

การเขียนฟังก์ชัน

ภาควิชาวิศวกรรมคอมพิวเตอร์

จุฬาลงกรณ์มหาวิทยาลัย

๒๕๖๒

หัวข้อ

- Function call
- Defining a function
- Parameter and local variable
- return statement

การเรียกใช้ฟังก์ชัน

- เขียนฟังก์ชันเพื่อแบ่งหน้าที่การทำงานเป็นส่วน ๆ
- เคยใช้ built-in function มาหลายฟังก์ชันแล้ว
 - `print("Hello")`, `input()`, `len(x)`, `round(2/3, 2)`, `abs(-2)`,...
- จะใช้ฟังก์ชันใด ต้องรู้
 - ชื่อฟังก์ชัน, ข้อมูลที่ต้องส่งให้ฟังก์ชัน, ผลที่ได้จากฟังก์ชัน
- ตัวอย่าง: `print(x)`
 - `x` เป็นข้อมูลแบบใด ๆ ก็ได้
 - ไม่มีผลที่คืนกลับจากฟังก์ชัน (เราจึงไม่เขียน `a = print(3/5)`)
- ตัวอย่าง: `round(x,d)`
 - `x` เป็นจำนวน, `d` เป็นจำนวนหลักหลังจุดทศนิยม
 - ผลที่ได้คือจำนวนจริง `x` ที่มีจำนวนหลักจุดตามที่กำหนด เช่น `round(2/3,2)` ได้ 0.67

เริ่มด้วยฟังก์ชันง่าย ๆ

```
def hello(name):  
    print("Hello", name)
```

```
BLACKPINK = ["Jisoo", "Jennie", "Rosé", "Lisa"]  
for e in BLACKPINK:  
    hello(e)
```



```
Hello Jisoo  
Hello Jennie  
Hello Rosé  
Hello Lisa
```

ค่าของ **e** ถูกส่งไปให้ตัวแปร **name**
ของฟังก์ชัน **hello**
แล้วก็ย้ายการทำงานไปที่ **hello**
ทำคำสั่งในฟังก์ชันเสร็จ ก็กลับมาทำต่อ

ฟังก์ชันหนึ่งเรียกอีกฟังก์ชันก็ได้

```
def hello(name):  
    print("Hello", name)
```

```
def hello_all(names):  
    for e in names:  
        hello(e)
```

```
BLACKPINK = ["Jisoo", "Jennie", "Rosé", "Lisa"]  
hello_all( BLACKPINK )  
hello_all( ["Joe", "John", "Kong"] )
```

```
Hello Jisoo  
Hello Jennie  
Hello Rosé  
Hello Lisa  
Hello Joe  
Hello John  
Hello Kong
```

แบบฝึกหัด: แสดงอะไร ?

```
def print1(a):  
    for e in a:  
        print(e)  
  
def print_all(A):  
    for e in A:  
        print1(e)  
  
print_all( ["A", ["AB", "C"]] )  
print_all( ["ABC"] )
```

แบบฝึกหัด: เครื่องบวกเลขฐานสอง

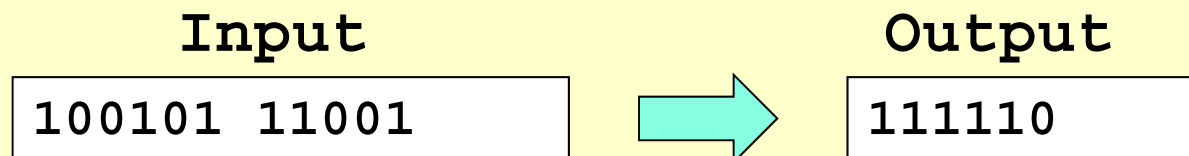
bin(x) :

Return a binary string prefixed with "0b" constructed from an integer **x**

int(x, base) :

Return an integer constructed from a string **x** in the specified **base** (default is 10).

จงเขียนโปรแกรมรับจำนวนเต็มในรูปฐานสอง 2 จำนวน เพื่อหาผลบวกและแสดงในรูปฐานสอง โดยใช้ built-in fuctions **bin** กับ **int** ข้างบนนี้



องค์ประกอบของฟังก์ชัน

ชื่อฟังก์ชัน

รายการของพารามิเตอร์

```
def dot( u, v ) :  
    p = 0  
    for i in range( len(u) ) :  
        p += u[i] * v[i]  
    return p
```

จบการทำงานของฟังก์ชัน กลับไปทำงานต่อที่ผู้เรียกฟังก์ชัน และยังคืนผลจากฟังก์ชันให้ผู้เรียกได้ด้วย

ต้องเยื้องคำสั่งทั้งหลายในฟังก์ชันไปทางขวาให้ตรงกัน

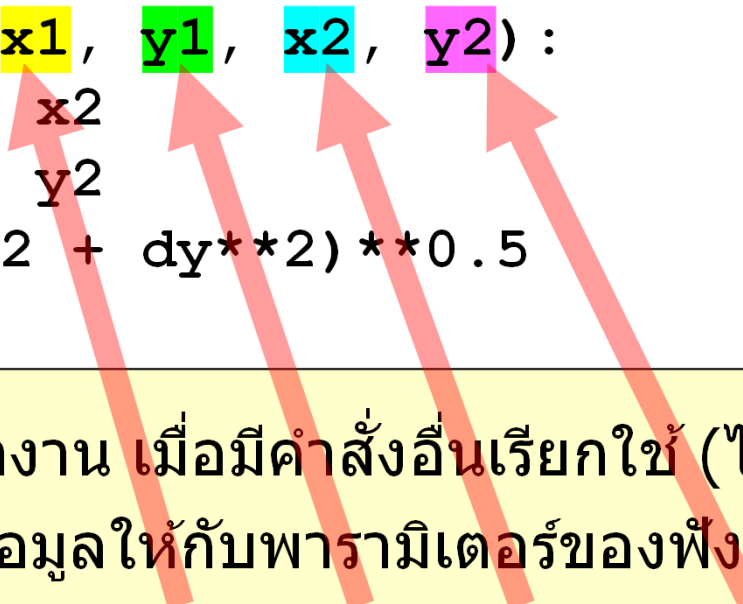
ตัวแปรภายในฟังก์ชันใดเป็นของฟังก์ชันนั้น

- พารามิเตอร์และตัวแปรในฟังก์ชันใด
 - ชื่อซ้ำกับของฟังก์ชันอื่นได้
 - เรียกใช้ตัวแปร ที่อยู่ในฟังก์ชันอื่นไม่ได้

```
def read_data():  
    d = []  
    for i in range(int(input())):  
        d.append(int(input()))  
    return d  
#-----  
def get_mean(d):  
    return sum(d)/len(d)  
#-----  
def get_median(d):  
    d.sort()  
    n = len(d)  
    return (d[(n-1)//2]+d[n//2])/2
```

พารามิเตอร์รับข้อมูลจากผู้เรียกฟังก์ชัน

```
def distance(x1, y1, x2, y2):  
    dx = x1 - x2  
    dy = y1 - y2  
    d = (dx**2 + dy**2)**0.5  
    return d
```



- คำสั่งในฟังก์ชันเริ่มทำงาน เมื่อมีคำสั่งอื่นเรียกใช้ (ไม่เรียกไม่ทำ)
- คำสั่งที่เรียก ต้องส่งข้อมูลให้กับพารามิเตอร์ของฟังก์ชัน

```
d1 = distance(10.5, 12.0, 30.0, 50.5)
```

```
x = 10.5; y = 11.2
```

```
d2 = distance(x, y, 30.0, 50.5)
```

- เมื่อคำสั่งในฟังก์ชันเริ่มทำงาน ถือได้เลยว่าพารามิเตอร์มีค่าแล้ว

```
def distance(x1, y1, x2, y2):  
    x1 = float(input()) # แปลก !!!
```

return : คำนีการทำงานกลับสู่ผู้เรียก

- ใช้คำสั่ง **return** เฉย ๆ
- หรือไมก็ เมื่อทำงานถึงคำสั่งล่างสุดของฟังก์ชัน ก็คือการคืนการทำงาน

```
def hello(name):  
    print("Hello", name)  
    return
```

```
def hello(name):  
    print("Hello", name)
```

```
def hello(name):  
    if name == "":  
        return  
    print("Hello", name)
```

return: คำนวณการทำงาน และคืนผลการทำงานได้ด้วย

```
def read_vector() :  
    x = input().split()  
    v = []  
    for i in range(len(x)) :  
        v.append( float(x[i]) )  
    return v
```


return คำนวณการทำงานกลับสู่ผู้เรียก
และยังคืนผลให้ผู้เรียกได้ด้วย

```
def dot(u, v) :  
    p = 0  
    for i in range(len(u)) :  
        p += u[i]*v[i]  
    return p
```

```
u = read_vector() # เรียกครั้งแรกก็ได้กลับหนึ่งลิสต์  
v = read_vector() # เรียกอีกครั้งก็ได้กลับมาอีกลิสต์  
dot_p = dot(u, v)  
print("u.v =", dot_p)
```

return: คืนผลหลายค่าก็ได้

```
# roots of ax**2 + b*x + c = 0
def roots(a, b, c):
    t = (b**2 - 4*a*c)**0.5
    r1 = (-b + t)/(2*a)
    r2 = (-b - t)/(2*a)
    return r1, r2
```



```
x1, x2 = roots(6, -6, -36)
print(x1, x2)
```

```
def get_odds( x ):
    odds = []
    for e in x:
        if e%2 == 1:
            odds.append(e)
    return odds
```

หรือใช้การคืนลิสต์ก็ได้

ไม่คืนผลทาง return แต่แก้ไขลิสต์ที่ผู้เรียกส่งมา

```
def abs_all( data ) :  
    d = [0] * len(data)  
    for k in range(len(data)) :  
        d[k] = abs(data[k])  
    return d
```

```
x = [1, -2, 3, -4, -5]  
x = abs_all(x)  
print( x )
```

- สร้างลิสต์ที่เก็บผล
- คืนผลด้วย return

```
def abs_all( data ) :  
    for k in range(len(data)) :  
        data[k] = abs(data[k])
```

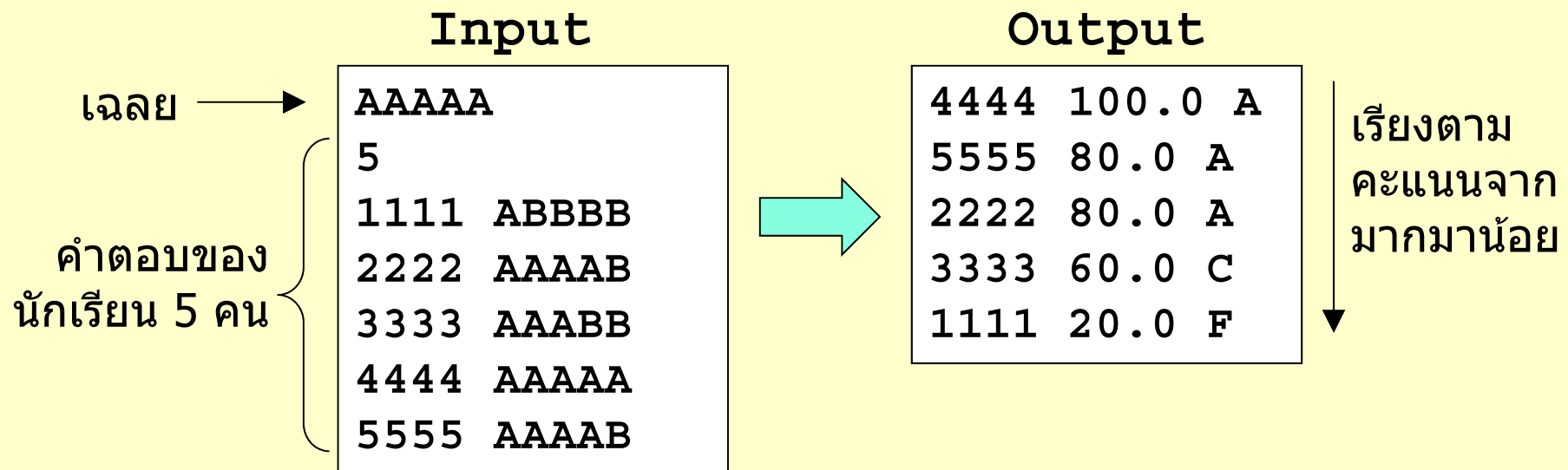
```
x = [1, -2, 3, -4, -5]  
abs_all(x) # ข้อมูลในลิสต์ x เปลี่ยน  
print( x )
```

- แก้ไขข้อมูลในลิสต์ที่รับมา
- ไม่มีผลส่งคืน

แบบฝึกหัด: ตรวจ + ให้เกรด + เรียง + แสดงผล

เขียนโปรแกรมอ่านเฉลยและคำตอบของนักเรียน (หลายคน)
จากนั้นตรวจให้คะแนนคำตอบและให้เกรดทุกคน แล้วก็นำผลที่ได้
ไปเรียงลำดับ ปิดท้ายด้วยการแสดงผลลัพธ์ (ดูตัวอย่าง)

มีฟังก์ชันเขียนมาให้แล้วจำนวนหนึ่ง ให้เขียนโปรแกรมที่เรียกใช้
ฟังก์ชันเหล่านี้ให้ทำงานตามที่ต้องการ (ตัวโปรแกรมที่ต้องเขียน
มีคำสั่งไม่เกิน 6 คำสั่ง)



แบบฝึกหัด: ทำความเข้าใจฟังก์ชันเหล่านี้

```
def read_answers():
    N = int(input())
    answers = []
    for k in range(N):
        sid, ans = input().split()
        answers.append([sid, ans])
    return answers

def marking(answer, solution):
    p = 0
    for i in range(len(answer)):
        if answer[i] == solution[i]:
            p += 1
    return p

def grading(score):
    g = [[80,"A"], [70,"B"], [60,"C"], [50,"D"]]
    for a,b in g:
        if score >= a:
            return b
    return "F"
```


แบบฝึกหัด: ทำความเข้าใจฟังก์ชันเหล่านี้


```
def scoring(answers, solution):
    scores = []
    for [sid, ans] in answers:
        score = marking(ans, solution) / \
            len(solution) * 100
        grade = grading(score)
        scores.append([sid, score, grade])
    return scores


def report(scores):
    for [sid,score,grade] in scores:
        print(sid, grade)

def sort(scores):
    x = []
    for [sid,score,grade] in scores:
        x.append([score, sid, grade])
    x.sort()
    for i in range(len(x)):
        scores[i] = [x[i][1], x[i][0], x[i][2]]
```

ข้อควรระวัง: อย่าแก้ไขค่าของพารามิเตอร์ด้วย =

 `def f1(x):`
 # ตัวแปร `x` เปลี่ยนค่า แต่ตัวแปรที่ส่งค่ามาไม่เปลี่ยน
 `x = 0`

 `def f2(x):`
 # ตัวแปร `x` เปลี่ยนค่า แต่ตัวแปรที่ส่งค่ามาไม่เปลี่ยน
 `x = [0] * len(x)`

 `def f3(x):`
 for i in range(len(x)):
 `x[i] = 0` # ตัวแปรที่ส่งค่ามาก็เปลี่ยนด้วย

```
a = 9
f1(a)           # a เหมือนเดิม
b = [1, 2, 3, 4]
f2(b)           # b เหมือนเดิม
f3(b)           # b เปลี่ยน
```

ความผิดพลาดที่พบบ่อย

```
def hello(name):  
    print("Hello", name)
```

```
def double(u):  
    return 2*u
```

```
x = hello("Bell")      # ไม่มีผลคืนมา แต่มีตัวรับผล  
double([1,2,3])        # คืนผลแต่ไม่มีตัวรับผล
```

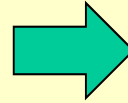
แบบนี้ x มีค่าเป็น None
(None เป็นค่าพิเศษค่าหนึ่งใน Python)

```
def clip(x):  
    if x < 0:  
        return 0  
        x += 2          # มีคำสั่งอยู่ตามหลัง return  
    return x
```

```
def is_odd(x):          # บางกรณีคืนค่า บางกรณีไม่คืน  
    if x%2 == 1: return True
```

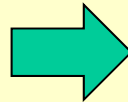
Tips

```
def is_odd(n):  
    if n%2 == 1:  
        return True  
    else:  
        return False
```



```
def is_odd(n):  
    return n%2 == 1
```

```
def foo(n):  
    if n%2 == 1:  
        return 3*n+1  
    else:  
        return n//2
```



```
def foo(n):  
    if n%2 == 1:  
        return 3*n+1  
    return n//2
```

แบบฝึกหัด: ระยะระหว่างจุดสองจุด

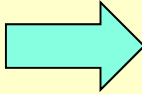
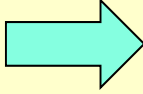
```
def distance(x1, y1, x2, y2):  
    # คำนวณระยะห่างระหว่างจุด (x1,y1) กับ (x2,y2)
```

```
d = distance(1, 1, 4, 5)  
print(d)
```

แบบฝึกหัด: แสดงอะไร ?

```
def f(x):  
    return 2*x  
  
def g(x):  
    return x+5  
  
def h(x):  
    return x//2  
  
x = 4  
y = f(g(h(x)))  
print(x, "-->", y)
```

แบบฝึกหัด: หาจำนวนเฉพาะตัวแรกที่มีค่ามากกว่า N

| Input | | Output |
|-------|---|--------|
| 12 |  | 13 |
| 43 |  | 47 |

```
def is_prime(n):  
    if n <= 1:  
        return False  
    for k in range(2,n):  
        if n%k == 0:  
            return False  
    return True  
def next_prime(N):
```