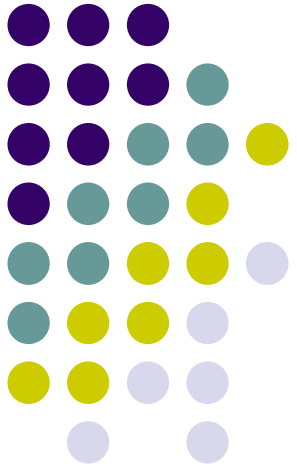
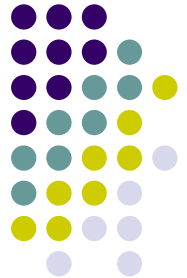
Python程式設計入門 基礎(2/2)

葉難

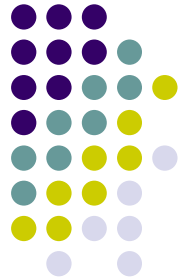


大綱



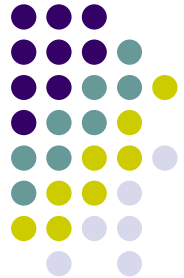
- 物件、型別、名稱、指派（assignment）
- 運算式、運算元、運算子
- 述句（statement），條件判斷、迴圈等
- 函式、參數、回傳、範圍
- 異常（exception）
- 模組、匯入、標準程式庫（內建模組）

函式（function）



- 把程式包起來，抽象化機制
- def述句、lambda運算式、與其他
- 呼叫（call）、參數、回傳
- 參數傳遞機制：骨子裡就是指派（綁定關係）
- 回傳物件
- 名稱可視範圍：全域、區域

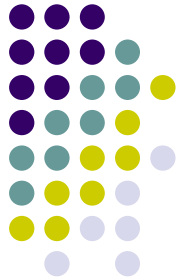
def述句：函式定義



- 我個人認爲函式是電腦科學最重要最基礎的抽象化工具，目標是打造黑盒子、降低心智負擔
- 語法（此處介紹並不全，之後會完整介紹）

```
def 函式名稱(參數0, 參數1, 參數2...)
    述句
    述句
    述句...
```

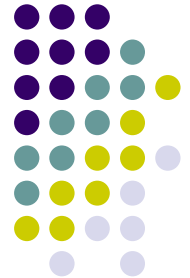
def述句範例



- 算總和

```
def my_sum(numbers):      # 函式頭
    total = 0             # 函式體
    for n in numbers:
        total += n
    return total          # 回傳

print(my_sum([30, 41, 52]))
print(my_sum((3, 4, 5, 6, 7)))
```

參數



- 參數傳遞機制：骨子裡就是指派（綁定關係）

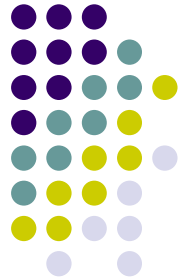
```
def ctof(temp):          # 攝氏溫度轉華氏
    ''' docstring
        Put function spec here
    '''
    rst = []
    for t in temp:
        rst += [temp[i] * 9.0 / 5.0 + 32]
    return rst
data = [-32.5, -18, 0, 10, 28, 33]
print(ctof(data))
```

位置參數、關鍵字參數

- 次序可錯亂
- 明確寫出參數名稱，避免搞錯位置傳錯

```
def f(a, b, c, d):
  print(a, b, c, d)

f(3, 5, c='Amy', d=(3, 4, 5))
f(a=3, b=5, c='Amy', d=(3, 4, 5))
f(b=5, a=3, d=(3, 4, 5), c='Amy')
```

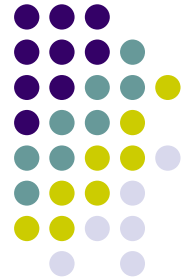
回傳



- 函式結束時，一定會回傳東西（物件）
- 若不以return述句指定，預設回傳None
- 若要回傳兩個以上的物件，要放在list或tuple裡

```
def total_avg(scores, initial=0):   # 參數預設值
    n = 0
    total = initial
    for x in scores:
        total += x
        n += 1
        return (total, total/n)
print(total_avg([60, 70, 80])
```

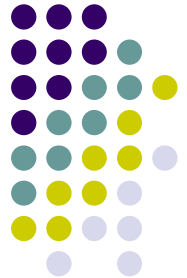
名稱的可視範圍（scope）



- 之前定義在函式之外的名稱（物件）：全域
- 函式參數與內部名稱（物件）：區域
- 在函式裡取用某名稱時，先到函式區域範圍內尋找，然後到全域範圍，然後到「內建」範圍

```
a, b = 3, 4          # 全域
def foo(n):
    m = 5 + a        # 區域內找不到a，會到全域去找
    return 100 + n + m
x = foo(b)
y = a + x + n        # 出錯，找不到n
```

global述句

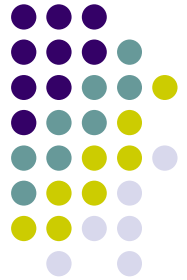


- 若在函式內想重新指派全域名稱指向的物件，須使用global述句；若不，則會建立區域名稱

```
a = 3          # 全域
def foo(n)
    a = 99     # 區域
    return n + a
```

```
a = 3          # 全域
def foo(n)
    global a
    a = 99
    return n + a
```

Python內建函式



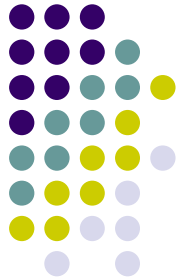
- int、float、str、list、tuple
- abs、pow、round、divmod
- range、len、min、max、sum
- sorted、reversed、zip、enumerate
- id、type、chr、ord
- print、input
- 其他見表格，定義請查Python官方文件

len、min、max、sum

- len：list或tuple或str的長度
- min、max、sum：傳入list或tuple，回傳元素最小值、最大值、總和

```
>>> li = [0, 1, 2, 3, 4]
>>> len(li)
5
>>> sum(li)
10
>>> min(li), max(li)
(0, 4)
```

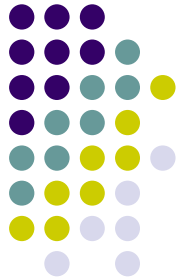
abs、pow、round、divmod



- 絕對值、次方、捨入、商和餘數

```
>>> abs(-3), abs(-4.56)
(3, 4.56)
>>> pow(2, 5), 2 ** 5, pow(2, 5, 3)
(32, 32, 2)
>>> round(3.14159, 2)
3.14
>>> divmod(11, 3)
(3, 2)
>>> divmod(11.5, 3)
(3.0, 2.5)
```

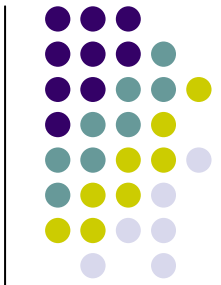
int、float、str



- 型別轉換

```
>>> int(3), int('345'), int(4.56)
(3, 345, 4)
>>> float(3), float('3.14'), float(-4.5)
(3.0, 3.14, -4.5)
>>> str(3), str(3.14), str(-4.5)
('3', '3.14', '-4.5')
>>> int(), float(), str()
(0, 0.0, '')
```

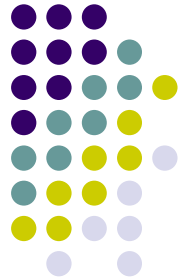
chr、ord



- chr：傳入字元碼（整數）、回傳該字元碼對應的字串（僅含一個字元）
- ord：給定含一個字元的字串，回傳字元碼

```
>>> chr(97)
'a'
>>> ord('a')
97
>>> chr(ord('a') + 1)
'b'
```

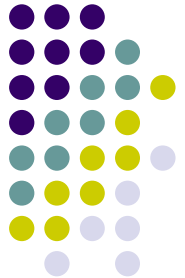
list、tuple



- 建立或複製list物件、建立或複製tuple物件

```
>>> a = [0, 1, 2]
>>> b = list(a)
>>> a is b, a == b
(False, True)
>>> list('xyz')
['x', 'y', 'z']
>>> list()
[]
```

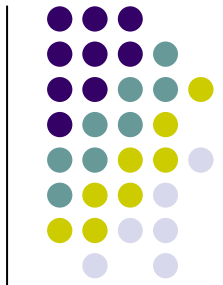
sorted、reversed



- 排序、逆轉

```
>>> a = [3, 7, 9, 5, 1]
>>> b = sorted(a)
>>> b
[1, 3, 5, 7, 9]
>>> c = reversed(b)     # c不是list物件
>>> list(c)
[9, 7, 5, 3, 1]
```

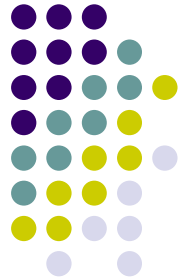
enumerate



- enumerate：加上列舉值（索引）

```
names = ['Amy', 'Bob', 'Cathy']
for x in names:
    print(x)
for i, x in enumerate(names):
    print(i, x)
# 0 Amy
# 1 Bob
# 2 Cathy
```

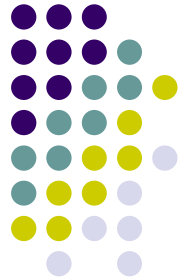
zip：想像成拉鍊



- 依序鍊起兩個（或以上）的list物件（或tuple）

```
>>> x = ['a', 'b' 'c']
>>> y = [0, 1, 2, 3]
>>> list(zip(x, y))
[('a', 0), ('b', 1), ('c', 2)]
>>> list(zip('abcde', y))
[('a', 0), ('b', 1), ('c', 2), ('d', 3)]
>>> list(zip('abc', [0, 1, 2], (10, 11, 12)))
[('a', 0, 10), ('b', 1, 11), ('c', 2, 12)]
```

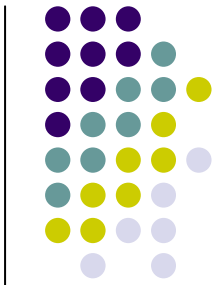
id、type



- 每個物件都擁有獨一無二的id，以及型別（type）、以及值（內容）

```
>>> a, b, c, d = 3, 4.56, [0, 1, 2], 'hello'
>>> id(a)
505996176                    # id就是個數字
>>> type(a), type(b)
(<class 'int'>, <class 'float'>)
>>> type(c), type(d)
(<class 'list'>, <class 'str'>)
```

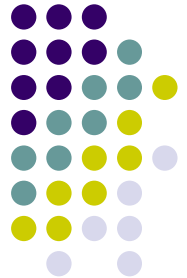
print



- 印出，預設會在最後加上換行字元

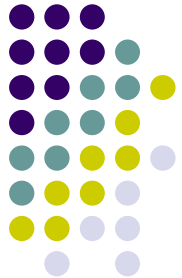
```
>>> print(3, 4.56, 'hi', [0, 1, 2])
3 4.56 hi [0, 1, 2]
>>> print(2016, 7, 7, sep='/')   # 相隔
2016/7/7                         # 預設結尾是換行
>>> print(3, 4, end='XXX')       # 指定結尾
3 4XXX>>>
>>> print(3, 4, end='')          # 無結尾
3 4>>>
```

2.x版的print是述句



```
>>> print 'hello'
hello
>>> print 'hello', 'python'
hello python
>>> print('hello', 'python')  # 這是tuple
('hello', 'python')
>>> print 'hello', 'python' ; print '---'
hello python
---
>>> print 'hello', 'python', ; print '---'
hello python ---
```

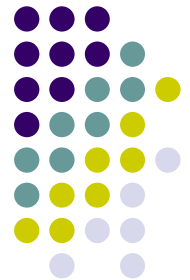
input



- 輸入，取得的東西是個str物件

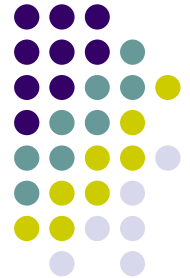
```
>>> x = input('Input your name: ')
Input your name: John     # 輸入John
>>> x
'John'
>>> int(input('Input your age: '))
Input your age: 18        # 輸入18
18
```

問題：身分證檢查碼
id_checksum.py



- 共10碼：1個英文字母、8個數字、1個數字（驗證碼）
- 英文字母轉成數字（下一頁），個位數乘9再加上十位數，得到一數(A)
- 8個數字，從左到右分別乘上8、7、6、...、1，然後相加，得到一數(B)
- (A)加(B)得到(C)，除以10算出餘數(D)
- (D)若是0，驗證碼就是0；若(D)非0，則驗證碼是「10減(D)」

身分證開頭字母對應數字

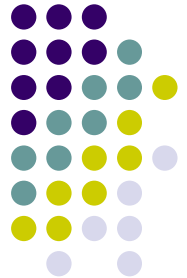


- A=10 台北市
  B=11 台中市
  C=12 基隆市
  D=13 台南市
  E=14 高雄市
  F=15 台北縣
  G=16 宜蘭縣
  H=17 桃園縣
  I=34 嘉義市
  J=18 新竹縣
  K=19 苗栗縣
  L=20 台中縣
  M=21 南投縣
  N=22 彰化縣
  O=35 新竹市
  P=23 雲林縣
  Q=24 嘉義縣
  R=25 台南縣
  S=26 高雄縣
  T=27 屏東縣
  U=28 花蓮縣
  V=29 台東縣
  W=32 金門縣
  X=30 澎湖縣
  Y=31 陽明山
  Z=33 連江縣
- PS：字母後面的第一個數字，1代表男性，2代表女性

舉例：身分證檢查碼

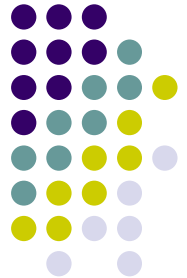
A	1	2	3	4	5	6	7	8	9
	8	7	6	5	4	3	2	1	
1	8	14	18	20	20	18	14	8	121
									1
									9
S	2	1	2	3	0	5	4	3	8
	8	7	6	5	4	3	2	1	
56	16	7	12	15	0	15	8	3	132
									2
									8

模組（**module**）



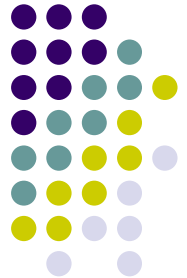
- Python程式的基本執行單位，須符合一定規格
- Python程式檔就是模組
- 第一支交給直譯器的Python程式檔，通常稱爲進入點、主程式檔或主模組檔
- Python標準程式庫（內建模組）
- CPython提供的內建模組，以C語言撰寫

sys.argv：命令列參數



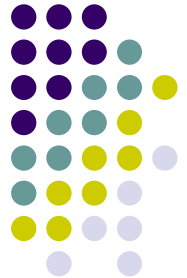
- 是個list
- argv[0]是程式檔的檔名
- argv[1]、argv[2]...是命令列後面跟著的參數
- 例「python hello.py -f a.txt -y」
- argv[0]會是'hello.py'
- argv[1]會是'-f'
- argv[2]會是'a.txt'
- argv[3]會是'-y'

import述句：內建模組keyword



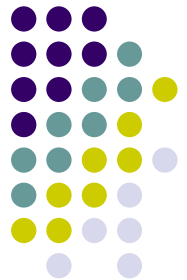
```
>>> import keyword
>>> keyword.iskeyword('import')
True
>>> len(keyword.kwlist)
33
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as',
  'assert', 'break', 'class', 'continue',
  'def', 'del', 'elif', 'else', 'except',
  'finally', 'for', 'from', 'global', 'if',
  'import', 'in', 'is', 'lambda', 'nonlocal',
  'not', 'or', 'pass', 'raise', 'return',
  'try', 'while', 'with', 'yield']
```

import述句：內建模組random



```
>>> import random
>>> random.randint(1, 6)   # 1~6
3
>>> import random as r      # 取個短名稱
>>> r.choice(['ape', 'dog', 'cat'])
'cat'
>>> li = [1, 2, 3]; r.shuffle(li); li
[2, 3, 1]
>>> r.random()     # 0~1.0（不含）的亂數
0.8579643436609136
```

範例(1/2)：模組檔mymath.py

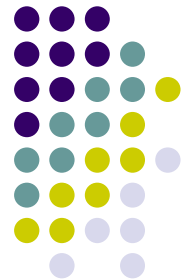


```
pi = 3.14
def gcd(a, b)   # 最大公因數
    while b:
        a, b = b, a%b
    return a
def factorial(n):   # 階乘
    result = 1
    for i in range(1, n+1):
        result *= i
    return result

if __name__ == '__main__':
    print('mymath as main program')
else:
    print('mymath as module')
```

範例(2/2)：主程式檔myhello.py

```
import mymath

print('pi is ' + str(mymath.pi))
print('gcd of 24 and 16 is ' +
  str(mymath.gcd(24, 16)))
print('factorial of 6 is ' +
  str(mymath.factorial(6)))

if __name__ == '__main__':
    print('myhello as main program')
else:
    print('myhello as module')
```

未來匯入（**future import**）：很像匯入模組，但其實不是



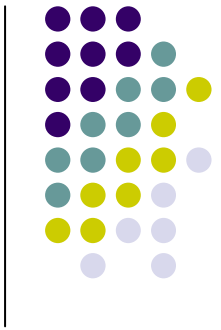
- 在程式檔案開頭處，其他程式碼之前、其他import述句之前
- 2.x版若想使用3.x版的print函式

```
from __future__ import print_function
```

- 2.x版若想使用3.x版的「/」

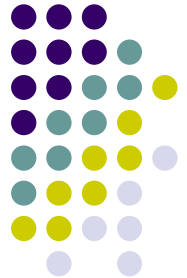
```
from __future__ import division
```

類別（**class**）



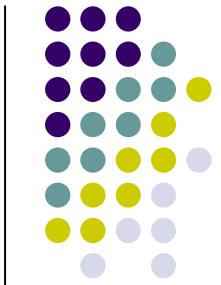
- 2.x版：型別（type）與類別（class）
- 3.x版，兩者合併，一般以型別指Python內建提供的，以類別指我們自己定義的
- 2.x版分新舊式類別，3.x版只有新式
- 把資料與行爲（操作動作）包起來
- 類別、建構式（初始化）、屬性項（attribute）、方法（method）、「.」存取語法

Python內建型別complex



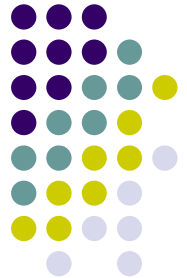
```
>>> z = 3 + 4j                  # 語法，j表虛部
>>> z = complex(5, 6)           # 建構式
(5+6j)
>>> type(z)
<class 'complex'>
>>> z.real, z.imag              # 屬性項
(5.0, 6.a0)
>>> z.conjugate()               # 方法，共軛複數
(5-6j)
```

Person類別(1/2)：定義 Person.py



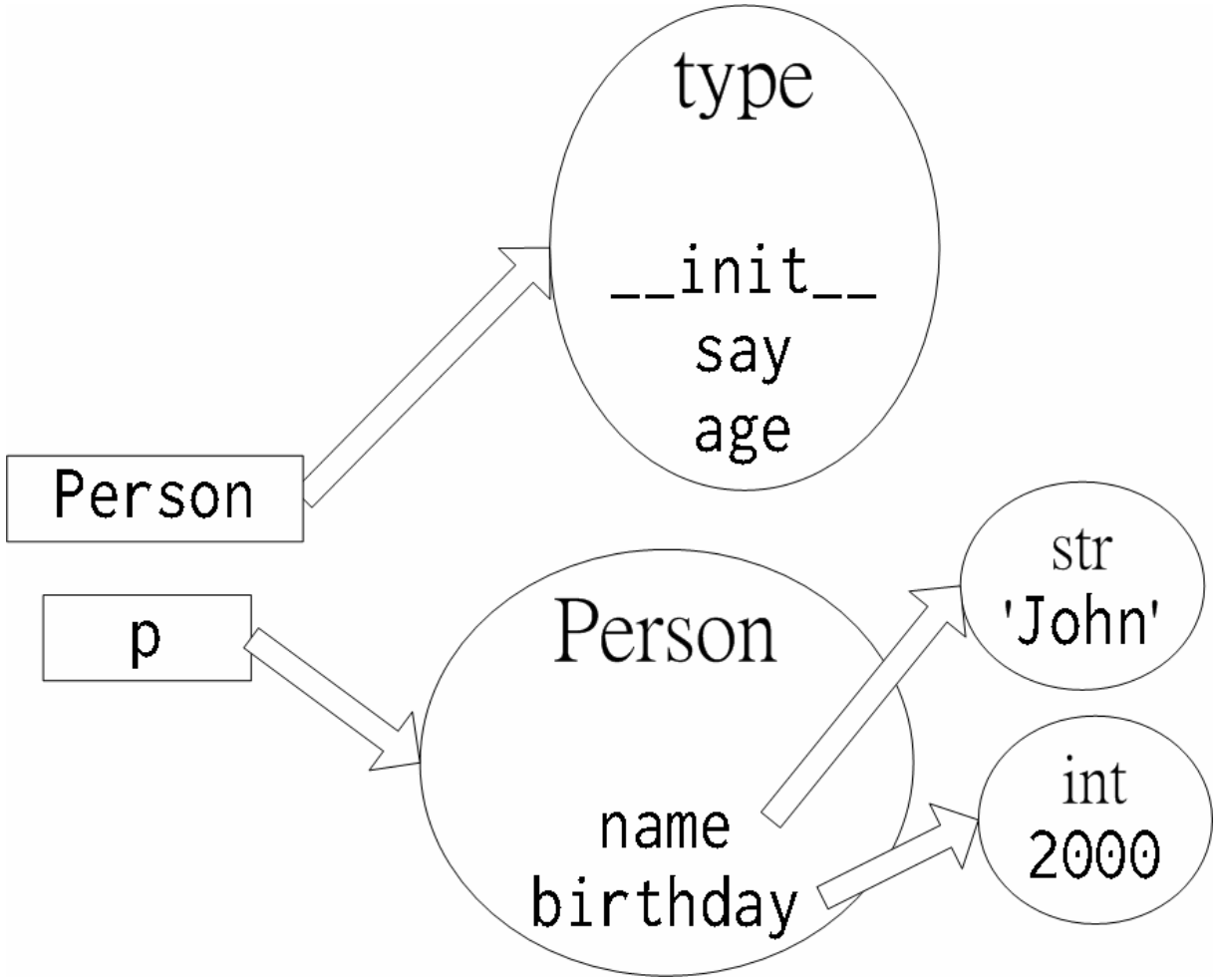
```
from datetime import datetime

class Person():
    def __init__(self, name, birthyear):
        self.name = name
        self.birthyear = birthyear

    def say(self, msg='Hello'):
        print(self.name + ' says ' + msg)

    def age(self):
        return datetime.now().year - self.birthyear
```

Person類別(2/2)：使用

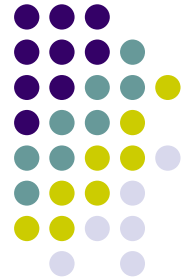


```
if __name__ == '__main__':
    p = Person('John', 2000)

    print(p.name, p.birthyear)  # 屬性項

    p.say()                     # 方法
    p.say('it is so hot')

    print(p.name + ' is ' + str(p.age()) + '
  years old.')
```

Person類別：示意圖

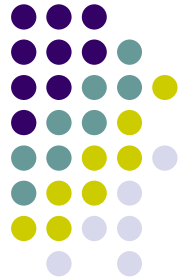


異常（exception）



- 錯誤種類：功能邏輯、語法、錯字、縮排等
- 異常處理機制：異常物件、引發、捕抓
- 發生錯誤的地方，若不知道怎麼辦則引發異常，在適當之處捕抓並處理
- 內建異常物件的型別：BaseException、Exception、SyntaxError、StopIteration、ValueError、TypeError、等等

異常的語法



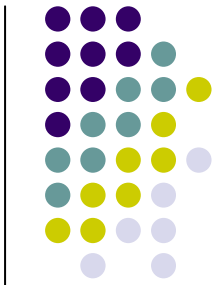
- 引發異常

raise 異常型別名或物件

- 捕抓異常

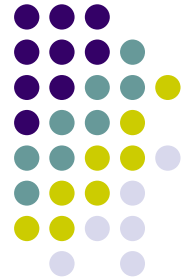
```
try:                                    # 可能引發異常的程式碼
   述句...
except 異常型別名:                       # 捕抓異常
      述句...
except 異常型別名 as 名稱:               # 賦予異常物件名稱
      述句...
else:                 # 若無發生異常的話，
      述句...         # 執行此處程式碼
finally:              # 不論有無發生異常，
      述句...         # 都會執行此處
```

異常範例(1/2) ：階乗函式定義 fact_exception.py



```
def fact(n):
    if n < 0:
        raise ValueError('Argument must be non-negative')

    result = 1
    for i in range(1, n+1):
        result *= i
    return result
```

異常範例(2/2)：使用



```
while True:
    try:
        s = input('input n: ')
        sn = int(s)
        f = fact(sn)
    except TypeError as e:
        print('Error: ' + str(e))
    except ValueError as e:
        print('Error: ' + str(e))
    except Exception:
        print('Error: unknown error')
    else:
        print(str(sn) + '! = ' + str(f))
```

Q&A