

The background is a gradient from red at the top to blue at the bottom, with a starry space pattern. Overlaid on the left are several concentric circles and arcs, some with degree markings (40, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260) and arrows, suggesting a circular or rotational theme.

多媒體數據分析 與應用

張家瑋 博士

助理教授

國立臺中科技大學資訊工程系

BASIC PYTHON3

基礎PYTHON3

Code Structure

匯入函示庫



```
import library
import library1 as lib1
from library import sub-library as sublib
```

```
print('Hello World')
```

四個空白

```
for i in range(10):
    print('Hi!')    #印出十次 'Hi!'
```

```
def sayhi():
    print('Hi')
```

```
sayhi()            #呼叫 function sayhi(), 印出一次
                    'Hi'
```

1

變數(Variables)

數字 : int, float, long, complex
字串 : string

常見的數值運算

(int, float, long, complex)

```
>>> 1+1
2
>>> 1-1
0
>>> 2*3
6
>>> 2**3
8
>>> 100/3
33.333333333333336
>>> 100//3 #求整數部份，無條件捨去
33
>>> 100%3 #求餘數
1
```

常見的字串運算與處理

```
>>> a = "Hello!"
>>> b = "World!"
>>> a+b
'Hello!World!'
>>> a*2+b
'Hello!Hello!World!'
>>> len(a) #字串長度
6
```

```
>>> s = "abcdefghij"
>>> s[3:5]
'de'
>>> s[:5]
'abcde'
>>> s[5:]
'fghij'
>>> s[::2]
'acegi'
>>> s[:]
'abcdefghij'
```

```
>>> s = "abcdefghij"
>>> s[:-5]
'abcde'
>>> s[-5:]
'fghij'
>>> s[::-2]
'jhfdb'
```


2

容器(Containers)

- List
- Array
- Dictionary

列表 (List)

```
>>> a = [1, 2, 3]
>>> b = [4, 5]
>>> a.append(b)
>>> a
[1, 2, 3, [4, 5]]
```

```
-----
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5]
```


列表 (List)

```
>>> numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> numbers[3:5]
[3, 4]
>>> numbers[5:]
[5, 6, 7, 8, 9]
>>> numbers[:5]
[0, 1, 2, 3, 4]
>>> numbers[::2]
[0, 2, 4, 6, 8]
>>> numbers[:]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

矩陣 (Array)

```
>>> l = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> import numpy as np
>>> a = np.asarray(l)
[0 1 2 3 4 5 6 7 8 9]
```

```
-----
>>> import numpy as np
>>> a = np.arange(10)
[0 1 2 3 4 5 6 7 8 9]
>>> a[0]
0
>>> a[9]
9
```


矩陣 (Array)

因為Python沒有 array 型態，
必須透過上一頁的方法取得
numbers = [0 1 2 3 4 5 6 7 8 9]

```
>>> numbers[3:5]  
[3 4]
```

```
>>> numbers[5:]  
[5 6 7 8 9]
```

```
>>> numbers[:5]  
[0 1 2 3 4]
```

```
>>> numbers[::2]  
[0 2 4 6 8]
```

```
>>> numbers[:]  
[0 1 2 3 4 5 6 7 8 9]
```

字典 (Dictionary)

```
>>> dictionary = { 1: 'one' , 2: 'two' , 3: 'three' }
```

```
>>> square = {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

```
>>> square[5]
```

```
25
```

```
>>> square.keys()
```

```
dict_keys([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
>>> square.values()
```

```
dict_values([0, 1, 4, 9, 16, 25, 36, 49, 64, 81])
```


3

迴圈與條件式

- for loop
- if...else...

For 迴圈

```
>>> numbers = []  
>>> for i in range(10):  
    numbers.append(i)
```

```
>>> numbers  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

#簡寫

```
[i for i in range(10)]  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> square = {number: number**2 for number in  
range(10)}  
>>> square  
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```


進階用法

在 Python 内置了工厂函数，`range` 函数将会返回一个序列，总共有三种使用方法

1 `range(start, stop)`

其中 `start` 将会是序列的起始值，`stop` 为结束值，但是**不包括**该值，类似 数学中的表达 `[start, stop)`，左边为闭区间，右边为开区间。

```
for i in range(1, 10):  
    print(i)
```

上述表达将会返回 `1-9` 所有整数，但不包含 `10`

2 `range(stop)`

如果省略了 `start` 那么将从 `0` 开始，相当于 `range(0, stop)`

3 `range(start, stop, step)`

`step` 代表的为步长，即相隔的两个值得差值。从 `start` 开始，依次增加 `step` 的值，直至**等于或者大于** `stop`

```
for i in range(0, 13, 5):  
    print(i)
```

将会输出 `0, 5, 10`。

進階用法

Python 共内置了 `list`、`tuple`、`dict` 和 `set` 四种基本集合，每个集合对象都能够迭代。

tuple 类型

```
tup = ('python', 2.7, 64)
for i in tup:
    print(i)
```

程序将以此按行输出 'python'、2.7 和 64。

dictionary 类型

```
dic = {}
dic['lan'] = 'python'
dic['version'] = 2.7
dic['platform'] = 64
for key in dic:
    print(key, dic[key])
```

输出的结果为：`platform 64`、`lan python`、`version 2.7`，字典在迭代的过程中将 `key` 作为可迭代的对象返回。注意字典中 `key` 是乱序的，也就是说和插入的顺序是不一致的。如果想要使用顺序一致的字典，请使用 `collections` 模块中的 `OrderedDict` 对象。

set 类型

```
s = set(['python', 'python2', 'python3', 'python'])
for item in s:
    print(item)
```

将会输出 `python`、`python3`、`python2` set 集合将会去除重复项，注意输出的结果也不是按照输入的顺序。

進階用法

Python 共内置了 `list`、`tuple`、`dict` 和 `set` 四种基本集合，每个集合对象都能够迭代。

tuple 类型

```
tup = ('python', 2.7, 64)
for i in tup:
    print(i)
```

程序将以此按行输出 'python'、2.7 和 64。

dictionary 类型

```
dic = {}
dic['lan'] = 'python'
dic['version'] = 2.7
dic['platform'] = 64
for key in dic:
    print(key, dic[key])
```

输出的结果为：`platform 64`、`lan python`、`version 2.7`，字典在迭代的过程中将 `key` 作为可迭代的对象返回。注意字典中 `key` 是乱序的，也就是说和插入的顺序是不一致的。如果想要使用顺序一致的字典，请使用 `collections` 模块中的 `OrderedDict` 对象。

set 类型

```
s = set(['python', 'python2', 'python3', 'python'])
for item in s:
    print(item)
```

将会输出 `python`、`python3`、`python2` set 集合将会去除重复项，注意输出的结果也不是按照输入的顺序。

if...else...

```
>>> numbers = []  
>>> for i in range(10):  
    numbers.append(i)
```

```
>>> numbers  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

#簡寫

```
numbers = [i for i in range(10)]  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
-----  
for i in range(len(numbers)):  
    if( i == 0 ):  
        print(i, '是奇數也是偶數')  
    elif( i % 2 == 1 ):  
        print( i, '奇數' )  
    else:  
        print( i, '偶數' )
```


4

函式與匿名函式

- function
- lambda

function

```
>>>def add(x, y):  
>>>    return x+y
```

```
>>>add(1, 1)  
2  
>>>add(1, -3)  
-2
```

```
>>>def minus(x, y):  
>>>    return x-y
```

```
>>>minus(1, 1)  
0  
>>>minus(1, -3)  
4
```


lambda

```
>>> add = lambda x, y: x + y  
>>> add(1, -3)  
-2
```

```
>>> newValue = lambda x: -x  
>>> newValue(9)  
-9
```

PRINT

印出

print 字符串 ¶

python 中 print 字符串 要加"或者""

```
>>> print('hello world')  
"  
hello world  
"  
>>> print("hello world 2")  
"  
hello world 2  
"
```

字符串相加

print 字符串叠加

可以使用 `+` 将两个字符串链接起来, 如以下代码.

```
>>> print('Hello world'+'Hello Hong Kong')  
Hello world Hello Hong Kong
```


基本運算

可以直接 `print` 加法 `+`, 减法 `-`, 乘法 `*`, 除法 `/`. 注意: 字符串不可以直接和数字相加, 否则出现错误。

```
>>> print(1+1)
2
>>> print(3-1)
2
>>> print(3*4)
12
>>> print(12/4)
3.0
>>> print('iphone'+4) #字符串不可以直接和数字相加
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    print('iphone'+4)
TypeError: Can't convert 'int' object to str implicitly
```

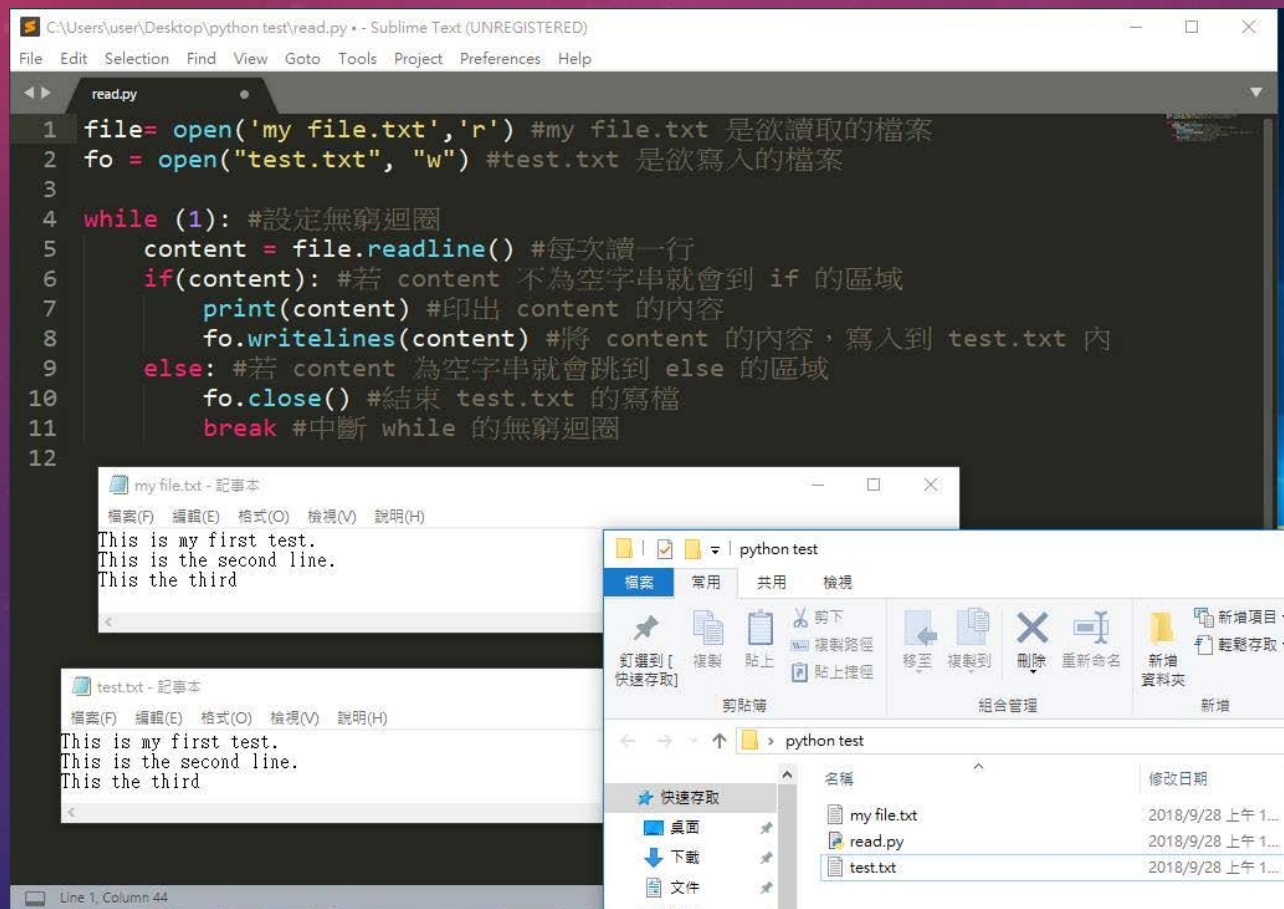
基本運算

`int()` 和 `float()`: 当 `int()` 一个浮点型数时, `int` 会保留整数部分, 比如 `int(1.9)`, 会输出 `1`, 而不是四舍五入。

```
>>> print(int('2')+3) #int为定义整数型
.....
5
.....
>>> print(int(1.9)) #当int一个浮点型数时, int会保留整数部分
.....
1
.....
>>> print(float('1.2')+3) #float()是浮点型, 可以把字符串转换成小数
.....
4.2
.....
```


寫檔讀檔

寫檔讀檔



寫檔讀檔

- `file = open("my file.txt", "r", encoding='utf-8')`
- `fo = open("test.txt", "w", encoding='utf-8')`

The background features a vertical color gradient from deep blue at the bottom to a vibrant red at the top. Overlaid on this are several faint, white, semi-transparent circular and arc-like patterns. Some of these patterns include small arrows indicating a clockwise direction. A prominent circular scale with numerical markings (40, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260) is visible on the left side, curving upwards. The overall aesthetic is modern and technical.

THANK YOU