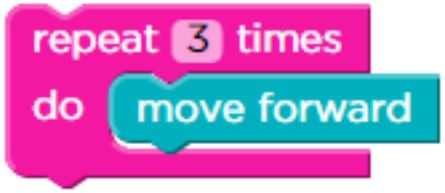
Selection + Loop

ทบทวน

การทำซ้ำคือการทำงานคล้ายคลึงเดิม มีรูปแบบหลักอยู่ 2 แบบคือ

- การทำซ้ำตามจำนวนครั้งที่กำหนด
- การทำซ้ำจนกว่าเงื่อนไขที่กำหนดเป็นเท็จ



ใน Python นั้นมีคำสั่งสำหรับการทำซ้ำ (Loops) อยู่ 2 ชนิด ได้แก่ for และ while

"for"

คำสั่ง for นั้นจะทำซ้ำบนสิ่งของภายในลำดับใดๆ (list หรือ string) ตามลำดับที่ปรากฏในลำดับเหล่านั้น เช่น

```
words = ['apple', 'banana', 'coconut']
for fruit in words:
    print(fruit, len(fruit))
```

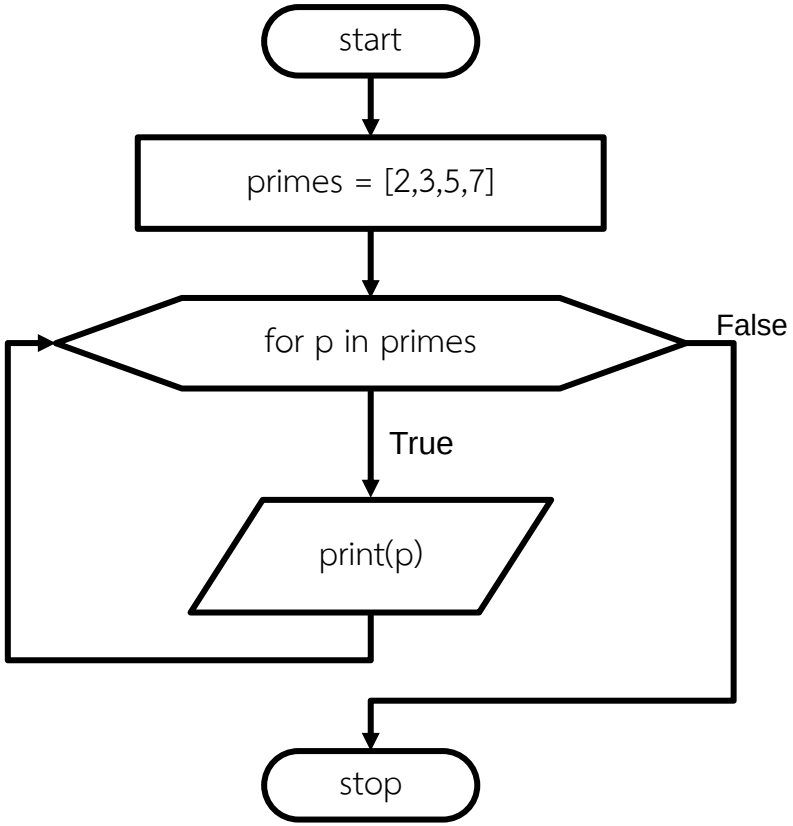
ผลลัพธ์

```
apple 5
banana 6
coconut 7
```

"for" loop

"for" loop จะทำซ้ำตามลำดับที่กำหนดให้ ตัวอย่างเช่น

```
primes = [2, 3, 5, 7]
for p in primes:
    print(p)
```



Output

```
2
3
5
7
```

"for" loop สามารถทำซ้ำบนลำดับของตัวเลขได้โดยใช้ฟังก์ชัน "range"

ฟังก์ชัน "range" จะคืนค่าลำดับที่ไม่เปลี่ยนแปลงของตัวเลขจากช่วงที่กำหนดให้ โดยทั่วไปจะถูกใช้งานคู่กับการทำซ้ำตามจำนวนครั้งที่กำหนดใน "for" loop

"for" loop with range

การใช้ "for" loop กับฟังก์ชัน range กำหนดให้ ตัวอย่างเช่น

```
for i in range(5):
    print(i)
```

Output

```
0
1
2
3
4
```

ฟังก์ชัน range()

เป็นฟังก์ชันที่ทำหน้าที่สร้างลำดับของตัวเลขเช่น ลำดับของเลขที่ติดกัน 0 – 3 หรือลำดับแบบช่วงห่างเท่ากันเช่น 2, 4, 6, 8

มี 3 รูปแบบการใช้งาน

- range(stop)
- range(start, stop)
- range(start, stop, step)

ทั้งนี้ start, stop, step ต้องเป็นเลขจำนวนเต็มเท่านั้น

range(stop)

การใช้งาน range(stop) เป็นการสร้างลำดับ 0, 1, 2, …, n-1

ตัวอย่าง

```
list(range(5))
```

```
[0, 1, 2, 3, 4]
```

```
list(range(8))
```

```
[0, 1, 2, 3, 4, 5, 6 ,7]
```

range(start, stop)

การใช้งาน range(start, stop) เป็นการสร้างลำดับ start, start+1, …, stop-1 ตัวอย่าง

```
list(range(10, 15))
```

```
[10, 11, 12, 13, 14]
```

```
list(range(-9, -1))
```

```
[-9, -8, -7, -6, -5, -4, -3, -2]
```

```
list(range(100, -15))
```

```
[]
```

range(start, stop, step)

การใช้งาน range(start, stop, step) เป็นการสร้างลำดับ
ที่เปลี่ยนแปลงจากจากค่าเดิมด้วยค่า step โดยมีสองกรณีขึ้นกับค่า step
ว่าเป็นค่าบวกหรือลบ

สำหรับ step ที่มีค่าเป็น**บวก**

r[i] = start + step*i เมื่อ i >=0 และ r[i] < stop

สำหรับ step ที่มีค่าเป็น**ลบ**

r[i] = start + step*i เมื่อ i >=0 และ r[i] > stop

range(start, stop, step)

ตัวอย่าง

```
list(range(0, 10, 3))
```

```
[0, 3, 6, 9]
```

```
list(range(0, 10, -3))
```

```
[]
```

```
list(range(0, -10, -3))
```

```
[0, -3, -6, -9]
```

เมื่อใช้ฟังก์ชัน range ลำดับของตัวเลขที่ได้สามารถทำการดำเนินการทั่วไปได้ ยกเว้น concatenation และ repetition

เช่น

```
r = range(0, 20, 2)
11 in r
```

```
False
```

```
10 in r
```

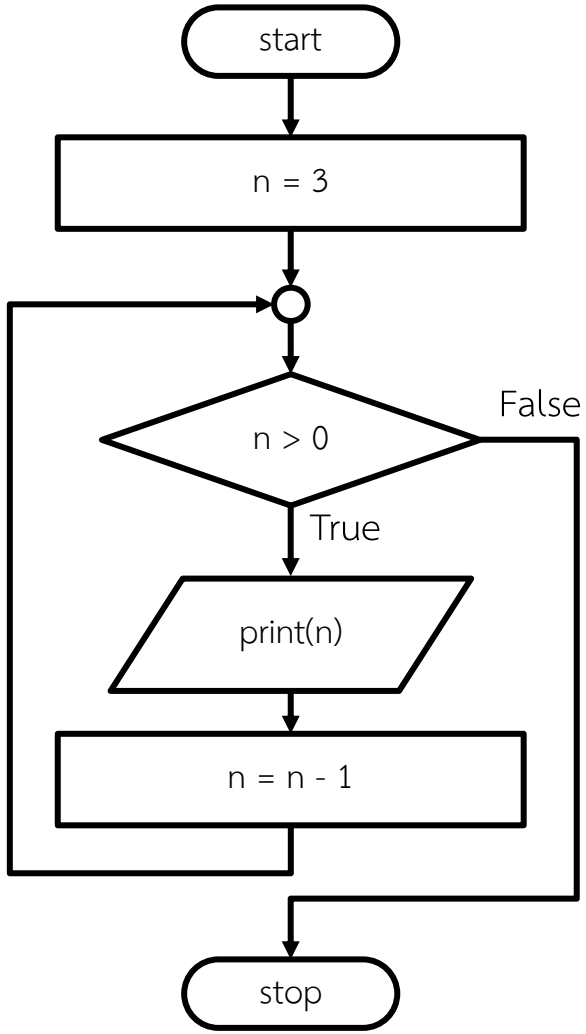
```
True
```

ข้อดีของการใช้ฟังก์ชัน range เมื่อเทียบกับการใช้ list หรือ tuple คือ range ใช้หน่วยความจำที่น้อยกว่า

"while" loop

"while" loop จะทำซ้ำจนกว่าเงื่อนไขที่กำหนดเป็นเท็จ ตัวอย่างเช่น

```
n = 3
while n > 0:
    print(n)
    n = n - 1
```



ผลลัพธ์

```
3
2
1
```

เราได้เรียนโครงสร้างการเขียนโปรแกรม

- แบบลำดับ (Sequence)
- มีการตัดสินใจ (Decision)
- มีการวนซ้ำ (Iteration, Loops)

จากนี้ไปเราจะนำเอาโครงสร้างการเขียนโปรแกรมทั้ง 3 แบบมาใช้ร่วมกัน โดยเราจะประยุกต์ใช้กับการแก้ปัญหา

วาดรูปสี่เหลี่ยมจัตุรัส

- รับข้อมูลเลขจำนวนเต็ม 1 ตัว (n) จากนั้นพิมพ์สี่เหลี่ยมจัตุรัสด้วยอักขระ x ให้เป็นรูปสี่เหลี่ยมจัตุรัส
- ตัวอย่างเช่น

รับค่า

3

แสดงผล

xxx

xxx

xxx

วิเคราะห์โจทย์

รับค่า n

ต้องพิมพ์ x กี่ตัว

ต้องพิมพ์ x กี่บรรทัด

ต้องพิมพ์ x บรรทัดละกี่ตัว

วิเคราะห์โจทย์

รับค่า n

ต้องพิมพ์ x กี่ตัว

n^2 (nxn)

ต้องพิมพ์ x กี่บรรทัด

n บรรทัด

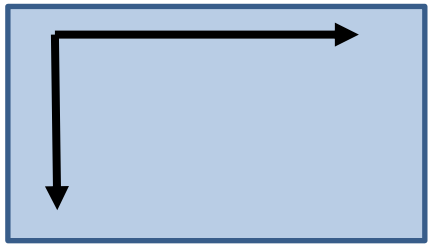
ต้องพิมพ์ x บรรทัดละกี่ตัว

n ตัว

พบว่ามีการทำซ้ำเดิม

ถามว่าจะวนทำแบบไหนดี

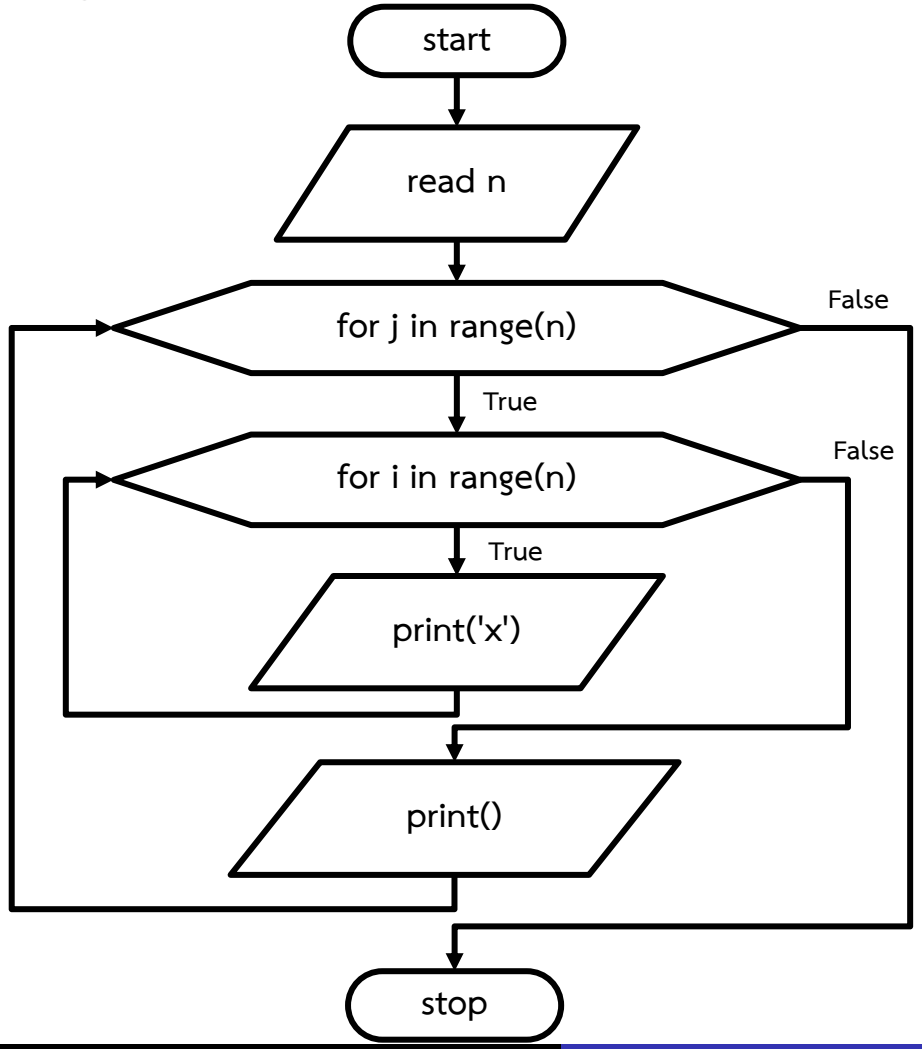
- วนทำทีละบรรทัด
- วนทำทีละหลัก



เราขึ้นบรรทัดใหม่แล้วย้อนกลับมาไม่ได้ดังนั้นเราจะวนทำทีละบรรทัด

flowchart

flowchart ของปัญหานี้ควรมีหน้าตาเป็นอย่างไร



- เราเลือกวนทำทีละบรรทัด
- แต่ละบรรทัดเราทำอะไร

  พิมพ์ x จำนวน n ตัว
- เขียน code อย่างไร

```
for i in range(n):
    print('x', end="")
```

หมายเหตุ หากต้องการ print โดยที่ไม่ขึ้นบรรทัดใหม่ให้เพิ่ม ,end="" ต่อท้ายข้อความ

เราจะวนทำ n บรรทัด

คิดว่าเป็นตัวต่อ เหมือนใน Lab code.org

วนทำ 1 รอบต้องทำอะไรบ้าง

1 พิมพ์ x จำนวน n ตัว

2 จากนั้นขึ้นบรรทัดใหม่ (ทำสองอย่าง)

```
for j in range(n):
    for i in range(n):
        print('x', end="")
    print()
```

- เมื่อรวม code ทั้งหมด

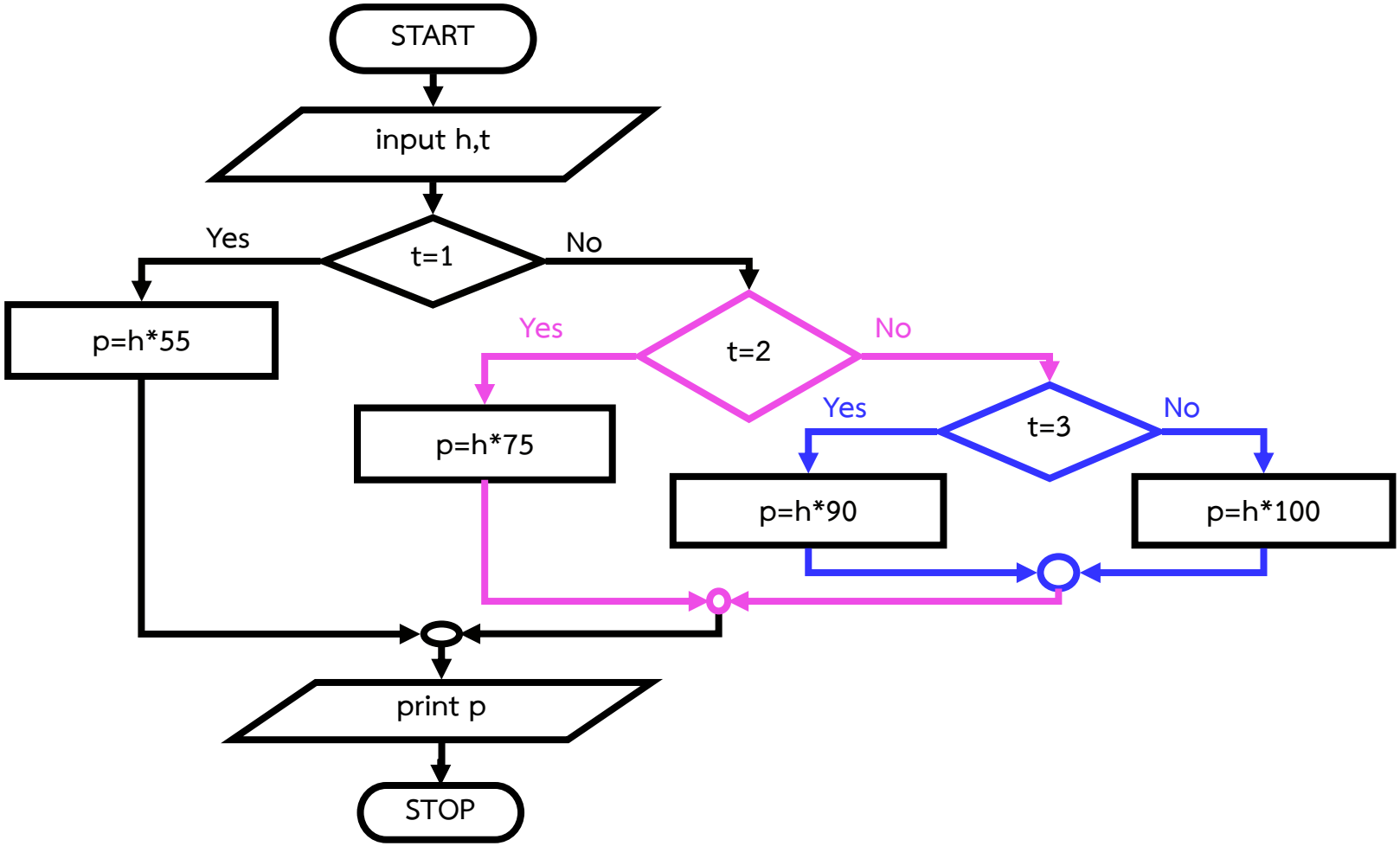
```
n = int(input(''))
for j in range(n):
    for i in range(n):
        print('x', end="")
    print()
```

```
3
xxx
xxx
xxx
```

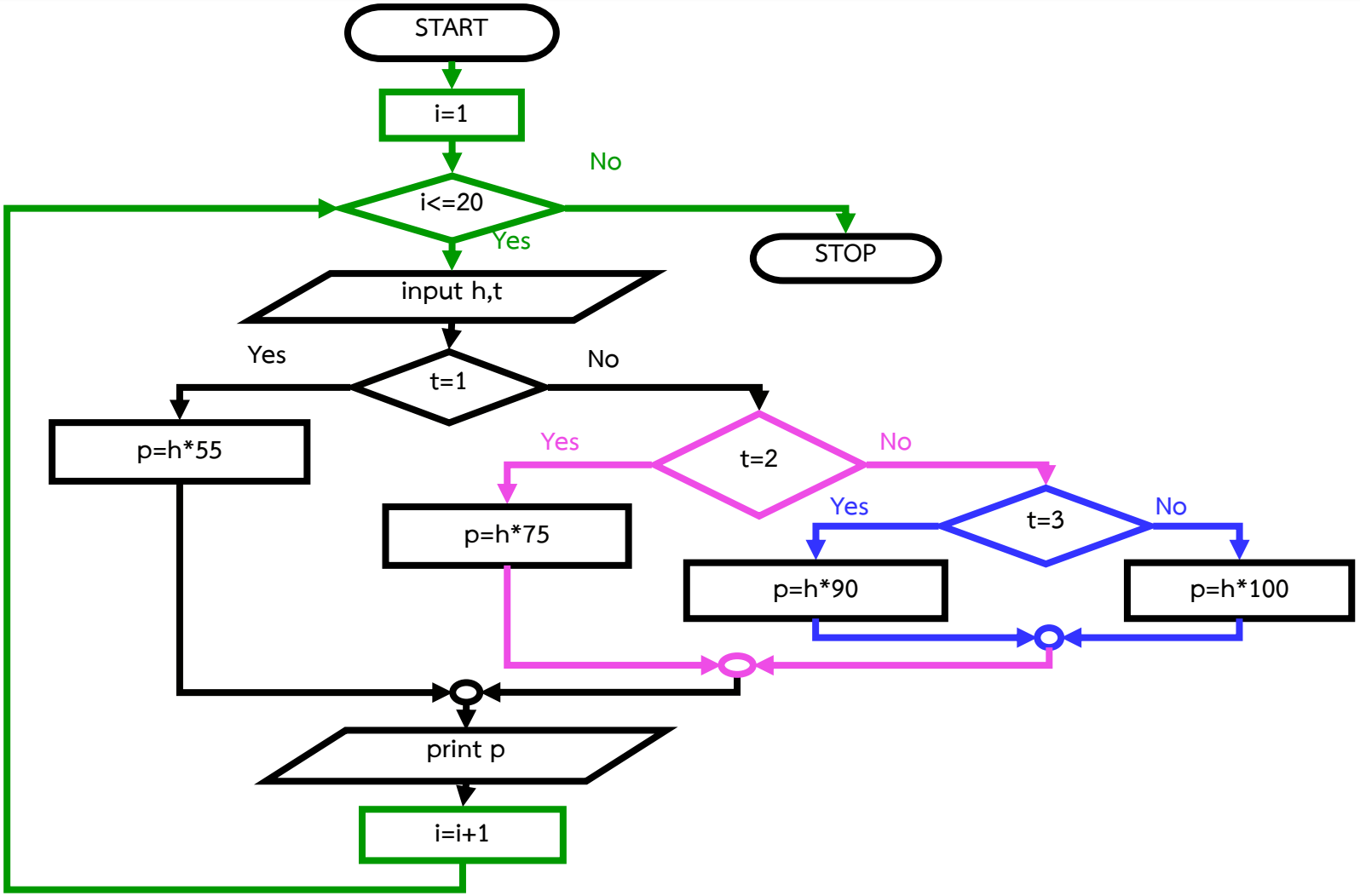
จงเขียนผังงานและโปรแกรมเพื่อคำนวณเงินค่าจ้างของพนักงานจำนวน 20 คน โดยกำหนดให้รับข้อมูลจำนวนชั่วโมงทำงานและข้อมูลประเภทของพนักงาน ซึ่งพนักงานแต่ละประเภทได้รับค่าจ้างต่อชั่วโมงต่างกัน ดังนี้

ประเภทของพนักงาน	ค่าจ้างบาท / ชั่วโมง
1	55
2	75
3	90
4	100

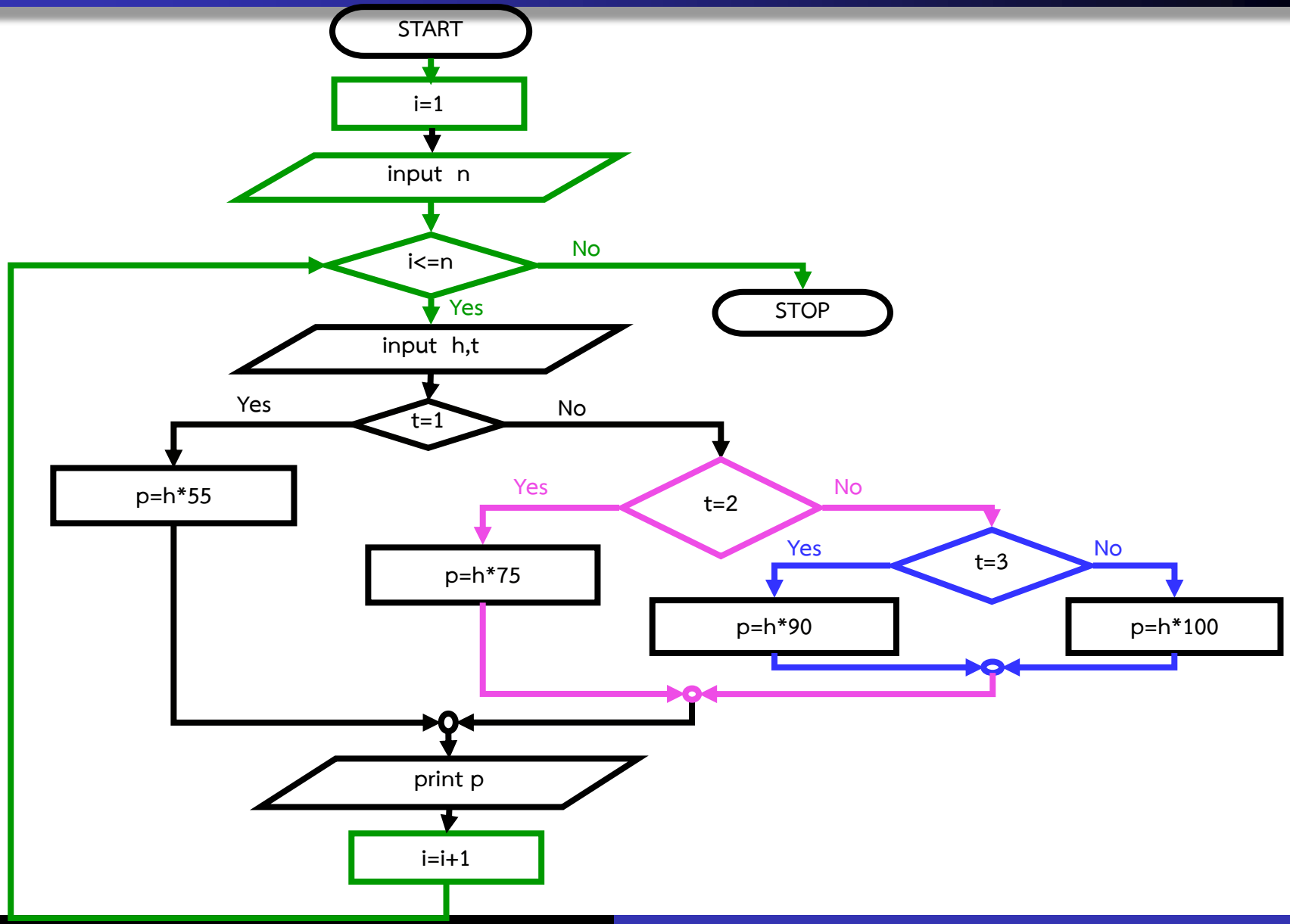
เมื่อคิดค่าจ้างของลูกจ้างคนเดียว



เมื่อคิดค่าจ้างของลูกจ้าง 20 คน



เมื่อคิดค่าจ้างของลูกจ้าง N คน :Flowchart



ค่าจ้างของลูกจ้าง N คน :Program

```
i=1
n = int(input('N =  '))
while i <= n:
    h = int(input('H =  '))
    t = int(input('T =  '))
    if t==1 :
        p=h*55
    elif t==2 :
        p=h*75
    elif t==3 :
        p=h*90
    else:
        p=h*100
    print(p)
    i=i+1
```

หาค่า max

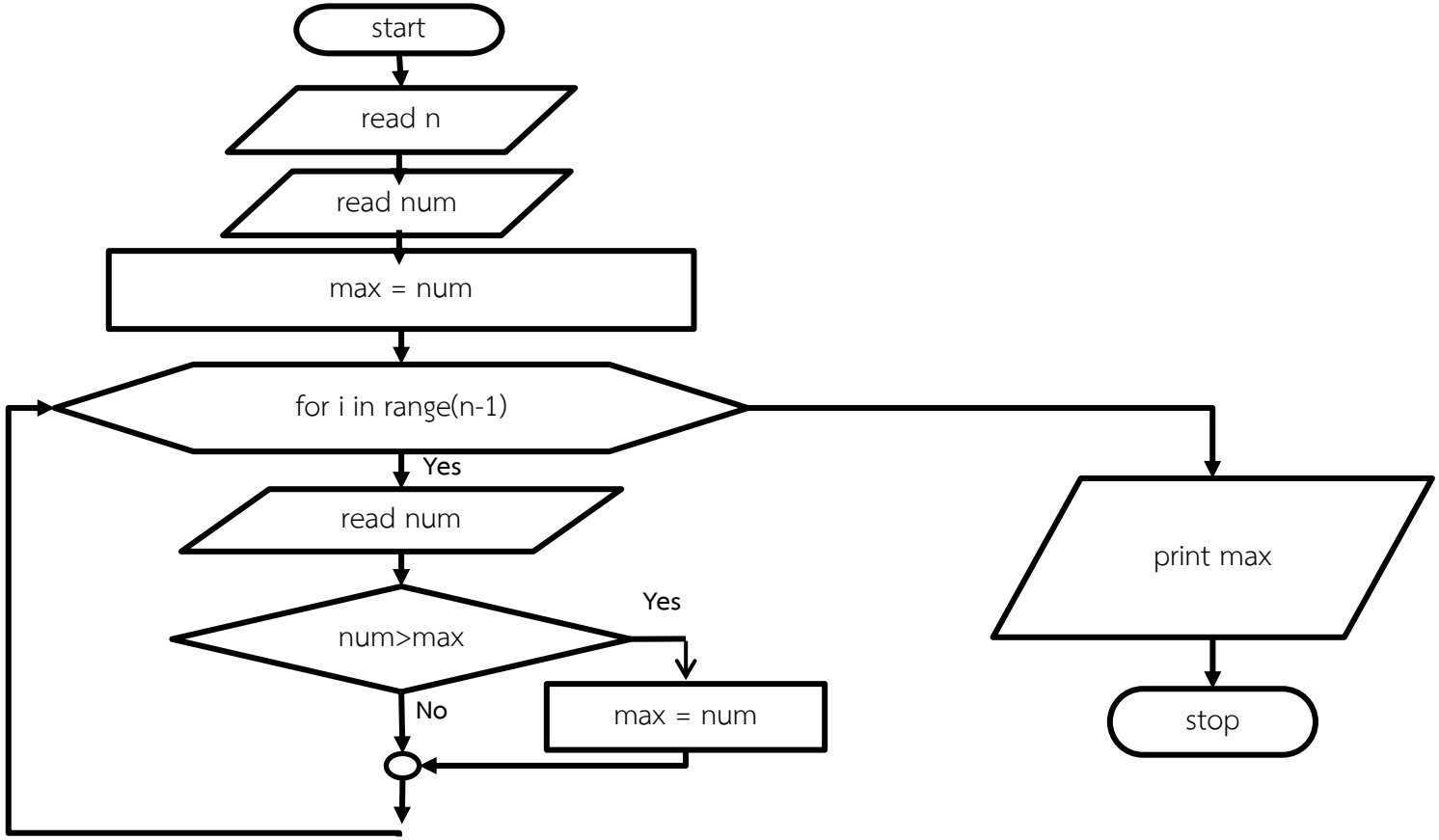
- รับข้อมูลจำนวนเต็ม 1 ตัว (n) จากนั้นรับค่าเลขจำนวนจริง n ครั้ง แล้วหาค่า max ของเลขเหล่านี้
- ตัวอย่างเช่น

รับค่า
3
7
2
5

ผลลัพธ์
max = 7

flowchart

flowchart ของปัญหานี้ควรมีหน้าตาเป็นอย่างไร



คำถาม

- ทำไมเราต้องกำหนดให้ค่าแรกที่รับมาเป็น max = num
- ทำไมใน for loop เราวนค่า range(n-1)

เมื่อรวม code ทั้งหมด

```
n = int(input('enter n '))
num = float(input('enter no. '))
max=num
for i in range(n-1):
    num = float(input('enter no.'))
    if num>max:
        max=num
print("max = %.2f"%max)
```