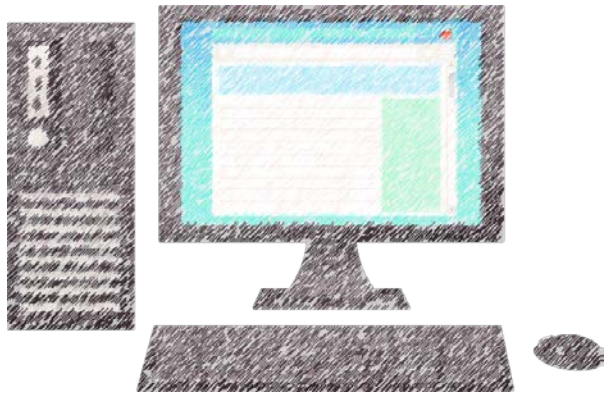
# 桃園市政府勞動局 112 年勞工學苑產業應用班
# Python 程式設計：從零基礎入門到進階

## 第 5 單元

# 字串



林柏江老師
元智大學 電機工程學系 助理教授
pclin@saturn.yzu.edu.tw

# 預計課程進度

| 週次 | 日期 | 上午課程內容 (09:00 ~ 12:00) | 下午課程內容 (13:00 ~ 16:00) |
|---|---|---|---|
| 1 | 2023/07/23 | 01. 運算思維簡介 | 02. Python 快速上手 |
| 2 | 2023/07/30 | 03. Python 基礎 | 04. Python 基本資料結構 |
| 3 | 2023/08/06 | 05. 字串 | 06. 字典 |
| 4 | 2023/08/13 | 07. 流程控制 | 08. 函式 |
| 5 | 2023/08/20 | 09. 模組與作用域 | 10. Python 程式檔 |
| 6 | 2023/08/27 | 11. 檔案系統的使用與檔案讀寫 | 12. 例外處理 |
| 7 | 2023/09/03 | 13. 類別與物件導向程式設計 | 14.初探資料結構與演算法 |
| 8 | 2023/09/10 | 15. 陣列 | 16. 鏈結串列 |
| 9 | 2023/09/17 | 17. 堆疊與佇列 | 18. 圖形結構 |
|  | 2023/09/24, 2023/10/01, 2023/10/08 (共三周) 放假 |  |  |
| 10 | 2023/10/15 | 19. 樹狀結構 | 20. 分治法 |
| 11 | 2023/10/22 | 21. 動態規劃 | 22. 貪婪演算法 |
| 12 | 2023/10/29 | 23. 回溯 | 24. 分支定界法 |

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# 字串為字元序列

- list 與 tuple 中的元素是依照順序排列的 (有序的)。
- 像這種有序的資料結構，在 Python 中稱為序列 (sequence) 型別。
- 字串是由字元依照順序排列而成的。
- 在 Python 中，字串也是一種序列型別。
- Python 序列型別都可以使用索引 `[n]` 來指定元素位置，或使用 `[m:n]` 來切片。
- 我們可以使用索引或切片從字串中取得字元或新字串。
- 字串是不可變的物件，所以從字串中取出或修改的內容，都會存放在新字串中。

# 字串的索引與切片範例

```
>>> x = "Hello"
>>> x[0]
'H'
>>> x[-1]
'o'
>>> x[1:]
'ello'
```

# 把換行符號從字串尾端刪除

```
>>> x = "Goodbye\n"
>>> x = x[:-1]
>>> x
'Goodbye'
```

# 使用 len() 計算字串中的字元數量

```
>>> len("Goodbye")
7
```

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 `%` 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# 基本的字串操作

使用字串連接算符 +：

```
>>> x = "Hello " + "World"
>>> x
'Hello World'
```

多個字串會自動連接在一起，字串之間可以有零個或一個以上空格：

```
>>> x = "Hello "   "World"
>>> x
'Hello World'
```

使用乘法算符 *，可用來建立新字串並初始化：

```
>>> 8 * "x"
'xxxxxxxx'
```

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# 轉義字元

轉義字元 (也稱作跳脫序列，Escape Sequence) 是以反斜線 \ 開頭的特殊字元。

| 轉義字元 | 代表字元 |
| --- | --- |
| \' | 單引號 (') |
| \" | 雙引號 (") |
| \\ | 反斜線 (\) |
| \a | 發出嗶聲 (Bell) |
| \b | 倒退鍵 (Backspace) |
| \f | 換頁 |
| \n | 換行 |
| \r | 歸位符號 (Carriage Return) |
| \t | 定位符號 (Tab) |
| \v | 垂直定位符號 (Vertical Tab) |

# ASCII 字元的八進位與十六進位轉義字元

- 可使用八進位或十六進位數值形式的轉義字元來表示任何 ASCII 字元：
- 八進位轉義字元是以反斜線 \ 後面跟著三個數字來定義：\nnn，其中 nnn 是一個八進位的數字。
- 十六進位轉義字元是以 \x 後面跟著兩個數字來定義：\xnn，其中 nn 是一個十六進位的數字。

```
>>> 'm'
'm'
>>> '\155'
'm'
>>> '\x6D'
'm'
>>> '\x6d'
'm'
```

```
>>> '\n'
'\n'
>>> '\012'
'\n'
>>> '\x0A'
'\n'
```

# ASCII TABLE

| Decimal | Hexadecimal | Binary | Octal | Char |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | *[NULL]* |
| 1 | 1 | 1 | 1 | *[START OF HEADING]* |
| 2 | 2 | 10 | 2 | *[START OF TEXT]* |
| 3 | 3 | 11 | 3 | *[END OF TEXT]* |
| 4 | 4 | 100 | 4 | *[END OF TRANSMISSION]* |
| 5 | 5 | 101 | 5 | *[ENQUIRY]* |
| 6 | 6 | 110 | 6 | *[ACKNOWLEDGE]* |
| 7 | 7 | 111 | 7 | *[BELL]* |
| 8 | 8 | 1000 | 10 | *[BACKSPACE]* |
| 9 | 9 | 1001 | 11 | *[HORIZONTAL TAB]* |
| 10 | A | 1010 | 12 | *[LINE FEED]* |
| 11 | B | 1011 | 13 | *[VERTICAL TAB]* |
| 12 | C | 1100 | 14 | *[FORM FEED]* |
| 13 | D | 1101 | 15 | *[CARRIAGE RETURN]* |
| 14 | E | 1110 | 16 | *[SHIFT OUT]* |
| 15 | F | 1111 | 17 | *[SHIFT IN]* |
| 16 | 10 | 10000 | 20 | *[DATA LINK ESCAPE]* |
| 17 | 11 | 10001 | 21 | *[DEVICE CONTROL 1]* |
| 18 | 12 | 10010 | 22 | *[DEVICE CONTROL 2]* |
| 19 | 13 | 10011 | 23 | *[DEVICE CONTROL 3]* |
| 20 | 14 | 10100 | 24 | *[DEVICE CONTROL 4]* |
| 21 | 15 | 10101 | 25 | *[NEGATIVE ACKNOWLEDGE]* |
| 22 | 16 | 10110 | 26 | *[SYNCHRONOUS IDLE]* |
| 23 | 17 | 10111 | 27 | *[ENG OF TRANS. BLOCK]* |
| 24 | 18 | 11000 | 30 | *[CANCEL]* |
| 25 | 19 | 11001 | 31 | *[END OF MEDIUM]* |
| 26 | 1A | 11010 | 32 | *[SUBSTITUTE]* |
| 27 | 1B | 11011 | 33 | *[ESCAPE]* |
| 28 | 1C | 11100 | 34 | *[FILE SEPARATOR]* |
| 29 | 1D | 11101 | 35 | *[GROUP SEPARATOR]* |
| 30 | 1E | 11110 | 36 | *[RECORD SEPARATOR]* |
| 31 | 1F | 11111 | 37 | *[UNIT SEPARATOR]* |
| 32 | 20 | 100000 | 40 | *[SPACE]* |
| 33 | 21 | 100001 | 41 | ! |
| 34 | 22 | 100010 | 42 | " |
| 35 | 23 | 100011 | 43 | # |
| 36 | 24 | 100100 | 44 | $ |
| 37 | 25 | 100101 | 45 | % |
| 38 | 26 | 100110 | 46 | & |
| 39 | 27 | 100111 | 47 | ' |
| 40 | 28 | 101000 | 50 | ( |
| 41 | 29 | 101001 | 51 | ) |
| 42 | 2A | 101010 | 52 | * |
| 43 | 2B | 101011 | 53 | + |
| 44 | 2C | 101100 | 54 | , |
| 45 | 2D | 101101 | 55 | - |
| 46 | 2E | 101110 | 56 | . |
| 47 | 2F | 101111 | 57 | / |
| 48 | 30 | 110000 | 60 | 0 |
| 49 | 31 | 110001 | 61 | 1 |
| 50 | 32 | 110010 | 62 | 2 |
| 51 | 33 | 110011 | 63 | 3 |
| 52 | 34 | 110100 | 64 | 4 |
| 53 | 35 | 110101 | 65 | 5 |
| 54 | 36 | 110110 | 66 | 6 |
| 55 | 37 | 110111 | 67 | 7 |
| 56 | 38 | 111000 | 70 | 8 |
| 57 | 39 | 111001 | 71 | 9 |
| 58 | 3A | 111010 | 72 | : |
| 59 | 3B | 111011 | 73 | ; |
| 60 | 3C | 111100 | 74 | < |
| 61 | 3D | 111101 | 75 | = |
| 62 | 3E | 111110 | 76 | > |
| 63 | 3F | 111111 | 77 | ? |
| 64 | 40 | 1000000 | 100 | @ |
| 65 | 41 | 1000001 | 101 | A |
| 66 | 42 | 1000010 | 102 | B |
| 67 | 43 | 1000011 | 103 | C |
| 68 | 44 | 1000100 | 104 | D |
| 69 | 45 | 1000101 | 105 | E |
| 70 | 46 | 1000110 | 106 | F |
| 71 | 47 | 1000111 | 107 | G |
| 72 | 48 | 1001000 | 110 | H |
| 73 | 49 | 1001001 | 111 | I |
| 74 | 4A | 1001010 | 112 | J |
| 75 | 4B | 1001011 | 113 | K |
| 76 | 4C | 1001100 | 114 | L |
| 77 | 4D | 1001101 | 115 | M |
| 78 | 4E | 1001110 | 116 | N |
| 79 | 4F | 1001111 | 117 | O |
| 80 | 50 | 1010000 | 120 | P |
| 81 | 51 | 1010001 | 121 | Q |
| 82 | 52 | 1010010 | 122 | R |
| 83 | 53 | 1010011 | 123 | S |
| 84 | 54 | 1010100 | 124 | T |
| 85 | 55 | 1010101 | 125 | U |
| 86 | 56 | 1010110 | 126 | V |
| 87 | 57 | 1010111 | 127 | W |
| 88 | 58 | 1011000 | 130 | X |
| 89 | 59 | 1011001 | 131 | Y |
| 90 | 5A | 1011010 | 132 | Z |
| 91 | 5B | 1011011 | 133 | [ |
| 92 | 5C | 1011100 | 134 | \ |
| 93 | 5D | 1011101 | 135 | ] |
| 94 | 5E | 1011110 | 136 | ^ |
| 95 | 5F | 1011111 | 137 | _ |
| 96 | 60 | 1100000 | 140 | ` |
| 97 | 61 | 1100001 | 141 | a |
| 98 | 62 | 1100010 | 142 | b |
| 99 | 63 | 1100011 | 143 | c |
| 100 | 64 | 1100100 | 144 | d |
| 101 | 65 | 1100101 | 145 | e |
| 102 | 66 | 1100110 | 146 | f |
| 103 | 67 | 1100111 | 147 | g |
| 104 | 68 | 1101000 | 150 | h |
| 105 | 69 | 1101001 | 151 | i |
| 106 | 6A | 1101010 | 152 | j |
| 107 | 6B | 1101011 | 153 | k |
| 108 | 6C | 1101100 | 154 | l |
| 109 | 6D | 1101101 | 155 | m |
| 110 | 6E | 1101110 | 156 | n |
| 111 | 6F | 1101111 | 157 | o |
| 112 | 70 | 1110000 | 160 | p |
| 113 | 71 | 1110001 | 161 | q |
| 114 | 72 | 1110010 | 162 | r |
| 115 | 73 | 1110011 | 163 | s |
| 116 | 74 | 1110100 | 164 | t |
| 117 | 75 | 1110101 | 165 | u |
| 118 | 76 | 1110110 | 166 | v |
| 119 | 77 | 1110111 | 167 | w |
| 120 | 78 | 1111000 | 170 | x |
| 121 | 79 | 1111001 | 171 | y |
| 122 | 7A | 1111010 | 172 | z |
| 123 | 7B | 1111011 | 173 | { |
| 124 | 7C | 1111100 | 174 | \| |
| 125 | 7D | 1111101 | 175 | } |
| 126 | 7E | 1111110 | 176 | ~ |
| 127 | 7F | 1111111 | 177 | *[DEL]* |

https://commons.wikimedia.org/wiki/File:ASCII-Table.svg

# 使用 print() 函式<br>輸出帶有轉義字元的字串

在 Python 互動模式下直接輸出此字串，以及使用 print() 函式輸出此字串，會有不同的結果。

```
>>> 'a\n\tb'
'a\n\tb'
>>> print('a\n\tb')
a
    b
```

字串以原形式呈現。

字串被 print() 函式輸出到終端機，會依照換行符號、定位符號的意義來進行換行與定位。

# 讓 print() 函式不添加換行符號

通常 print() 函式會自動在字串尾端添加換行符號。
若不想要 print() 函式自動添加換行符號，可把參數 end 設定為空字串。

```
>>> print("abc\n")
abc

>>> print("abc\n", end="")
abc
>>>
```

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# 字串常用的方法

- 大多數 Python 字串的方法都是內建的。
- 所有字串物件都可以直接使用這些方法。
  - 字串名稱`.method()`
- 字串是不可變的，所以字串的方法會回傳一個新的字串，而不是更改原本的字串。
- 有關字串處理的完整方法說明，可參閱 Python 文件：
  https://docs.python.org/3/library/stdtypes.html#string-methods

# 使用 join() 方法連接字串

使用一個字串，把其他字串連接起來，建立新字串。
以下範例使用 " " (一個空格) 把後面字串連接成新字串。

```
>>> " ".join(["join", "puts", "spaces", "between", "elements"])
'join puts spaces between elements'
```

以下範例使用 "::" 把後面字串連接成新字串。

```
>>> "::".join(["Separated", "with", "colons"])
'Separated::with::colons'
```

以下範例使用空字串 "" 把後面字串連接成新字串。

```
>>> "".join(["Separated", "by", "nothing"])
'Separatedbynothing'
```

# 使用 split()方法切割字串

split() 會把字串切割成字串 list。預設會以空白字元 (whitespace) 來切割，可透過參數自訂分割符號。
語法為：str.split(sep, maxsplit)
其中 sep 是分割符號，maxsplit 是分割的次數。

```
>>> x = "You\t\t can have tabs\t\n \t and newlines \n\n " \
"mixed in"
>>> x.split()
['You', 'can', 'have', 'tabs', 'and', 'newlines', 'mixed', 'in']
>>> x = "Mississippi"
>>> x.split("ss")
['Mi', 'i', 'ippi']
```

# 使用 split()方法切割字串 (續)

若指定要分割 n 次，split() 會產生一個有 n+1 個字串的 list。

```
>>> x = 'a b c d'
>>> x.split(' ', 1)
['a', 'b c d']
>>> x.split(' ', 2)
['a', 'b', 'c d']
>>> x.split(' ', 9)
['a', 'b', 'c', 'd']
```

```
>>> x = 'a\nb c d'
>>> x.split(' ', 2)
['a\nb', 'c', 'd']
>>> x.split(None, 2)
['a', 'b', 'c d']
```

# 使用 int() 和 float() 函式把字串轉換成數字

- int() 函式把字串轉換成整數。
- float() 函式把字串轉換成浮點數。
- 如果字串無法解釋為指定型別的數字，會引發 ValueError 例外。
- int() 函式的第二個參數是選用的 (optional)，可以用來指定要用二進位、十六進位等基底來解釋輸入字串。

```
>>> float('123.456')
123.456
>>> float('xxyy')
Traceback (innermost last):
 File "<stdin>", line 1, in ?
ValueError: could not convert string to float: 'xxyy'
>>> int('3333')
3333
>>> int('123.456')
Traceback (innermost last):
 File "<stdin>", line 1, in ?
ValueError: invalid literal for int() with base 10: '123.456'
>>> int('10000', 8)
4096
>>> int('101', 2)
5
>>> int('ff', 16)
255
>>> int('123456', 6)
Traceback (innermost last):
 File "<stdin>", line 1, in ?
ValueError: invalid literal for int() with base 6: '123456'
```

# 使用 strip()、lstrip()、rstrip() 移除多餘的空白

- strip() 會移除字串開頭或結尾處的任何空白。
- lstrip() 和 rstrip() 則是分別會移除字串左邊或右邊的空白。

```
>>> x = "  Hello,    World\t\t "
>>> x.strip()
'Hello,    World'
>>> x.lstrip()
'Hello,    World\t\t '
>>> x.rstrip()
'  Hello,    World'
```

# 空白字元

哪些字元屬於空白字元，可能會因作業系統而異。
以下指令可找出被 Python 視為空白的字元。

```
>>> import string
>>> string.whitespace
' \t\n\r\x0b\x0c'
>>> " \t\n\r\v\f"
' \t\n\r\x0b\x0c'
```

# 指定 strip()、lstrip()、rstrip() 要移除的字元

```
>>> x = "www.python.org"
>>> x.strip("w")
'.python.org'
>>> x.strip("gor")
'www.python.'
>>> x.strip(".gorw")
'python'
```

# 檢查字串是否屬於各種形式的方法

```
>>> x = "123"
>>> x.isdigit()
True
>>> x.isalpha()
False
>>> x = "MM"
>>> x.islower()
False
>>> x.isupper()
True
```

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# in 算符

```
>>> x = "The string"
>>> "str" in x
True
>>> "sTr" in x
False
>>> "e s" in x
True
```

# find() 方法

把要搜尋的文字當作參數。
若找到的話，會回傳搜尋文字第一個字元所在的索引位置。
若找不到，會回傳 -1。

```
>>> x = "Mississippi"
>>> x.find("ss")
2
>>> x.find("zz")
-1
```

# find(string, start, end) 方法

start 參數可指定要從字串的哪一個索引位置開始搜尋。
end 參數可指定要從字串的哪一個索引位置之前停止搜尋。

```
>>> x = "Mississippi"
>>> x.find("s")
2
>>> x.find("s", 2)
2
>>> x.find("s", 4)
5
>>> x.find("s", 4, 5)
-1
>>> x.find("ss", 3)
5
>>> x.find("ss", 0, 3)
-1
```

# rfind() 方法

rfind() 函式與 find() 函式幾乎相同，只有搜尋方向相反：
從字串的結尾朝向開頭方向進行搜尋。
若找到的話，會回傳搜尋文字最後出現的索引位置。
若找不到，會回傳 -1。

```
>>> x = "Mississippi"
>>> x.rfind("ss")
5
```

rfind() 函式也可使用 start 參數以及 end 參數，使用方式與 find() 函式相同。

# index() 和 rindex() 方法

index() 和 rindex() 方法分別與 rfind() 和 find() 方法相同，但是當 index() 或 rindex() 方法找不到文字時，不會回傳 -1，而是會引發 ValueError 例外。

# count() 方法

count() 方法可統計特定文字在字串中出現了幾次。

```
>>> x = "Mississippi"
>>> x.count("ss")
2
```

# startswith() 和 endswith() 方法

startswith() 和 endswith() 方法分別搜尋字串是否以特定文字開頭或結尾，依結果回傳 True 或 False。

```
>>> x = "Mississippi"
>>>
x.startswith("Miss")
True
>>>
x.startswith("Mist")
False
>>> x.endswith("pi")
True
>>> x.endswith("p")
False
```

# startswith() 和 endswith() 方法 (續)

startswith() 和 endswith() 方法可以一次搜尋多筆文字，依結果回傳 True 或 False。

```
>>> x = "Mississippi"
>>> x.endswith(("i", "u"))
True
```

檢查字串是否以 "i" 或 "u" 結尾

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# replace() 方法

replace() 方法可把字串中的某些文字 (第一個參數) 替換成新文字 (第二個參數)。

```
>>> x = "Mississippi"
>>> x.replace("ss", "+++")
'Mi+++i+++ippi'
```

# maketrans() 和 translate() 方法

maketrans() 方法搭配 translate() 方法，可以把字串內的字元轉換成不同的字元。

```
>>> x = "~x ^ (y % z)"
>>> table = x.maketrans("~^()", "!&[]")
>>> x.translate(table)
'!x & [y % z]'
```

# 字串修改的其他相關方法

- `lower()`：把字串中的所有字母轉換成小寫。
- `upper()`：把字串中的所有字母轉換成大寫。
- `capitalize()`：把字串中的第一個字母轉換成大寫。
- `title()`：把字串中所有單字的第一個字母轉換成大寫。
- `swapcase()`：把字串中的所有大寫字母轉換成小寫，所有小寫字母轉換成大寫。
- `expandtabs()`：以指定數目的空格替換字串中的定位符號。

# 以特定文字來填充字串的方法

- `ljust()`、`rjust()`、`center()`：用空格填充字串，以便向左、向右、以及置中對齊某個欄位寬度。
- `zfill()`：在數值字串的左邊補零，以達到指定的長度。

# 透過 list 來修改字串

字串是不可變的物件。
可先把字串轉換成字元 list，即可進行想要的操作，
然後在把結果轉換回字串。

```
>>> text = "Hello, World"
>>> wordList = list(text)
>>> wordList[6:] = []
>>> wordList.reverse()
>>> text = "".join(wordList)
>>> print(text)
,olleH
```

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# repr() 和 str() 把物件轉換成字串表示

- repr() 函式會把物件轉換成正式字串表示法 (formal string representation)。
  - 可從這個字串重建原始物件。
  - repr() 函式回傳的字串是給程式讀取的。
  - 適合程式除錯，可以使用 repr() 函式把變數內容印出來。
- str() 函式會把物件轉換成非正式字串表示法 (informal string representation)。
  - 無法從這個字串重建原始物件。
  - str() 函式回傳的字串是給人看的。

# 使用 repr() 把物件轉換成字串表示

```
>>> repr([1, 2, 3])
'[1, 2, 3]'
>>> x = [1]
>>> x.append(2)
>>> x.append([3, 4])
>>> 'the list x is ' + repr(x)
'the list x is [1, 2, [3, 4]]'
```

# 使用 str() 把物件轉換成字串表示

```
>>> from datetime import datetime
>>> now = datetime.now()
>>> str(now)
'2019-04-20 12:56:26.814148'
>>> print(now)
2019-04-20 12:56:26.814148
>>> repr(now)
'datetime.datetime(2019, 4, 20, 12, 56, 26, 814148)'
```

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# 使用 format() 方法來格式化字串

- format() 方法是一種功能強大的字串格式化迷你語言。
- 它提供了幾乎無限種可能性的字串格式化操作方式。
- format() 方法的完整功能可參考 Python 標準函式庫的字串格式化部分。
  https://docs.python.org/3/library/string.html#format-string-syntax

# 使用位置參數

使用位置參數來填入對應的 {}。

```
>>> "{} is the {} of {}".format("Ambrosia", "food", "the gods")
'Ambrosia is the food of the gods'
>>> "{{Ambrosia}} is the {} of {}".format("food", "the gods")
'{Ambrosia} is the food of the gods'
```

若要顯示 { 以及 } 字元，需重複寫兩次 {{ 以及 }} 。

# 使用位置參數 (續)

使用位置參數傳給 format() 的不一定要是字串，
可以是任何物件，Python 會自動把其轉換為字串。

```
>>> "{} + {} = {}".format(1, 2, 1+2)
'1 + 2 = 3'
>>> x = [1, 2, "three"]
>>> "The {} contains: {}".format("list", x)
"The list contains: [1, 2, 'three']"
```

# 使用編號參數

使用編號參數來設定替換欄位。

```
>>> "{2} is the {0} of {1}".format("food", "the gods", "Ambrosia")
'Ambrosia is the food of the gods'
```

```
>>> '{0}{1}{0}'.format('abc', 'def')
'abcdefabc'
```

# 使用指名參數 (named parameter)

使用指名參數來設定替換欄位。

```
>>> "{food} is the food of {user}".format(food="Ambrosia", user="the gods")
'Ambrosia is the food of the gods'
```

可同時使用編號參數和指名參數。

```
>>> "{0} is the food of {user[1]}".format("Ambrosia", user=["men", "the gods", "others"])
'Ambrosia is the food of the gods'
```

編號參數依定要放在最前面，否則會發生錯誤。

```
>>> "{0} is the food of {user}".format(user="the gods", "Ambrosia")
  File "<stdin>", line 1
SyntaxError: non-keyword arg after keyword arg
```

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# 以 % 符號來格式化字串

這種方式是 Python 早期版本的標準，將被棄用，未來會從 Python 語言中刪除。請避免在新的程式碼中使用。

```
>>> "%s is the %s of %s" % ("Ambrosia", "food", "the gods")
'Ambrosia is the food of the gods'
```

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# 以 f-string 來格式化字串

- 從 Python 3.6 開始，有一種更簡便的語法，可以建立格式化字串，稱為 f-string。
- 以 f 為字首，其語法類似 `format()`。

```
>>> value = 42
>>> message = f"The answer is {value}"
>>> print(message)
The answer is 42
```

```
>>> pi = 3.1415
>>> print(f"pi is {pi:{10}.{2}}")
pi is        3.1
```

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# bytes 物件

- bytes 物件乍看之下有點類似 string 物件。
- 重要區別：
  - string 物件是 Unicode 字元序列，每一個元素 (字元) 依照其編碼可能有 1 ~ 4 Bytes 大小。
  - bytes 物件是一個值為 0 ~ 255 的整數序列，每一個元素固定是 1 Byte 大小。
- bytes 物件適用於處理二進位資料，例如讀取圖片、音樂等資料檔。
- bytes 物件的用法跟字串不同，且不能與字串結合使用。

# bytes 物件的範例

```
>>> unicode_string = '中'
>>> unicode_string
'中'
>>> xb = unicode_string.encode()
>>> xb
b'\xe4\xb8\xad'
>>> xb += 'A'
Traceback (most recent call last):
File "<pyshell#35>", line 1, in
<module>
xb += 'A'
TypeError: can't concat str to bytes
>>> xb.decode()
'中'
```

使用 encode() 方法，把 Unicode 字串轉換為 bytes 物件。

使用 decode() 方法，把 bytes 物件轉換回 Unicode 字串。

# 課程大綱

1. 字串為字元序列
2. 基本的字串操作
3. 特殊字元和轉義字元
4. 字串常用的方法與函式
5. 字串的搜尋
6. 字串的修改
7. `repr()` 和 `str()` 把物件轉換成字串表示
8. 使用 `format()` 方法來格式化字串
9. 以 % 符號來格式化字串
10. 以 f-string 來格式化字串
11. bytes 物件
12. 使用 `print()` 控制字串輸出

# 使用 print() 控制字串輸出

可連續呼叫，印在不同行。

```
>>> print("a")
a
>>> print(1)
1
```

可接受多個參數，印在同一行。

```
>>> print("a", "b", "c")
a b c
```

使用 sep 參數以及 end 參數。

```
>>> print("a", "b", "c", sep="|")
a|b|c
>>> print("a", "b", "c", end="\n\n")
a b c
```

使用 file 參數輸出到檔案。

```
>>> print("a", "b", "c", file=open("testfile.txt", "w"))
```

# 重點整理

| | 有序 | 無序 |
|---|---|---|
| 可變更 (mutable) | list | set |
| 不可變更 (immutable) | tuple、字串 | |

- Python 方便的字串操作，可以幫助我們處理與清理各種龐雜的文字資料。
- `repr()` 在除錯時可以看到更細節的狀態。
- bytes 物件是資料最原始的狀態。
- 字串格式化功能強大，善加利用可以讓程式碼易讀、易維護。