python 函數整理

1. 字串函數 (String Functions)

|  |  |  |
| --- | --- | --- |
| python 字串函數 | 功能 | 說明 |
| + | 字串連結 | print('ABC'+'DEF') #ABCDEF |
| \* | 重複字串 | print('AB'\*3) #ABABAB |
| %s | 字串插值 | product = 'orange'; price = 20  st = 'the %s is %s dollars.' % (product, price)  print(st) #the orange is 20 dollars |
| s='ABCDEFGHIJ'  s[x,y]  s[:y]  s[::3]  s[3:] | 取得部分 子字串 | s='ABCDEFGHIJ'#s[x:y] 從第x+1個字開始取到第y個字  print(s[1:4]) #BCD從第(1+1)個字開始取到第4個字 |
| print(s[3:8]) #DEFGH從第(4+1)個字開始取到第8個字 |
| print(s[:8]) #ABCDEFGH從第(1)個字開始取到第8個字  print(s[::3]) #ADG從第1個字開始隔3個字再選到結尾  print(s[3:]) #DEFGHIJ 從第4個字開始到結尾 |
| s='ABCDEFGHIJ'  s.relace(x1,x2) | 字串取代 | s='ABCDEFGHIJ'  print('new s='+s.replace('B','XY')) # AXYCDEFGHIJ #值不變  s=s.replace('B','XY') #則s為AXYCDEFGHIJ #值變 |
| s.find('x1')  s.index('x1') | 字串位置 | print(s.find('DE')) #3  print(s.find('GHI')) #6  #index()找不到會回覆 ValueError，find() 則回傳 -1。 |
| 利用\* | 字串填補 | print 'A'+'0'\*5)+'B' #A00000B  print 'A'+' '+'B' #A B |
| len(s) | 字串長度 | s='ABCDEFGHIJ'  print(len(s)) -- 10 |
| t.rstrip()  t.lstrip()  t.strip() | t=' Hello '  刪除空白 | 刪除右邊(後)空白 print(t.rstrip()) #' Hello’  刪除左邊(前)空白 print(t.lstrip()) #'Hello '  刪除前後空白 print(t.strip()) #'Hello' |
| t.rjust()  t.ljust()  t.center() | t='Hello'  刪除空白 | 刪除右邊(後)空白 print(t.rjust(7)) #' Hello’  刪除左邊(前)空白 print(t.ljust(7)) #'Hello '  刪除前後空白 print(t.center(7)) #'Hello' |
| List=['A','B','C']  s.jont(list) | 將串列中元素以s字串作為連接字元組成一個字串 | print(''.join(list1)) #ABC  print(' '.join(list1)) #A B C  print('xy'.join(list1)) #AxyBxyC |
| t='Hello'  t.lower()  t=upper() | 字串轉大寫  字串轉小寫 | print(t.lower() # hello  print(t.upper() # HELLO |
| REVERSE() 轉 list  ''.join(list) | 字串的字元順序反轉序 | s='ABCDEFGHIJ'  list1=list(reversed(s)) #轉成串列list1  print(''.join(list1)) #使用join轉成string |
| c=','為分隔字元  st.splite(c) | 以分隔字元c分割字串st成為一個串列 | st='A;B;C;D;E;F'; ss='A B C D E F'  print(st.split(';')) # ['A','B','C','D','E','F']  print(st.split(' ')) # ['A','B','C','D','E','F'] |
| starstwith(c)  endswith(c) | 字串是否以c開始  字串是否以c結尾 | s='ABCDEFGHIJ'  print(s.startswith('A')) #True  print(s.endswith('G')) #False |
| s.islower()  s.isupper() | 字串是否皆為小寫  字串是否皆為大寫 | print(s.islower()) #False  print(s.isupper()) #True |
| s1='This is a Book'  s1.isalpha()  s1. isdecimal()  s1.isalnum()  s1.istitle() | s2='1234568'  字串內是否只含字母  字串內是否只含數字  字串內只含有字母或數字  字串內第每個詞的第一個字母為大寫，其餘皆小寫 | s3='1@az2wsx3edc'  print(s1.isalpha()) #False 因含有空白  print(s2.isdecimal()) #True  print(s3.isalnum()) #True  print(s1.istitle()) #False 但因為有空白 |

1. 數學計算函數

|  |  |  |  |
| --- | --- | --- | --- |
| python 數學函數 |  | 功能 | 說明 |
| int(x)  int(x, base) | X  x,base | 將 x 轉換成整數。  將x轉換成base進位的整數 | print(int('123')) #123  print(int(123.999)) #123  print(int()) #0  print(int(True)) #1  print(int(False)) #0  print(int('123', base=8)) # 83 (八進位)  print(int('123', base=16)) # 291(十六進位) |
| float() | x | 將 x 轉換成浮點數。 | print(float('123')) #123.0  print(float(123)) #123.0  print(float()) #0.0  print(int(True)) #1  print(int(False)) #0 |
| abs() | x | 回傳 x 的絕對值。 | print(abs(-123)) #123 |
| divmod() | x, y | 回傳 x 除以 y 的商和餘數。 | r = divmod(9,2) #(4, 1) tuple  print('商：'+str(r[0])+' 餘數:'+str(r[1])) |
| max() | iter | 回傳可迭代物件 iter 裡的最大值。 | print(max('100200300')) #3 string 中  print(max([120,240,360])) #360 list中  print(max((120,240,360))) #360 tuple中 |
| min() | iter | 回傳可迭代物件 iter 裡的最小值。 | print(min('100200300')) #0 string 中  print(min([120,240,360])) #120 list中  print(min((120,240,360))) #120 tuple中 |
| pow() | x, y, z | 回傳「x 的 y 次方」或「x 的 y 次方除以 z 的餘數」。 | print(pow(2, 3)) #8 2的3次方  print(pow(2, 3, 3)) #2 2的3次方，再除以3的餘數  print(pow(3, 3)) #27 3的3次方  print(pow(3, 3, 4)) #3 3的3次方，再除以4的餘數 |
| sqrt() | x | 會回傳 x 的平方根，計算結果以浮點數 float 格式呈現 | print(math.sqrt(2)) # 1.4142135623730951  print(math.sqrt(3)) # 1.7320508075688772  print(math.sqrt(5)) # 2.23606797749979  print(math.sqrt(16)) # 4.0 |
| exp() | x | math.exp(x) 會回傳 e 常數的 x 次方。 | print(math.exp(1)) # 2.718281828459045  print(math.exp(2)) # 7.38905609893065  print(math.exp(3)) # 20.085536923187668 |
| sum() | iter, y | 回傳串列或 tuple 的數值與 y 的加總。 | print(sum([1,2,3,4])) #10 list  print(sum((1,2,3,4))) #10 tuple  print(sum((1,2,3,4), 5)) #10 tuple |
| bool() | x | 將參數 x 轉變成布林值 False 或 True。 | print(bool(1))  print(bool(0))  print(bool())  print(bool(999))  print(bool('hello'))  print(bool([0]))  print(bool([])) |
| round() | x  x, y | 將x四捨五入, y為小數點位數。  ※round(3.5,1) #4 round(4.5,1) #4.0 可用銀行家法判斷 (限於小數0位)，其他應採較離二端較近者，但需要顯示小數20位才看得出；可以使用自訂函數判斷。 | print(round(3.1415)) #3  print(round(3.1415, 3)) #3.142  print(round(3.1425, 3)) #3.143  print(round(4.1415, 3)) #4.141 #!!說明於下二式  print("{:1.20f}".format(4.1415)) #4.14149999999999973710 近於4.141,遠於4.412  print(round(4.1425, 3)) #4.143 #!!說明於下二式  print("{:1.20f}".format(4.1425)) #4.14250000000000007105 近於4.143,遠於4.142 |
| math.ceil(x) |  | 無條件進位  x =123.7567  import math | print(math.ceil(x))#124  print(math.ceil(x\*1)/1.0) #124.0 進位至小數點後 0 位  print(math.ceil(x\*10)/10.0) #123.8 進位至小數點後 1 位  print(math.ceil(x\*100)/100.0) #123.76 進位至小數點後 2 位 |
| math.floor(x) |  | 無條件捨去  x =123.7567  import math | print(math.floor(x))#123  print(math.floor(x\*1)/1.0) #123.0 進位至小數點後 0 位  print(math.floor(x\*10)/10.0) #123.7 進位至小數點後 1 位  print(math.floor(x\*100)/100.0) #123.75 進位至小數點後 2 位 |
| factorial(x) | x | 回傳 x 的階乘 ( x! )，x 只能使用正整數。 | print(math.factorial(4)) #24 4x3x2x1=24  print(math.factorial(5)) #120 5x4x3x2x1=120  print(math.factorial(6)) #120 6x5x4x3x2x1=720 |

※round(3.5,1) #4 round(4.5,1) #4.0 可用銀行家法判斷 (限於小數1位)，其他應採較離二端較近者，但需要顯示小數20位才看得出；可以使用自訂函數判斷。

1. 日期函數()

python中時間日期格式化符號： now = datetime.now() **#2024-04-25 14:07:56**

now.strftime("**%?**")

|  |  |  |
| --- | --- | --- |
| %y  %Y  %m  %d  %H  %I  %M  %S  %a  %A  %b  %B  %c  %j  %p  %U  %w  %W  %x  %X  %Z  %% | 兩位數的年份表示（00-99）  四位數的年份表示（000-9999）  月份（01-12）  月內中的一天（0-31）  24小時制小時數（0-23）  12小時制小時數（01-12）  分鐘數（00=59）  秒（00-59）  本地簡化星期名稱  本地完整星期名稱  本地簡化的月份名稱  本地完整的月份名稱  本地相應的日期表示和時間表示  年內的一天(001-366)  本地A.M.或P.M.的等價符  一年星期數(00-53)星期天為開始  星期(0-6)，星期天為星期的開始  一年星期數(00-53)星期一為開始  本地相應的日期表示  本地相應的時間表示  當前時區的名稱 # 亂碼  %號本身 | now.strftime("%y") #24  now.strftime("%Y") #2024  now.strftime('%m') #04 now.strftime('%d') #25  now.strftime('%H') #14  now.strftime('%I') #02  now.strftime('%M') #07  now.strftime('%I') #56  now.strftime('%a') #Thu  now.strftime('%A') #Thursday  now.strftime('%b') #Apr  now.strftime('%B') # April  now.strftime('%c') #Thu Apr 25 14:07:56 2024\ now.strftime('%j') #116\ now.strftime('%p') #PM  now.strftime('%U') #16  now.strftime('%w') #4 (週四)  now.strftime('%W') #17  now.strftime('%x') #04/25/24  now.strftime('%X') #14:07:56 |

# （1）當前時間戳記

1714030011.846188

時間->時戳 now.timestamp()： 1714030011.846188

datetime.today()：2024-04-25 15:26:51.846189

now=datetime.now()： 2024-04-25 15:26:51.846188

|  |  |  |
| --- | --- | --- |
| **名稱** | **描述** | **結果** |
| datetime.now() #now | 當前系統日期、時間 | 2024-04-25 15:26:51.846188 |
| datetime.today() | 當前系統日期、時間 | 2024-04-25 15:26:51.846189 |
| timestamp() #ts | 時間轉時間戳記 | 1714030011.846188 |
| ctime() | 回傳日期和時間的字串 | Thu Apr 25 16:34:07 2024 |
| year | 取得年份 | 2024 |
| month |  | 4 |
| day |  | 25 |
| isoformat() #tiso | 時間-> ISO格式的日期字串 | 2024-04-25T15:26:51.846189 |
| fromisoformat(tiso) | ISO時間-> 時間 |  |
| utcnow() #tutc | 時間->utc時間 (差8小時) | 2024-04-25 07:26:51.846189 |
| utcfromtimestamp(ts) | 時戳(ts)->utc時間 |  |
| fromtimestamp(ts) | 時戳->時間 |  |
| now.replace(year=2020) | 取代日期，產生新的物件 | 2020-04-25 15:26:51.846189 |
| weekday() | 一星期中的第幾天，星期一為 0 | 3 |
| isoweekday() | 一星期中的第幾天，星期一為 1 | 4 |
| isocalendar() | 回傳一個 tuple，內容分別是 ( 年、第幾週、isoweekday ) | datetime.IsoCalendarDate(year=2024, week=17, weekday=4) |
| now+timedelta(days=7) | 未來7天 |  |
|  |  |  |
|  |  |  |

**strftime：**將給定格式的日期時間物件轉換為字串。**日期時間物件=>字串，控制輸出格式**

**strptime：**將字串解析為給定格式的日期時間對象。**字串=>日期時間物件，解析字串**

|  |  |  |
| --- | --- | --- |
|  | strftime | strptime |
| 用法 | 根據給定的格式將對日期時間象轉換為字串 | 將字串解析為給定相應格式的datetime 對象 |
| 類型 | 實例方法 | 類方法 |
| 方法 | date; datetime; time | datetime |
| 用法 | strftime(format) | strptime(date\_string, format) |
| 示例 | datetime.datetime(2006, 11, 21, 16, 30) => '2006-11-21 16:30' | "21/11/06 16:30" => datetime.datetime(2006, 11, 21, 16, 30) |

1. 資料形態轉換() --配合DtaTime.py
2. 日期轉各種格式(文字) ：datetime.strftime()

now.strftime("%**?**) #參見上方表格

dt**=**datetime**.**datetime(2006, 11, 21, 16, 30)

dt**.**strftime("%Y-%m-%d %H:%M") #'2006-11-21 16:30'

dt**.**strftime("%Y-%m-%d") #'2006-11-21'

1. 文字轉日期() --配合DtaTime.py

**datetime.strptime 日期格式的文字 轉 日期型態**

dt**=**datetime**.**datetime**.**strptime("21/11/06 16:30", "%d/%m/%y %H:%M")

**print**(dt) #2006**-**11**-**21 16:30:00

1. 文字轉整數

print(int('12'))

1. 整數轉文字()

print(str(12))

print(f’{12}’) #f格式化

1. 實數轉文字()

print(str('12.345'))

print(f”{'12.345'}”) #f格式化

1. 文字轉實數

print(float(12.345))

1. Round(x, y)、FLOOR 、CEILING函數

---Round(x, y)：傳回數值，捨入到指定的長度或有效位數(y)。 #配合(round(x, y).py

print round(10.44,1) --10.40

print round(10.45,1) --10.50.

print round(10.46,1)

print round(1000210/100,0)

print round(1000210.0/100,0)

print round(1000210.1/100,0)

--- floor 、ceil函數，只能處理到「整數」。無法如同 ROUND 函數一般，自行指定要處理的有效位數。

---floor()：傳回小於或等於指定數值運算式的最大整數。

print(math.floor(1000210/100)) #10002

print(math.floor(1000210.0/100)) # 0002

print(math.floor(1000210.1/100)) #10002

---Ceiling()：傳回大於或等於指定數值運算式的最小整數。

print(math.ceil(1000210/100)) #10003

print(math.ceil(1000210.0/100)) #10003

print(math.ceil(1000210.1/100)) #10003