Python

一種廣泛使用的直譯式、進階程式、通用型程式語言，強調代碼的可讀性和簡潔的語法，尤其是使用空格縮排劃分代碼塊，支援多種程式範式，包括物件導向、結構化、指令式、函式式和反射式程式，擁有動態型別系統和垃圾回收功能，能夠自動管理記憶體使用，並且其本身擁有一個巨大而廣泛的標準庫

執行直譯器interpreter = Python來執行檔案hello.py

python hello.py

逐行執行

python <hello.py>

格式化字符串

name = input("Name: ")

print("Hello, " + name)

等於

#{}中不能有\n，SyntaxError: f-string expression part cannot include a backslash

print(f"Hello, {input("Name: ")}")

數據類型Data Type

bool True或False

float 實數

int 整數

str 字串

range 通常用在迴圈中的數字序列

list 可變值的序列

tuple 不可變值序列

set 無序集合

dict 鍵key和值value對的關聯陣列，鍵必須是可雜湊的類型鍵

變量Variable

在Python中不用宣告數據類型Data Type，只能用初始化宣告變量Variable

( in C)

int x = 54;

string word = “this is cs50”;

( in Python)

不用結尾放分號

x = 54

支持單或雙引號的字串

word = “this is cs50”或word = ‘this is cs50’

條件語句Conditional Statements

IF語法

( in C)

if (x < 10 || y == 10)

{

//code

}

else if (x == 10 && y == 10)

{

//code

}

else if (y != 10)

{

//code

}

else

{

//code

}

( in Python)

if x < 10 or y == 10:

#code

elif x == 10 and y == 10:

#code

elif not y == 10:

#code

else:

#code

?:三元運算符

( in C)

char letter = get\_char();

bool x = isalpha(letter) ? true : false;

( in Python)

x = True if input().isalpha() else False

循環Loops

WHILE語法

( in C)

int counter = 0;

while (counter < 100)

{

printf(“%i\n”, counter);

counter++;

}

( in Python)

counter = 0

while counter < 100:

#自動換行，不用\n

print(counter)

counter += 1

FOR語法

( in C)

for (int x = 0; x < 100; x ++) //或for (int x = 0; x < 100; x += 2)

{

printf(“%i\n”, x);

}

( in Python)

for x in range(100): #或for x in range(0,100,2):

print(x)

for不侷限於數字

names = ["Harry", "Ron", "Hermione"]

for name in names:

print(name)

#Harry

#Ron

#Hermione

列表List

可變大小，可以包含混合的類型，可以添加或刪除

4個元素1,2,3,4的列表num

num = [1, 2, 3, 4]

將5插入num最末位元素

num.append(5)

將5插入num第4位元素

num.insert(4, 5)

將list 中元素5,6縫合到num之後，num成為6個元素1,2,3,4,5,6的列表

num[len(num):] = [5, 6]

500個元素的空白列表num

num = [x for x in range(500)]

空白列表num

num = list()

元組tuple

不可變大小，可以包含混合的類型，不能添加或刪除，通常在需要將多個值存儲在一起的情況下使用

一個由元組組成的列表

war = [(‘ww1', 1918), ('ww2', 1945)]

集合set

不包含重複項，可以包含混合的類型，如果可雜湊

s = set()

s.add(1)

s.add(2)

s.add(3)

s.add(1)

s.remove(2)

print(s) #{1, 3, 4}

print(f"The set has {len(s)} elements.") #The set has 2 elements.

字典dict

可變大小

pizza = {

‘cheese’ : 10,

‘tomato’ : 12,

‘chicken’ : 15

}

for pie, price in pizza.items():

print(‘{} pizza costs {}’.format(pie, price)) #或print(pie + ’pizza costs’ + str(price))

#cheese pizza costs 10

#tomato pizza costs 12

#chicken pizza costs 15

函式Function

避免主程式運行函式尚未宣告

def main():

cough(3)

def cough(n):

for i in range(n):

print("cough")

main()

或

cough(3)

def cough(n):

for i in range(n):

print("cough")

if \_\_name\_\_ == “\_\_main\_\_”:

main()

在一個文件functions.py中編寫函式並在另一個文件square.py中運行它們

在functions.py中

def square(x):

return x \* x

在square.py中

#導入functions中的square

from functions import square

for i in range(10):

print(f"The square of {i} is {square(i)}")或

#導入整個functions

import functions

for i in range(10):

print(f"The square of {i} is {functions.square(i)}")

類Class

#類Person

class Person:

#使用\_\_init \_\_()函式將值分配給創建對象的屬性或方法

# self來表示當前正在使用的物件Object，應為Python類中方法的第一個參數

def \_\_init\_\_(self, fname, lname):

#屬性Attribute = firstname、lastname

self.firstname = fname

self.lastname = lname

#方法Method = printname

def printname(self):

print(self.firstname, self.lastname)

物件Object

物件x

x = Person("John", "Doe")

x.printname() # John Doe

繼承Inheritance

# class定義不能為空，若不包含任何內容的定義，請放入pass語句

#子類Student從該父類Person繼承屬性和方法

class Student(Person):

pass

#子類\_\_init\_\_() 函式將覆蓋父類 \_\_init\_\_()函式的繼承

class Student(Person):

def \_\_init\_\_(self, fname, lname):

#若要保留父\_\_init\_\_() 函式的繼承，請添加對父函式的調用\_\_init\_\_()

class Student(Person):

def \_\_init\_\_(self, fname, lname):

Person.\_\_init\_\_(self, fname, lname)

或

#super()函式可以使子類從其父類繼承所有方法和屬性

class Student(Person):

def \_\_init\_\_(self, fname, lname):

super().\_\_init\_\_(fname, lname)

在子類Student增加屬性graduationyear

class Student(Person):

def \_\_init\_\_(self, fname, lname, year):

super().\_\_init\_\_(fname, lname)

self.graduationyear = year

x = Student("Mike", "Olsen", 2019)

修飾模式Decorators

修飾模式相比生成子類更為靈活，這樣可以給某個對象而不是整個類添加一些功能，類繼承在編譯時候增加行為，而裝飾模式是在運行時增加行為

def announce(f):

def wrapper():

print("About to run the function")

f()

print("Done with the function")

return wrapper

@announce

def hello():

print("Hello, world!")

hello()

#About to run the function

#Hello, world!

#Done with the function

匿名函式Lambda Functions

無需定義識別碼（函式名）的函式或子程式

def f(x):

return x \* x

等同

lambda x: x \* x

people = [

{"name": "Harry", "house": "Gryffindor"},

{"name": "Cho", "house": "Ravenclaw"},

{"name": "Draco", "house": "Slytherin"}

]

people.sort(key=lambda person: person["name"])

print(people)

# [{'name': 'Cho', 'house': 'Ravenclaw'}, {'name': 'Draco', 'house': 'Slytherin'}, {'name': 'Harry', 'house': 'Gryffindor'}]

Try 和Except

try可以測試代碼是否存在錯誤，except可以處理錯誤

import sys

try:

x = int(input("x: "))

y = int(input("y: "))

except ValueError:

print("Error: Invalid input")

sys.exit(1)

try:

result = x / y

except ZeroDivisionError:

print("Error: Cannot divide by 0.")

# Exit the program

sys.exit(1)

print(f"{x} / {y} = {result}")

Python 文件處理

open()函數

有文件名和模式兩個參數

有四種打開文件的不同方法（模式）：

"r" 讀取-默認值，打開文件進行讀取，如果文件不存在則報錯

"a" 追加-打開供追加的文件，如果不存在則創建該文件

"w" 寫入-打開文件進行寫入，如果文件不存在則創建該文件

"x" 創建-創建指定的文件，如果文件存在則返回錯誤

f = open("demofile.txt")

可以指定文件是應該作為二進制還是文本模式進行處理

"t" 文本-默認值

"b" 二進制，例如圖像

f = open("demofile.txt", "rt")

#read()函數用於讀取文件的內容

print(f.read())

#close()函數用於關閉文件

f.close()

正規表示式Regular Expression

| 或

+ 前面的字元可以出現1次或多次

goo+gle可以符合google、gooogle、goooogle

? 前面的字元可以出現0次或1次

colou?r可以符合color、colour

\* 前面的字元可以出現0次或1次或多次

0\*42可以符合42、042、0042

() 定義運算子的範圍和優先度

gr(a|e)y等於gray|grey

(grand)?father符合father和grandfather

^ 符合輸入字串的開始位置

$ 符合輸入字串的結束位置

. 符合除了\r和\n之外的任何單個字符

.\* 用於0個或更多字符

.+ 用於1個或更多字符