesperson with $18,000 in monthly sales will earn a 16 percent commission ($2,880).

Question 2 (cont.)

The following general algorithm outlines the steps:

1. Get the salesperson's monthly sales. → get_sales()
2. Get the amount of advanced pay. → get_advanced_pay()
3. Use the amount of monthly sales to determine the commission rate. → determine_comm_rate()
4. Calculate the salesperson's pay using the formula previously shown. If the amount is negative, indicate that the salesperson must reimburse the company.

Define main()
Call main()

Review

Textbook: chapter 4 and chapter 6: 6.1 and 6.2

補充資料

lambda function

Function in another way:use **lambda** to define functions. When the expression is executed, a function object will be generated.

- lambda can be used for anonymous function(no name)
- lambda is not a name of function. Is an instruction.
- lambda syntax(one line only) :
  lambda arg1, arg2, ... : expression
  - arg1、arg2... in the lambda is arguments when defining function. You can use those argument in the expression.
  - lambda is an operational expression, not declarative sentence. It should still be operational expression after":". There are no blocks in lambda.
- You can use lambda function for some simple calculation, and the regular function can be used for complex logic.

lambda function example: find_max()

- Define lambda function
  - f = lambda num1, num2: num1 if num1 > num2 else num2

Python code

- Call lambda function
  - f(5,9)
  - f(-5, -9)
  - f(num2=5, num1=-9)

lambda example:
make lambda function return list data, the argument had given default value

Q: Generate a list starting point 5 and ending point 10

Python code

```
start =5  #global variable
f = lambda stop = 10: [i for i in range(start, stop)]
    #build a lambda function, stop the default value for the argument
    #This lambda function will return a list data group
```

Call a lambda function using f :

```
f() # [5, 6, 7, 8, 9]
f(12) # [5, 6, 7, 8, 9, 10, 11]
```

lambda function characteristic

- lambda arg1, arg2: expression A if condition else expression B
  - lambda function has no function_name
  - lambda is not the function name, it's an instruction.
  - The lambda function is "dispose once used"
  - The most typical use of the lambda function is to use the two built-in functions filter() and map() to process data groups

lambda v.s. function

? In Python, it's able to use lambda to define a function

The function call λ function or Anonymous function

```
lambda a, b : a if a>b else b
```

lambda expression can't cont contain blocks

regular function

```
def max1(a, b):
    return a if a>b else
b
```

Recursive function

recursive function

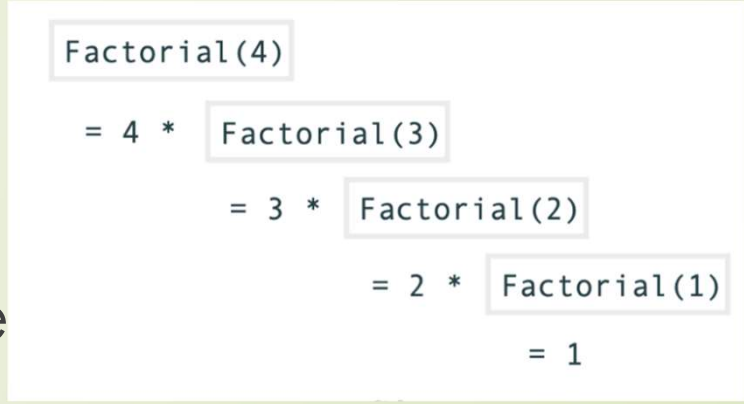
- When solving problems, we often encounter problems that are difficult to solve using loops or if statements. For example, when walking a maze and encountering a dead end, we need to return to the previous calculation, or when solving the Tower of Hanoi problem. The operations are the same, but because the parameters to be operated are different each time, we need to write the program repeatedly.
- Basic concept of recursion
  - The way to solve a problem is to break it down, and then solve the small problems individually to get the answer. We call this "Divide and Conquer"
  - recursion is based on this concept
- When a function continues to call itself during execution, we consider the function to be recursive.
- To prevent the function from endlessly self-calling, we also need to decide a clear termination.

recursive function

- Two key points of recursive function
  - The way recursive function call itself
  - Conditions to end the call
- Recursive functions can usually be replaced by for or while loops, but recursive functions are more logical, readable and flexible than loops.

recursive function example: factorial(4!)

- when n=0，F(n)=n!=0!=1
- when n>0，F(n)=n!=n*(n-1)!=n*F(n-1
- when n<0，F(n)=-1 (can't calculate



define recursive function

```
def factorial(n):
        if n==0 or n==1:
                return 1
        else:
                return n*
factorial(n-1)
```

Call function

```
print("0!=", factorial(0))
print("4!=", factorial(4))
```

function example: factorial

define function

```
def factorial_loop (n):
        factor = 1
        for I in range(1,n+1):
                factor *= I
        return factor
```

Call function

```
print("4!=", factorial_loop(4))
```