Unit 9.

Loop-2

Learning objectives

- ✓ ສາມາດກຳນົດຈຳນວນຮອບທີ່ຈະວົນຊ້ຳ ແລະ ໃຊ້ຄຳສັ່ງເພື່ອຂຽນປະໂຫຍກທີ່ວົນຊ້ຳ ຕາມເງື່ອນໄຂ
- ✓ ສາມາດ import random number module ເຂົ້າມາ ເພື່ອນຳໃຊ້ຟັງຊັນ random ຕົວ ເລກຕ່າງໆພາຍໃນໄດ້.
- ✓ ສາມາດຢຸດ ຫຼື ຄວບຄຸມຄຳສັ່ງວົນຊ້ຳ ໂດຍນຳໃຊ້ break ແລະ continue.
- ✓ ສາມາດແກ້ໄຂບັນຫາທີ່ຫຼາກຫຼາຍ ແລະ ສັບຊ້ອນກວ່າບັນຫາທີ່ຜ່ານມາ ໂດຍໃຊ້ double loop.

Lesson overview

- ✓ ຮຽນຮູ້ຄວາມຈຳເປັນຂອງການນຳໃຊ້ຄຳສັ່ງ while ແລະ ວິທີການນຳໃຊ້ຄຳສັ່ງດັ່ງກ່າວ.
- ✓ ຮຽນຮູ້ວິທີສ້າງຕົວເລກແບບສຸ່ມ ແລະ ນຳຄ່າໄປໃຊ້ກັບໂປຣແກຣມຕ່າງໆ.
- ✓ ຮຽນຮູ້ວິທີການກຳນົດເງື່ອນໄຂຕ່າງໆໃຫ້ກັບ while loop.
- ✓ ຮຽນຮູ້ວິທີການຂຽນໂປຣແກຣມທີ່ດຳເນີນການຄູນຊ້ຳ, ໂດຍໃຊ້ double statement.

Concepts You Will Need to Know From Previous Units

- ✓ ແນວຄວາມຄິດຂອງຄຳສັ່ງວົນຊ້ຳ ແລະ ສາມາດນຳມາໃຊ້ໃນຄຳສັ່ງທີ່ສັບຊ້ອນ.
- ✓ ສາມາດຂຽນໂປຣແກຣມໂດຍໃຊ້ for ໃນ expression ແລະ list.
- ✓ ຄວາມແຕກຕ່າງລະຫວ່າງຄຳສັ່ງແບບລຳດັບ, ຄຳສັ່ງແບບມີເງື່ອນໄຂ ແລະ ຄຳສັ່ງແບບວົນຊ້ຳ

Keywords

- **while**
- **random**
- **import command**
- **break, continue**
- **double loop**

Mission

1. Real world problem

1.1. Metaverse fever, meta

https://www.bestchoisinvest.com/post/_rblx

- Metaverse ຫຼື ໂລກ virtual ຂະຫຍາຍເປັນຄຳສັບໃໝ່ທີ່ລວມຄຳ ວ່າ "meta" ກັບ “Verse”, ເຊິ່ງໝາຍຄວາມວ່າ virtual ເໜືອທຳ ມະຊາດ, ແລະ "ຈັກກະວານ," ເຊິ່ງເຮັດໃຫ້ໂລກ ແລະ ຈັກກະວານ ສວຍງາມ.
- ໂດຍທົ່ວໄປຈະຖືກນຳໃຊ້ເປັນຄວາມໝາຍຂອງ 'ພື້ນທີ່ virtual ສາມ ມິຕິ, ເຊິ່ງກິດຈະກຳທາງສັງຄົມ ແລະ ເສດຖະກິດເຊັ່ນ: ໂລກແຫ່ງ ຄວາມເປັນຈິງຖືກນຳໃຊ້ທົ່ວໄປ'
- Roblox, ເຊິ່ງເປັນການບໍລິການ metaverse ທີ່ໃຫຍ່ທີ່ສຸດໃນ ໂລກ, ມີຜູ້ໃຊ້ລາຍວັນ (DAUS) 42.1 ລ້ານຄົນໃນໄຕມາດທຳອິດ ຂອງປີ 2021 ແລະ ພວກເຂົາເຈົ້າຢູ່ໃນ platform ຈຳນວນ 9.7 ຕື້ ຊົ່ວໂມງ. ໃນຄວາມເປັນຈິງ, ບາງຄົນຈະໃຊ້ເວລາຫຼາຍໃນ Roblox, ພື້ນທີ່ virtual, ຫຼາຍກ່ວາໃຊ້ເວລາກັບໝູ່ເພື່ອນ ຫຼື ຄອບຄົວ.
- ບາງຄົນຄາດຄະເນວ່າ metaverses ເຫຼົ່ານີ້ຈະເປັນຮຸ່ນຕໍ່ໄປຂອງອິນ ເຕີເນັດ ແລະ ຜູ້ຄົນຈະເພີດເພີນກັບການຊື້, ການເຮັດວຽກ ແລະ ການ ພັກຜ່ອນໃນພື້ນທີ່ຂອງ metaverse ໃນອະນາຄົດ.

1. Real world problem

1.2. ຫອຍທາກຕົກຢູ່ໃນນ້ຳສ້າງ

- ພວກເຮົາຕ້ອງການຈຳລອງການເຄື່ອນໄຫວຂອງຫອຍທາກ, ສິ່ງທີ່ມີຊີວິດໃນ metaverse ໄດ້.
- ສົມມຸດວ່າຫອຍທາກໂຕນີ້ຕົກລົງນ້ຳສ້າງໃນໂລກ metaverse. ເຖິງຢ່າງໃດກໍຕາມ, ຫອຍທາກຕ້ອງການອອກຈາກນ້ຳສ້າງ ແລະ ກັບໄປສູ່ໂລກກວ້າງ.
- ນ້ຳສ້າງນີ້ຂ້ອນຂ້າງຄ້າຍຄືກັນກັບຄວາມເປັນຈິງ ແລະ ຢູ່ໃນສະພາບທີ່ໜຶ່ນ. ສະນັ້ນ, ສົມມຸດວ່າມັນເປັນນ້ຳສ້າງທີ່ຢູ່ນິ້ງໆ.
- ຫອຍທາກມີຄວາມພະຍາຍາມຫຼາຍທີ່ຈະຂຶ້ນຈາກນ້ຳສ້າງທຸກຄັ້ງ, ແຕ່ມັນກໍຕະລຸດລົງມາ ແລະ ໄຕ່ຂຶ້ນມາໃໝ່.
- ໃນພາລະກິດນີ້, ພວກເຮົາຈະສ້າງໂປຣແກຣມເພື່ອຊອກຫາເວລາທີ່ຫອຍທາກຈະສາມາດຂຶ້ນຈາກນ້ຳສ້າງນີ້ໄດ້.

1. Real world problem

1.3. ຫອຍທາກໃນວິດີໂອອ້າງອີງ

Check https://www.youtube.com/watch?v=dRNYQ0zA9gM for additional information.

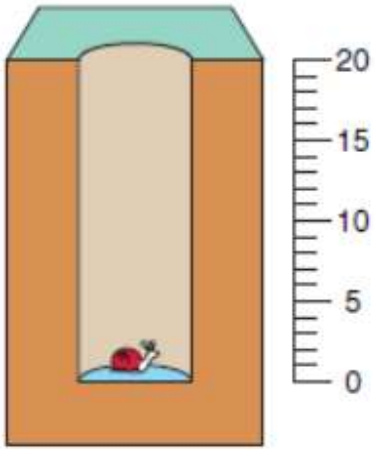
How many days to get out?

ໃຊ້ເວລາຈັກມື້ຈຶ່ງຈະອອກໄດ້?

https://www.youtube.com/watch?v=dRNYQ0zA9gM

2. Solution

2.1. ຫອຍທາກຢູ່ໃນນ້ຳສ້າງ



https://puzzlefry.com/puzzles/snail-in-well-riddle/

- ສົມມຸດວ່າມີນ້ຳສ້າງທີ່ມີຄວາມເລິກ 30 ແມັດ ແລະ ຫອຍທາກຢູ່ພື້ນຂອງນ້ຳສ້າງນີ້. ຫອຍທາກທີ່ອາໄສຢູ່ໃນນ້ຳສ້າງນີ້ ສາມາດປີນຂຶ້ນໄດ້ເຖິງ 7 ແມັດ, ຖ້າມັນໄຕ່ຂຶ້ນໄປຕະຫຼອດມື້. ເຖິງຢ່າງໃດກໍຕາມ, ໃນຂະນະທີ່ມັນກຳລັງພັກຜ່ອນໃນຕອນກາງຄືນ, ມັນຈະຕະລຸດລົງ 5 ແມັດ. ຖາມວ່າມັນຈະໃຊ້ເວລາຈັກມື້ຈຶ່ງຈະອອກຈາກນ້ຳສ້າງເລິກ 30 ແມັດນີ້ໄດ້?
- ຫຼັງຈາກໄຕ່ຂຶ້ນໄປ 7 ແມັດຕໍ່ມື້ ແລະ ຕະລຸດລົງ 5 ແມັດ, ມັນຈະສິ້ນສຸດການເຄື່ອນຍ້າຍຂຶ້ນທີ່ 2 ແມັດ, ແຕ່ມັນເປັນບັນຫາທີ່ບໍ່ຄວນຄິດວ່າມັນຈະງ່າຍໆ ວ່າຈະໃຊ້ເວລາ 15 ມື້ ເພື່ອເຄື່ອນຍ້າຍໄດ້ 30 ແມັດ. ຄຳຕອບແມ່ນ 13 ມື້, ແຕ່ມັນຈະໄດ້ຮັບການຢືນຢັນໂດຍຜ່ານການ coding.
- ດັ່ງນັ້ນ, ເພື່ອແກ້ໄຂບັນຫານີ້, ພວກເຮົາຈະນຳໃຊ້ຄຳສັ່ງໃໝ່ທີ່ເອີ້ນວ່າ while, ບໍ່ໃຊ້ຄຳສັ່ງ for ໃນພາລະກິດນີ້, ພວກເຮົາຈະນຳໃຊ້ຄຳສັ່ງ while ເພື່ອຊອກຫາຕຳແໜ່ງຂອງຫອຍທາກທຸກໆຕອນແລງ ແລະ ຄິດໄລ່ຈຳນວນມື້ທີ່ມັນຈະອອກຈາກນ້ຳສ້າງເລິກ 30 ແມັດ.

3. Mission

3.1. ວິທີການເຮັດວຽກຂອງຫອຍທາກໃນນ້ຳສ້າງ

```
current_pos = 0  # Current position of snail
days = 1         # date
move_up = 7      # distance it goes up a day : in meters
move_down = 5    # distance it goes down while sleep : in meters
destination = 30 # Distance goal : in meters
current_pos += move_up  # position of the snail on the first day

print('day :',days, 'Snail's position :', current_pos, 'meters')
while( current_pos <= destination):   # repeat if snail hasn't reached the distance goal
    current_pos -= move_down          # goes down again
    days += 1                         # next day
    current_pos += move_up            # distance it went up
    print('day :' ,days, 'snail's position :', current_pos, 'meters')
print()
print('It took {} days to get out of the well.'.format(days))
```

```
day : 1   Snail's position : 7 meters
day : 2   Snail's position : 9 meters
day : 3   Snail's position : 11 meters
day : 4   Snail's position : 13 meters
day : 5   Snail's position : 15 meters
day : 6   Snail's position : 17 meters
day : 7   Snail's position : 19 meters
day : 8   Snail's position : 21 meters
day : 9   Snail's position : 23 meters
day : 10   Snail's position : 25 meters
day : 11   Snail's position : 27 meters
day : 12   Snail's position : 29 meters
day : 13   Snail's position : 31 meters
```

ຫອຍທາກໃຊ້ເວລາ 13 ມື້ເພື່ອອອກຈາກນ້ຳສ້າງ

3. Mission

3.2. Programming Plan

Pseudocode

[1] Start

[2] Initialize the location, date, target location, daily travel distance, etc. of snail

[3] while If the snail's position is smaller than the target position

[4] goes down while sleep

[5] Next day. Increase the day every time

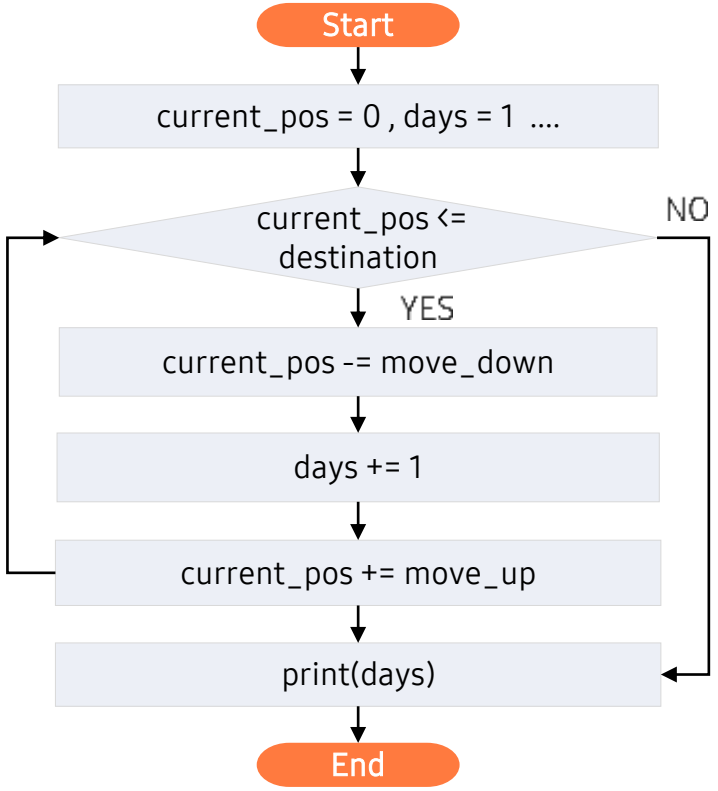
[6] Goes up again..

end while

[7] Print the day it took to escape.

[8] End

Flowchart



3. Mission

3.3. code ที่ดีสุดกับงานขึ้นจากน้ำสร้างของหอยทาก

```
current_pos = 0  # Current position of snail
days = 1         # date
move_up = 7      # distance it goes up a day : in meters
move_down = 5    # distance it goes down while sleep : in meters
destination = 30 # Distance goal : in meters
current_pos += move_up  # position of the snail on the first day

print('day :',days, 'Snail's position :', current_pos, 'meters')
while( current_pos <= destination):   # repeat if snail hasn't reached the distance goal
    current_pos -= move_down          # goes down again
    days += 1                         # next day
    current_pos += move_up            # distance it went up
    print('day :' ,days, 'snail's position :', current_pos, 'meters')
print()
print('It took {} days to get out of the well.'.format(days))
```

Key concept

1. ໂຄງສ້າງພື້ນຖານຂອງຄຳສັ່ງ while

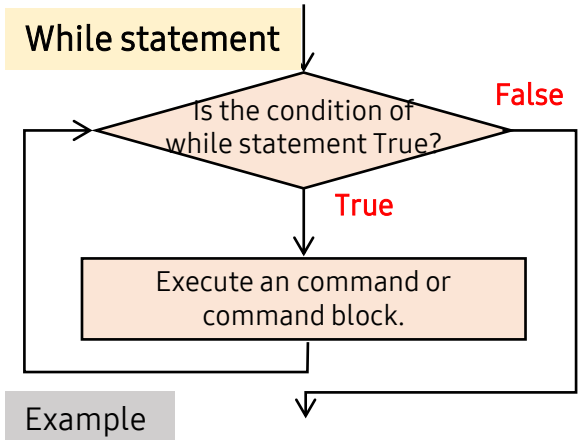
1.1. ຄຳສັ່ງ while ແມ່ນຫຍັງ?

- ຄຳສັ່ງ while ແມ່ນການປະຕິບັດວົນຊ້ຳຄືກັບຄຳສັ່ງ for.
- ຄຳສັ່ງ for ຈະສິ້ນສຸດລົງຫຼັງຈາກລົບຈຳນວນລຳດັບທັງໝົດໃນ keyword
- ຄຳສັ່ງ while ຈະປະຕິບັດຊ້ຳຄຳສັ່ງກໍຕໍ່ເມື່ອເງື່ອນໄຂໃດໜຶ່ງເປັນຈິງ.
  - ມັນຢູ່ໃນຮູບແບບຂອງ *while [Conditional expression]:.*
  - ຖ້າຂຽນໃນຂະນະທີ່ ເປັນຈິງ:, ມັນຈະເຂົ້າໃນວົງ loop ທີ່ບໍ່ສິ້ນສຸດ.
- ຖ້າເງື່ອນໄຂຍັງສືບຕໍ່ເປັນຈິງ, loop ນີ້ອາດຈະບໍ່ສິ້ນສຸດ, ສະນັ້ນຕ້ອງລະມັດລະວັງໃນເວລາຂຽນ code.

1. ໂຄງສ້າງພື້ນຖານຂອງຄຳສັ່ງ while

1.1. ຄຳສັ່ງ while ແມ່ນຫຍັງ?

- ໃນ unit ທີ່ຜ່ານມາ, ພວກເຮົາໄດ້ພິມ 'Welcome' ຫ້າຄັ້ງໂດຍໃຊ້ຄຳສັ່ງ for. ຕອນນີ້ໃຫ້ເຮົາມາເບິ່ງວິທີການໃຊ້ຄຳສັ່ງ while.
- ຄວາມໝາຍຂອງແຜນຜັງ ແລະ code ນີ້ແມ່ນໄດ້ສະແດງໄວ້ດັ່ງລຸ່ມນີ້



Example

```
i = 0
while i < 5:
   print('Welcome')
   i = i + 1
```

- ເລີ່ມຕົ້ນ i ເປັນສູນ.
- ຄຳສັ່ງ while ພິມຄຳວ່າ 'Welcome' ແລະ ເພີ່ມຄ່າ i ຂຶ້ນ 1 ຄ່າ, ເງື່ອນໄຂຂອງ i < 5 ມີຄ່າຄວາມຈິງເປັນ True.
- ເມື່ອງຈາກຄ່າ i ເບື້ອງຕົ້ນແມ່ນ 0, ການວົນຊ້ຳນີ້ຈະຖືກປະຕິບັດຫ້າເທື່ອ.

1. ໂຄງສ້າງພື້ນຖານຂອງຄຳສັ່ງ while

1.2. Syntax ຂອງຄຳສັ່ງ while

Form	Example

set initial value
while conditional statement :
    code block to be executed
	i = 0 # Set initial value
	while i < 5 : # conditional expression
	    print('Welcome to everyone!!')
	    i += 1

- ເຖິງແມ່ນວ່າຈະເປັນຄຳສັ່ງວົນຊ້ຳ, ແຕ່ກໍຄ້າຍຄືກັນກັບຄຳສັ່ງ if ຫຼາຍ
- ຖ້າຫາກເງື່ອນໄຂເປັນຈິງ, ມັນຈະວົນຊ້ຳຄືນ ແລະ ປະຕິບັດງານຕາມ code block

1. โครงสร้างพื้นฐานของคำสั่ง while

1.2. Syntax ของคำสั่ง while

```
1 i = 0  # initial value
2 while i < 5:    # block is executed when loop's condition is true
3     print('Welcome to everyone!!')
4     i += 1
```

```
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
Welcome to everyone!!
```

- เพิ่ม i จาก 0 เป็น 1 และ อินซ้ำ code พายใน Block, ในขะนะที่มีค่าน้อยกว่า 5 เท่านั้น.

1. ໂຄງສ້າງພື້ນຖານຂອງຄຳສັ່ງ while

1.3. ໂປຣແກຣມຊອກຫາຜົນບວກຂອງຄ່າ ຈົນເຖິງຄ່າທີ່ປ້ອນເຂົ້າ

ໂປຣແກຣມທີ່ໃຊ້ຄຳສັ່ງ while ໃນຂະນະທີ່ຊອກຫາຜົນບວກຂອງຄ່າ ຈົນເຖິງຄ່າທີ່ປ້ອນຂໍ້ມູນເຂົ້າໄປ.

```
1 n = int(input('Enter a number to sum up to : '))
2 s = 0
3 i = 1
4 while i <= n:
5     s = s + i
6     i += 1
7 print('Sum of 1 to {} is {}'.format(n, s))
```

```
Enter a number to sum up to : 5
Sum of 1 to 5 is 15
```

- code ຂ້າງເທິງໄດ້ຮັບເລກ 5 ເປັນການປ້ອນຂໍ້ມູນເຂົ້າ ແລະ ຄິດໄລ່ຜົນບວກ, ພ້ອມກັບພິມຜົນບວກຂອງຈຳນວນຖ້ວນແຕ່ 1 ຫາ 5.
- ດ້ວຍວິທີນີ້, ຖ້າຮູ້ຈຳນວນການວົນຊ້ຳຄືນທີ່ສັດເຈນ, ຄວນຈະໃຊ້ຄຳສັ່ງ for ເພາະວ່າ code ຂອງຄຳສັ່ງ while ຈະຍາວກວ່າ.
- ດັ່ງນັ້ນເມື່ອໃດພວກເຮົາຄວນໃຊ້ຄຳສັ່ງ while?

1. ໂຄງສ້າງພື້ນຖານຂອງຄຳສັ່ງ while

1.4. ການປຽບທຽບຄຳສັ່ງ while ແລະ ຄຳສັ່ງ for

ການປຽບທຽບຄຳສັ່ງ while ແລະ ຄຳສັ່ງ for ດ້ວຍ code ຕົວຢ່າງ

for

```
s = 0
for i in range(1, 11):
    s += i
print('Sum of 1 to 10:'.s)
```

while

```
n = 10
s = 0
i = 1
while i <= n:
    s = s + i
    i += 1
print('Sum of 1 to {} is {}'.format(n, s))
```

- ຄຳສັ່ງ while ຈະເໝາະສົມກວ່າເມື່ອມີເງື່ອນໄຂການປະຕິບັດການສັດເຈນ, ແຕ່ບໍ່ຮູ້ຈຳນວນການວົນຊ້ຳທີ່ແນ່ນອນ.
- ຖ້າຈຳນວນຂອງການວົນຊ້ຳສັດເຈນ, ຄຳສັ່ງ for ຈະເໝາະສົມກວ່າ.

1. ໂຄງສ້າງພື້ນຖານຂອງຄຳສັ່ງ while

1.5. while loop ແລະ ເງື່ອນໄຂການປ້ອນຂໍ້ມູນ

ຮັບພຽງແຕ່ 'scissors', 'rock' ແລະ 'paper' ເປັນວັດຖຸປ້ອນເຂົ້າຈາກຜູ້ໃຊ້ ແລະ ພິມຂໍ້ມູນນັ້ນ. ຖ້າການປ້ອນຂໍ້ມູນນອກເໜືອຈາກ 'scissors', 'rock' ແລະ 'paper' ເຂົ້າມາ, ໃຫ້ຮ້ອງຂໍການປ້ອນຂໍ້ມູນອີກຄັ້ງ.

```
selected = None
while selected not in ['scissors', 'rock', 'paper']:
    selected = input('Choose among scissors, rock, paper> ')
print('Chosen value:', selected)
```

```
Choose among scissors, rock, paper> no
Choose among scissors, rock, paper> why
Choose among scissors, rock, paper> paper
Chosen value: paper
```

- ໃນກໍລະນີຂອງ code ດັ່ງກ່າວ, ການແກ້ບັນຫາດ້ວຍຄຳສັ່ງ for ມັນເປັນໄປໄດ້ຍາກ.
- ຄຳສັ່ງ While ຈະເໝາະສົມສຳລັບຄຳສັ່ງວົນຊ້ຳ ທີ່ຈະຖືກປະຕິບັດການເມື່ອປ້ອນຄ່າທີ່ຕ້ອງການເທົ່ານັ້ນ.

One More Step

- ใน code ที่ก่าวเถิงก่อนหน้านี้, มีสำนวนที่เลือก = None.
- code นี้แม่นได้ฮับกามป้อนข้ำมูนจากแถวที 3 หังจากกวดสอบกามเลือกจากคำสั่ง while ในแถวที 2
- ถ้าบ่ได้กำนิดถ่าให้กับกามเลือก โดยที่ selected = None, code นี้จะเธ็ดให้เกิดถวามผิดพาด.
- เว้าอีกแบบหนึ่ง, None หมายถวามอ่า 'บ่มีถ่า', แต่ด้อยอิทินี้, มันเป็นสิ่งที่สำคัมในกามไห฼ของข้ำมูนในโปธแกธม.

```
1 selected = None
2 while selected not in ['scissors', 'rock', 'paper']:
3     selected = input('Choose among scissors, rock, paper> ')
4 print('Chosen value:', selected)
```

```
Choose among scissors, rock, paper>
```

1. ໂຄງສ້າງພື້ນຖານຂອງຄຳສັ່ງ while

1.5. while loop ແລະ ເງື່ອນໄຂການປ້ອນຂໍ້ມູນ

ໄດ້ຮັບຈຳນວນທຳມະຊາດ n ຈາກຜູ້ໃຊ້ ແລະ ຄິດໄລ່ຜົນບວກ 1 + 2 + ... + n. ເຖິງຢ່າງໃດກໍຕາມ, ຖ້າການປ້ອນຂໍ້ມູນຂອງຜູ້ໃຊ້ແມ່ນ -10, ໂປຣແກຣມມີຄວາມຜິດພາດທາງດ້ານຕັກກະສາດ. ດັ່ງນັ້ນ, ຄ່າທີ່ປ້ອນເຂົ້າໄປຕ້ອງເປັນຄ່າບວກ.

```
1 n = int(input('Enter a number to sum up to :'))
2 s = 0
3 for i in range(1, n+1) :
4     s = s + i
5 print('Sum of 1 to {} is {}'.format(n, s))
```

```
Enter a number to sum up to : -10
Sum of 1 to -10 is 0
```

- code ນີ້ມີຄວາມຜິດພາດທາງດ້ານຕັກກະສາດ.
- ຖ້າຈຳກັດຂອບເຂດຂອງການປ້ອນຄ່າ input ເປັນຈຳນວນທຳມະຊາດຈະດີກວ່າ. ໃນກໍລະນີນີ້, ຄວນໃຊ້ຄຳສັ່ງ while ເພື່ອປະຕິບັດວົນຊ້ຳ ຕາມເງື່ອນໄຂດີກ່ວາການໃຊ້ຄຳສັ່ງ for.

1. ໂຄງສ້າງພື້ນຖານຂອງຄຳສັ່ງ while

1.5. while loop ແລະ ເງື່ອນໄຂການປ້ອນຂໍ້ມູນ

ໄດ້ຮັບຈຳນວນທຳມະຊາດ n ຈາກຜູ້ໃຊ້ ແລະ ຄິດໄລ່ຜົນບວກ 1 + 2 + ... + n. ຄຳສັ່ງ while ຈະເໝາະສົມ ເມື່ອຕ້ອງການຮັບຄ່າ n ທີ່ເປັນຄ່າບວກ.

```
1 n = -1
2 while n <= 0:  # input() repeated until positive value is entered
3     n = int(input('Enter a positive number to sum up to :'))
4 s = 0
5 for i in range(1, n+1):
6     s = s + i
7 print('Sum of 1 to {} is {}'.format(n, s))
8
```

```
Enter a positive number to sum up to : -1
Enter a positive number to sum up to : 0
Enter a positive number to sum up to : 7
Sum of 1 to 7 is 28
```

Line 2

- ສິ່ງສຳຄັນແມ່ນການໄດ້ຮັບການປ້ອນຂໍ້ມູນຄືນໃໝ່ ຖ້າຖືກເງື່ອນໄຂ n <= 0 ເປັນຈິງ. ຖ້າປ້ອນ -1 ຫຼື 0, ການປ້ອນຂໍ້ມູນຖືກຮ້ອງຂໍໃຫ້ປ້ອນຂໍ້ມູນຄືນໃໝ່ອີກຄັ້ງ

2. ການສຸ່ມ (Random)

2.1. ໂມດູນແບບສຸ່ມ (random module)

- ໃຫ້ພິຈາລະນາຢ່າງລະອຽດກ່ຽວກັບໂມດູນແບບສຸ່ມທີ່ພວກເຮົາໄດ້ຮຽນໃນ unit ທີ່ຜ່ານມາ.
- ກ່ອນອື່ນໝົດ, ໃຫ້ເບິ່ງແນວຄວາມຄິດ ແລະ ຫຼັກໄວຍາກອນຂອງໂມດູນ.
- ໂມດູນໄດ້ອ້າງອີງເຖິງໄຟລ໌ສະຄຣິບທີ່ລວບລວມຟັງຊັນ, ຕົວປ່ຽນ ຫຼື classes ໃນພາສາ Python.
  - Python ມີຫຼາຍໂມດູນທີ່ສ້າງຂຶ້ນໂດຍອົງກອນ ແລະ ນັກພັດທະນາຫຼາຍຄົນ, ເຮັດໃຫ້ສາມາດພັດທະນາຊອບແວໄດ້ຢ່າງມີປະສິດທິພາບໂດຍໃຊ້ ໂມດູນເຫຼົ່ານີ້.
  - ໂມດູນທີ່ພັດທະນາດ້ວຍວິທີນີ້ແມ່ນມາພ້ອມກັບໄຟລ໌ການຕິດຕັ້ງ Python ພາຍໃຕ້ຊື່ຂອງ standard library.
- ເມື່ອນຳເອົາໂມດູນທີ່ສຳເລັດຮູບເຂົ້າມາ, ໃຫ້ຂຽນຊື່ໂມດູນດ້ວຍ "import" ແລະ ເມື່ອນຳໃຊ້ມັນ ໃຫ້ໃຊ້ເຄື່ອງໝາຍຈ້ຳເມັດ (.) ຕາມຫຼັງຊື່ໂມດູນ ແລະ ຕາມດ້ວຍສ່ວນປະກອບໃນໂມດູນ.

```
import module name1 [, module name2, ...]
```

2. ການສຸ່ມ (Random)

2.2. ລາຍລະອຽດຄຸນສົມບັດຂອງໂມດູນ Random

- ໂມດູນແບບສຸ່ມປະກອບດ້ວຍຟັງຊັນໃນການສ້າງຕົວເລກທີ່ມັກ ຫຼື ການປະສົມ ຫຼື ເລືອກອົງປະກອບໃນ list.
- ເພື່ອຄວາມສະດວກ, ຈະນຳໃຊ້ໂມດູນແບບສຸ່ມດ້ວຍຕົວຫຍໍ້ rd. ໃຊ້ຄຳສັ່ງ as ດັ່ງຕໍ່ໄປນີ້.

```
import [module name] as [abbreviation]
```

- ໜ້າທີ່ ແລະ ພາລະກິດຂອງໂມດູນ Random ມີດັ່ງຕໍ່ໄປນີ້

function	tasks
random()	ສ້າງຕົວເລກຈຳນວນຈິງລະຫວ່າງ 0 ແລະ 1. (ບໍ່ລວມເອົາ 1)
randrange()	ໃຫ້ຜົນຮັບເປັນຈຳນວນຖ້ວນພາຍໃນຂອບເຂດທີ່ລະບຸ.
randint(a, b)	ໃຫ້ຜົນຮັບເປັນຈຳນວນຖ້ວນແບບສຸ່ມ N ລະຫວ່າງ a <= N <=b.
shuffle(seq)	ປະສົມອົງປະກອບແບບສຸ່ມຈາກລາຍການ seq ທີ່ກຳນົດ.
choice(seq)	ເລືອກອົງປະກອບໃນລຳດັບໃດກໍໄດ້ຈາກ seq.
sample()	ສຸ່ມເລືອກອົງປະກອບຕາມຈຳນວນທີ່ລະບຸ.

2. ການສຸ່ມ (Random)

2.3. ຟັງຊັນທີ່ລວມຢູ່ໃນໂມດູນແບບສຸ່ມ

```
1 import random as rd      # import command that calls inner module and abbreviation of random module, rd
2
3 print(rd.random())           # Returns a random real
4 print(rd.random())           # Returns a random real
5 print(rd.randrange(1, 7))    # Returns an integer greater than or equal to 1 and less than 7.
6 print(rd.randrange(0, 10, 2))# Returns a multiple of 2 of integers greater than or equal to 0 and less than 10.
7 print(rd.randint(1, 10))     # Returns an arbitrary integer greater than or equal to 1 and less than
                               or equal to 10 (including10).
```

```
0.4753236338712161
0.4515339413828623
5
2
1
```

- ໃຫ້ຜົນຮັບເປັນຈຳນວນຈິງທີ່ໃຫຍ່ກວ່າ ຫຼື ເທົ່າກັບ 0 ແລະ ນ້ອຍກວ່າ 1 ເຊິ່ງໃຫ້ຜົນຮັບເປັນຈຳນວນຈິງທີ່ແຕກຕ່າງກັນໃນແຕ່ລະຄັ້ງ ທີ່ໃຊ້ . random()
- randrange 1 ແລະ 7: ສິ່ງກັບຄ່າຈຳນວນຖ້ວນທີ່ໃຫຍ່ກວ່າ ຫຼື ເທົ່າກັບ 1 ແລະ ນ້ອຍກວ່າ 7
- randrange (0, 10, 2): ສິ່ງກັບຜົນຄູນຂອງ 2 ຈຳນວນຖ້ວນທີ່ໃຫຍ່ກວ່າ ຫຼື ເທົ່າກັບ 0 ແລະ ນ້ອຍກວ່າ 10
- randint(1, 10): ສິ່ງກັບຄ່າຈຳນວນຖ້ວນທີ່ໃຫຍ່ກວ່າ ຫຼື ເທົ່າກັບ 1 ແລະ ນ້ອຍກວ່າ ຫຼື ເທົ່າກັບ 10 (ລວມທັງ 1 ແລະ 10).

2. ການສຸ່ມ (Random)

2.3. ຟັງຊັນທີ່ລວມຢູ່ໃນໂມດູນແບບສຸ່ມ

```
1 numlist = [10, 20, 30, 40, 50]
2 rd.shuffle(numlist)
3 print(numlist)
```

```
[30, 10, 20, 40, 50]
```

```
1 print(rd.choice(numlist))
```

```
40
```

```
1 print(rd.sample(numlist, 3))
```

```
[20, 30, 10]
```

- shuffle() : ສຸ່ມປະສົມອົງປະກອບຂອງລຳດັບແບບສຸ່ມ ແລະ ຄືນຜົນຮັບກັບອອກມາ
- choice(): ແຍກອົງປະກອບໃດໜຶ່ງອອກຈາກລຳດັບທີ່ປ້ອນເງື່ອນໄຂໃສ່
- sample() : ສຸ່ມ ແລະ ສົ່ງອົງປະກອບກັບຄືນ. ໃນເວລານີ້, ຈຳນວນອົງປະກອບທີ່ຈະສົ່ງຄືນສາມາດເພີ່ມຈຳນວນໄດ້.

2. ภาษสุ่ม (Random)

2.3. ฟังชันที่ลอมยู่ในโมดูมแบบสุ่ม

```
1 a = list(range(1, 11)) # returns integers from 1 to 10
2 rd.shuffle(a)          # randomly mix elements of list
3 print(a)
```

```
[9, 5, 3, 2, 8, 10, 7, 4, 1, 6]
```

- shuffle() จะสิ่งถ่าลำดับออกมา, เซิ่งลำดับจะแตกต่างกันในแต่ละถั้ง.

2. ภาษสุ่ม (Random)

2.4. sample function

พิมสามต๊อเลภจำนอมถ้อมละหว่าๆ 1 แล 10 โดยใช้ฟังชัม sample

```
1 import random as rd
2
3 a = list(range(1, 11))
4 b = rd.sample(a, 3)
5 print('Arbitrary integer between 1 to 10 :', b)
```

```
Arbitrary integer between 1 to 10 : [2, 5, 10]
```

Line 3 ~ 4

- 3: a ถืภกำมิดด้อย list ที่มิอิงปะภอบ 1 หา 10
- 4: สุ่มต๊อย่างจำนอมถ้อมตามใจ 3 จำนอม โดยใช้ฟังชัม sample และ ใส่ถ่าเภับไว้ ในต๊อป่ฐม b.

3. break, continue

3.1. Keyword ที่นำใช้ถอบถุมถำสั่งอ๊มซ้ำ

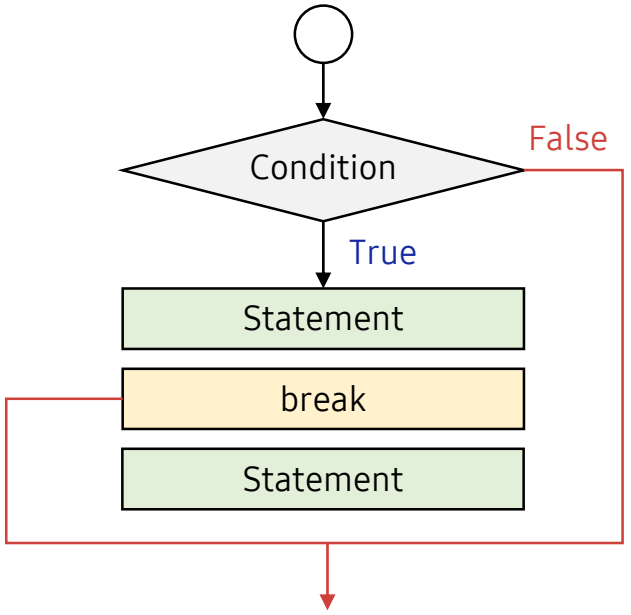
break ແລະ continue ແມ່ນ Keyword ທີ່ໃຊ້ເພື່ອຄວບຄຸມຄຳສັ່ງການວົນຊ້ຳ

- ສິ้ນສຸດການວົນຊ้ำ: break
- ເພື่ອຂ้ามพาภส่อมปะติบัดาามที่ยังเຫຼືอยู่ใม loop ภามอ๊มซ้ำ และ สืบต่ำเมินภามอ๊ม loop ต่ำไป: continue
- continue ดำเมินภามสืบต่ำไปบ่สิ้มสุดใมภามอ๊มซ้ำ

3. break, continue

3.2. ແຜນຜັງຂອງ break

- while ແລະ ຄຳສັ່ງການວົນຊ້ຳ ຈະປະຕິບັດຄຳສັ່ງພາຍໃນ block ຖ້າເງື່ອນໄຂເປັນຈິງ
- ຖ້າຫາກວ່າມັນພົບ break ຢູ່ສ່ວນກາງ ກໍຈະສິ້ນສຸດລົງ ແລະ ອອກຈາກການວົນຊ້ຳທັນທີ



Flowchart of break

3. break, continue

3.3. ตัวอย่างคำสั่ง break

```
1 st = 'Programming'
2 # function of executing only when it is consonant
3 for ch in st:
4     if ch in ['a','e','i','o','u']:
5         break    # end the loop if it is vowel
6     print(ch)
7 print('The end')
```

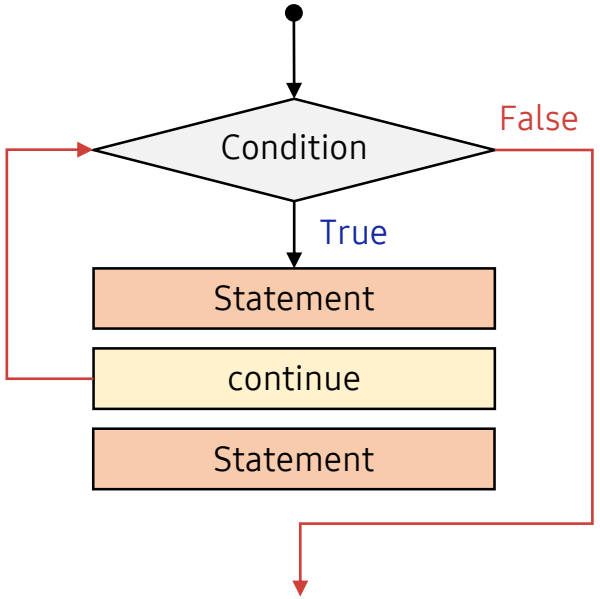
```
P
r
The end
```

- break ถ้าหากว่ามันเป็นสะห฼ะ, ถ้าบ่ดั่งนั้นให้พิมผิมรับออกมา.
- เมื่อ break เธ็ดอฯก, กามอิมซ้ำกำจะยุดโดยบ่มีกามปะติบัดส่อมที่เห฼ือขอฯ loop

3. break, continue

3.4. Flowchart of continue

- ມັນມີບົດບາດໃນການຂ້າມພຽງແຕ່ປະໂຫຍກຂ້າງລຸ່ມນີ້ເທົ່ານັ້ນໂດຍບໍ່ມີການອອກຈາກ loop.
- ການສິ້ນສຸດຂອງ loop ຈະໃຊ້ໄດ້ກໍຕໍ່ເມື່ອເງື່ອນໄຂບໍ່ເປັນຈິງ.



3. break, continue

3.5. ຕົວຢ່າງ ຂອງຄຳສັ່ງ continue

```
1 st = 'Programming'
2 # function of executing only when it is consonant
3 for ch in st:
4     if ch in ['a','e','i','o','u']:
5         continue     # skip the execution below if it is vowel
6     print(ch)
7 print('The end')
```

```
P
r
g
r
m
m
n
g
The end
```

- ເມື່ອຂຽນ continue, ສ່ວນທີ່ເຫຼືອທາງລຸ່ມຈະບໍ່ໄດ້ຖືກປະຕິບັດ
- ເວົ້າອີກຢ່າງໜຶ່ງ, ມັນຈະກັບຄືນໄປສູ່ຈຸດເລີ່ມຕົ້ນຂອງ loop ໂດຍບໍ່ມີການປະຕິບັດຄຳສັ່ງ print ໃນ ແຖວທີ 6

3. break, continue

3.6. ຄວາມສ່ຽງຂອງ break ແລະ continue ການຄວບຄຸມການໄຫຼຂອງຂໍ້ມູນ

- ສາມາດນຳໃຊ້ Break ແລະ continue ຢ່າງສະດວກ ເພື່ອຄວບຄຸມໂປຣແກຣມໃຫ້ມີປະສິດທິພາບ.
- ເຖິງຢ່າງໃດກໍຕາມ, ຖ້າໃຊ້ຄຳສັ່ງ break ແລະ continue ຫຼາຍເກີນໄປ, ມັນຈະເປັນການຍາກທີ່ຈະເຂົ້າໃຈໂປຣແກຣມ, ເນື່ອງຈາກການໄຫຼຂອງການຄວບຄຸມທີ່ບໍ່ສອດຄ່ອງກັນ. ດັ່ງນັ້ນ, ແນະນຳໃຫ້ໃຊ້ continue ແລະ break ເທົ່າທີ່ຈຳເປັນເທົ່ານັ້ນ.

4. Double loop

4.1. Double loop

- Double loop ໝາຍເຖິງໂຄງສ້າງທີ່ loops ອື່ນໆຖືກຝັນຈຸຢູ່ໃນ loop ອີກເທື່ອໜຶ່ງ.
- loop ທີ່ຊ້ອນກັນດ້ານລຸ່ມນີ້ປະກອບດ້ວຍຄຳສັ່ງ for ຊ້ອນຢູ່ໃນຄຳສັ່ງ for..
- code ນີ້ແມ່ນການຄູນ 2x1 ເຖິງ 9x9.
- ເບິ່ງໃນໂຄງສ້າງຂອງສູດຄູນຈະ ມີ 2 ຫາ 9 ຂັ້ນຕອນ, 1 ຫາ 9 ຈະຖືກຄູນໃນແຕ່ລະຂັ້ນຕອນ ແລະ ສະແດງອອກມາທາງໜ້າຈໍ.
- ໃນປະຕິບັດການນີ້, ຈຳເປັນຕ້ອງໃຊ້ຄຳສັ່ງ double for ເພື່ອຂຽນຄຳສັ່ງ for ຢູ່ໃນຄຳສັ່ງ for ອີກຄັ້ງໜຶ່ງ.

```
1 for i in range(2, 10):                # outer for loop
2     for j in range(1, 10):            #inner for loop
3         print('{}x{} = {:2d}, '.format(i, j, i*j), end = ' ')
4     print()       # executes inner loop and change line
```

```
2x1 =  2,  2x2 =  4,  2x3 =  6,  2x4 =  8,  2x5 = 10,  2x6 = 12,  2x7 = 14,  2x8 = 16,  2x9 = 18,
3x1 =  3,  3x2 =  6,  3x3 =  9,  3x4 = 12,  3x5 = 15,  3x6 = 18,  3x7 = 21,  3x8 = 24,  3x9 = 27,
4x1 =  4,  4x2 =  8,  4x3 = 12,  4x4 = 16,  4x5 = 20,  4x6 = 24,  4x7 = 28,  4x8 = 32,  4x9 = 36,
5x1 =  5,  5x2 = 10,  5x3 = 15,  5x4 = 20,  5x5 = 25,  5x6 = 30,  5x7 = 35,  5x8 = 40,  5x9 = 45,
6x1 =  6,  6x2 = 12,  6x3 = 18,  6x4 = 24,  6x5 = 30,  6x6 = 36,  6x7 = 42,  6x8 = 48,  6x9 = 54,
7x1 =  7,  7x2 = 14,  7x3 = 21,  7x4 = 28,  7x5 = 35,  7x6 = 42,  7x7 = 49,  7x8 = 56,  7x9 = 63,
8x1 =  8,  8x2 = 16,  8x3 = 24,  8x4 = 32,  8x5 = 40,  8x6 = 48,  8x7 = 56,  8x8 = 64,  8x9 = 72,
9x1 =  9,  9x2 = 18,  9x3 = 27,  9x4 = 36,  9x5 = 45,  9x6 = 54,  9x7 = 63,  9x8 = 72,  9x9 = 81,
```

4. Double loop

4.2. ໂຄງສ້າງຂອງ loop double

Double loop: ເມື່ອ loops ທັບຊ້ອນກັນ, loops ທັງສອງຈະຖືກແບ່ງອອກເປັນເງື່ອນໄຂຕ່າງໆເຊັ່ນ: loop ນອກ ແລະ loop ໃນ.

Outer loop

Inner loop

```
for i in range(2, 10):
    for j in range(1, 10):
        print('{}*{}={:2d}'.format(i, j, i*j), end=' ')
    print()
```

- double for loop ມີ loop ໃນ ແລະ loop ນອກ
- {} ໃນຄຳສັ່ງການພິມຄືຕົວຢູ່ຕຳແໜ່ງທີ່ລະບຸ, ຕຳແໜ່ງສຳລັບຜົນຮັບ, ເຊິ່ງເປັນເນື້ອໃນທີ່ມາຈາກ Unit8.

4. Double loop

4.3. ການປະຕິບັດການຂອງ double loop

When i = 2	When i = 3	When i = 4	...	When i = 9
j = 1 : 2 * 1	j = 1 : 3 * 1	j = 1 : 4 * 1	...	j = 1 : 9 * 1
j = 2 : 2 * 2	j = 2 : 3 * 2	j = 2 : 4 * 2	...	j = 2 : 9 * 2
j = 3 : 2 * 3	j = 3 : 3 * 3	j = 3 : 4 * 3	...	j = 3 : 9 * 3
j = 4 : 2 * 4	j = 4 : 3 * 4	j = 4 : 4 * 4	...	j = 4 : 9 * 4
j = 5 : 2 * 5	j = 5 : 3 * 5	j = 5 : 4 * 5	...	j = 5 : 9 * 5
j = 6 : 2 * 6	j = 6 : 3 * 6	j = 6 : 4 * 6	...	j = 6 : 9 * 6
j = 7 : 2 * 7	j = 7 : 3 * 7	j = 7 : 4 * 7	...	j = 7 : 9 * 7
j = 8 : 2 * 8	j = 8 : 3 * 8	j = 8 : 4 * 8	...	j = 8 : 9 * 8
j = 9 : 2 * 9	j = 9 : 3 * 9	j = 9 : 4 * 9	...	j = 9 : 9 * 9

- i ມີຄ່າແຕ່ 2 ຫາ 9.
- j ພາຍໃນມີຄ່າແຕ່ 1 ຫາ 9.
- ໂຄງສ້າງນີ້ແມ່ນການວົນຊ້ຳກັນ, ເຮັດໃຫ້ການຄິດໄລ່ ແລະ ໄດ້ຜົນຮັບທັງໝົດ 72 ລາຍການ.
- ໃນກໍລະນີຂອງ double for loop ຫຼື triple for loop, ມັນຈະເຮັດໃຫ້ເຂົ້າໃຈ code ໄດ້ຍາກ.
- ດ້ວຍເຫດຜົນນີ້, loops ທີ່ຊ້ອນກັນ ຈຶ່ງ ບໍ່ໃຊ້ໂຄງສ້າງຫຼາຍກ່ວາສາມ loops.

Paper coding

- ✓ ພະຍາຍາມທຳຄວາມເຂົ້າໃຈແນວຄວາມຄິດພື້ນຖານຢ່າງຖີ່ຖ້ວນ ກ່ອນທີ່ຈະກ້າວໄປສູ່ຂັ້ນຕອນຕໍ່ໄປ.
- ✓ ການບໍ່ມີຄວາມເຂົ້າໃຈແນວຄວາມຄິດພື້ນຖານຈະເພີ່ມພາລະໃນການຮຽນຮູ້ຫຼັກສູດນີ້, ເຊິ່ງອາດຈະເຮັດໃຫ້ທ່ານລົ້ມເຫຼວໃນການຮຽນໃນຫຼັກສູດນີ້.
- ✓ ໃນຕອນນີ້ມັນອາດຈະເປັນເລື່ອງຍາກ, ແຕ່ເພື່ອໃຫ້ສຳເລັດໃນຫຼັກສູດນີ້, ຂໍແນະນຳໃຫ້ເຂົ້າໃຈກັບແນວຄວາມຄິດຢ່າງລະອຽດ ແລະ ກ້າວໄປສູ່ຂັ້ນຕອນຕໍ່ໄປ.

Q1. จົ່ງຂຽນໂປຣແກຣມທີ່ພິມການຄູນດ້ວຍ 2 ໂດຍໃຊ້ຄຳສັ່ງ while ດັ່ງຕໍ່ໄປນີ້.

Example Output	2 * 1 = 2 2 * 2 = 4 2 * 3 = 6 2 * 4 = 8 2 * 5 = 10 2 * 6 = 12 2 * 7 = 14 2 * 8 = 16 2 * 9 = 18
Time	5min

ຂຽນ code ທັງໝົດ ແລະ ຂຽນຜົນທີ່ຄາດວ່າຈະໄດ້ຮັບລົງໃນປຶ້ມບັນທຶກ

Write the entire code and the expected output results in the note.

Q2. ให้เธ้าดัดแปงโปธแกธมข้างเท้งเพื่อพิมข้ันตอนที่ 1 ทา 9 ของตาตะลางผิมถูม. โดย ใช้ถำสั่ง while เท่ำนั้น

Example Output	1*1=1 1*2=2 1*3=3 1*4=4 1*5=5 1*6=6 1*7=7 1*8=8 1*9=9 2*1=2 2*2=4 2*3=6 2*4=8
Time	5min

ຂຽน code ทั้งหมด และ ขຽนผิมที่ถาดอ่าจะได้ธับลิงในปึ้มบันทึก

Write the entire code and the expected output results in the note.

Let's code

1. while Statement

1.1. ຄຳສັ່ງ while ຈະຖືກປະຕິບັດຊ້ຳໆຕາມເງື່ອນໄຂ

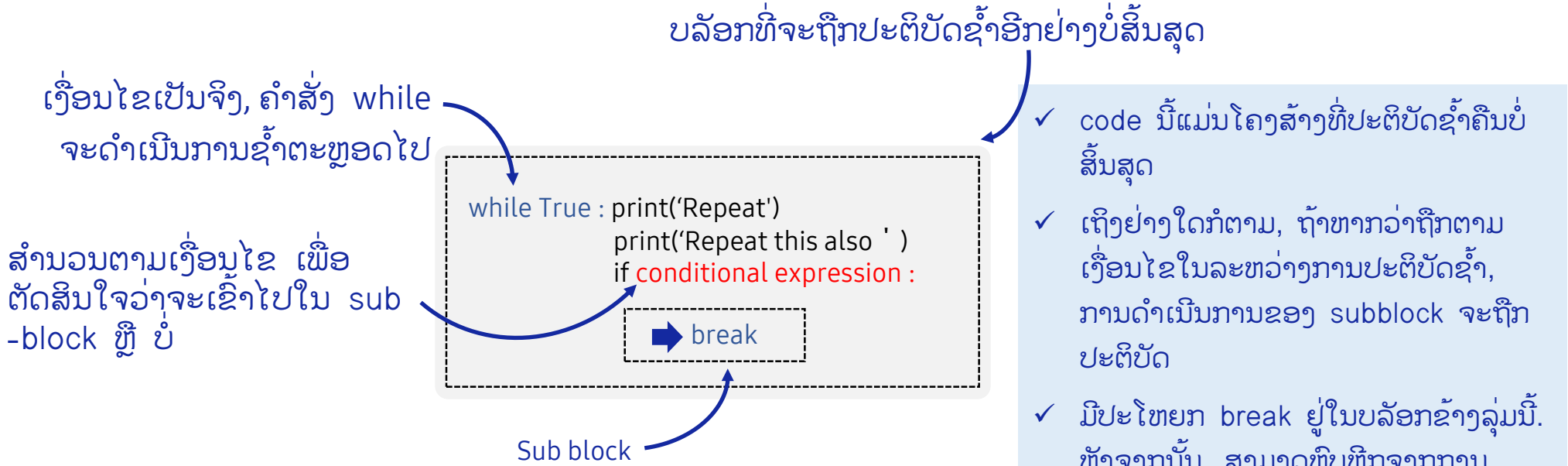
- ເງື່ອນໄຂຄວບຄຸມການປະຕິບັດຊ້ຳຄືນແມ່ນຊື່ທີ່ກຳນົດ ເພາະວ່າມັນຈະປະຕິບັດຊ້ຳຄືນໃນຂະນະທີ່ກົງຕາມເງື່ອນໄຂ.
- ຕົວຢ່າງ: ເຮົາກຳລັງຂີ່ລົດຖີບ, ແຕ່ມີການກໍ່ສ້າງຢູ່ທາງໜ້າຂອງເຮົາ, ດັ່ງນັ້ນຈຶ່ງບໍ່ສາມາດໄປຕໍ່ໄດ້. ໃນຂະນະທີ່ການກໍ່ສ້າງກຳລັງດຳເນີນການຢູ່, ເຮົາຈະເດີນທາງຊ້ຳທາງເດີມຊ້ຳແລ້ວຊ້ຳອີກ.
- ແຕ່ລະຮອບຂອງການຂີ່ລົດຖີບນີ້ຈະກວດສອບເບິ່ງວ່າການກໍ່ສ້າງສຳເລັດ ຫຼື ຍັງ. ຖ້າຢູ່ໃນລະຫວ່າງການກໍ່ສ້າງ, ມັນຍັງສືບຕໍ່ເຮັດວຽກຊ້ຳເກົ່າ ແລະ ຜ່ານໄປຫຼັງຈາກການກໍ່ສ້າງສຳເລັດ.
- ໃນກໍລະນີນີ້, ຈະໃຊ້ການຄວບຄຸມການວົນຊ້ຳແບບມີເງື່ອນໄຂ. ມັນສະດວກທີ່ຈະໃຊ້ການວົນຊ້ຳແບບ while ເພື່ອຄວບຄຸມການວົນຊ້ຳແບບມີເງື່ອນໄຂ.

ຂັ້ນຕອນທົ່ວໄປຂອງຄຳສັ່ງວົນຊ້ຳ ໂດຍໃຊ້ while loop

1. ຄຳສັ່ງ while

1.1. ຄຳສັ່ງ while ທີ່ຖືກປະຕິບັດຊ້ຳໆຕາມເງື່ອນໄຂ

ການວົນຊ້ຳບ່ສິ້ນສຸດ ມີໂຄງສ້າງດັ່ງຕ່ໄປນີ້. ອອກຈາກ loop ທີ່ບ່ສິ້ນສຸດໂດຍໃຊ້ break



- ✓ code ນີ້ແມ່ນໂຄງສ້າງທີ່ປະຕິບັດຊ້ຳຄືນບ່ສິ້ນສຸດ
- ✓ ເຖິງຢ່າງໃດກໍຕາມ, ຖ້າຫາກວ່າຖືກຕາມເງື່ອນໄຂໃນລະຫວ່າງການປະຕິບັດຊ້ຳ, ການດຳເນີນການຂອງ subblock ຈະຖືກປະຕິບັດ
- ✓ ມີປະໂຫຍກ break ຢູ່ໃນບລັອກຂ້າງລຸ່ມນີ້. ຫຼັງຈາກນັ້ນ, ສາມາດຫຼົບຫຼີກຈາກການວົນຊ້ຳນີ້ໄດ້

1. ຄຳສັ່ງ while

1.1. ຄຳສັ່ງ while ທີ່ຖືກປະຕິບັດຊ້ຳໆຕາມເງື່ອນໄຂ

- ມັນເປັນປະໂຫຍກທີ່ຖາມວ່າ "ການກໍ່ສ້າງສຳເລັດ ຫຼື ຍັງ?" ແລະ ໄດ້ຮັບຄຳຕອບຄື "Yes" ຫຼື "No" ແລະ ປະຕິບັດ ຊ້ຳອີກຈົນກ່ວາຄຳຕອບຈະເປັນ "Yes".
- ເງື່ອນໄຂຂອງການວົນຊ້ຳມີຄວາມສັດເຈນ, ແຕ່ບໍ່ຮູ້ຈຳນວນການວົນຊ້ຳທີ່ແນ່ນອນ. ໃນກໍລະນີນີ້, ຄວນໃຊ້ຄຳສັ່ງ while ດີກ່ວາຄຳສັ່ງ for.

```
1 under_construction = True
2 while under_construction:
3     response = input("Is the construction completed?");
4     if response == "Yes" :
5         under_construction = False
6
7 print("Escaped successfully.")
```

```
Is the construction completed? No
Is the construction completed? Yes
Escaped successfully.
```

ການກໍ່ສ້າງສຳເລັດແລ້ວ ຫຼື ຍັງ? No
ການກໍ່ສ້າງສຳເລັດແລ້ວ ຫຼື ຍັງ? Yes
ຜ່ານໄປໄດ້ສຳເລັດ.

1. ຄຳສັ່ງ while

1.2. ຕົວຢ່າງຄຳສັ່ງ while ທີ່ຖືກປະຕິບັດຊ້ຳໆຕາມເງື່ອນໄຂ

- ໃນກໍລະນີໃດທີ່ພວກເຮົາບໍ່ສາມາດຮູ້ຈຳນວນການວົນຊ້ຳໄດ້? ກໍລະນີທີ່ເປັນຕົວແທນຫຼາຍທີ່ສຸດ ເມື່ອໄດ້ຮັບຄ່າຈາກຜູ້ໃຊ້ ແລະ ປະມວນຜົນ. ຍ້ອນວ່າມັນຍາກທີ່ຈະຄາດເດົາຄ່າທີ່ປ້ອນຂໍ້ມູນເຂົ້າໄປຈາກຜູ້ໃຊ້.
- . ຕົວຢ່າງ, ສົມມຸດວ່າຜູ້ໃຊ້ປ້ອນລະຫັດຜ່ານ ແລະ ກວດສອບໂປຣແກຣມສຳລັບເບິ່ງລະຫັດຜ່ານ. ລະຫັດທີ່ພວກເຮົາຂຽນຈະດຳເນີນຕໍ່ໄປກັບຄຳຖາມເດີມ ຈົນກ່ວາຜູ້ໃຊ້ຈະປ້ອນລະຫັດຜ່ານທີ່ຖືກຕ້ອງ.
- ສົມມຸດວ່າລະຫັດຜ່ານແມ່ນ "pythonisfun". ໃສ່ລະຫັດຜ່ານຜິດ aaa ຕາມນີ້. ກວດເບິ່ງວ່າລະຫັດຜ່ານນີ້ຖືກຕ້ອງ ຫຼື ບໍ່ ແລະ ປ້ອນລະຫັດຜ່ານຄືນໃໝ່ຖ້າບໍ່ແມ່ນ.
- ເມື່ອລະຫັດຜ່ານ "pythonisfun" ຖືກໃສ່ຢ່າງຖືກຕ້ອງ, ມັນຈະອອກຈາກ while loop ແລະ ສະແດງ "** Login success**".

```
1 password = ""
2 while password != "pythonisfun":
3     password = input("Enter the password: ")
4 print("** Login success **")
```

```
Enter the password: aaaa
Enter the password: pythonisfun
** Login success **
```

1. ຄຳສັ່ງ while

1.3. ใช้ while ปะติบัดกามซ้ำถึมตามจำนอมที่กำมิด

- กามอิมซ้ำแบบ while บ่สามาดใช้ได้แต่กำละมิบ่รู้จำนอมถั้งเท่านั้น. เถิงแม่มอ่าท่ามรู้จำนอมถั้ง, ท่ามกำสามาดนำใช้กามอิมซ้ำแบบ while ได้.
- ตีอย่าง, ยวม code เพื่อถีดไล่ผึมบอกแต่ 1 หา 10 ใม while loop. ด้อยเหดนี้, ให้เลิ่มต้นจำนอมตีอป่วมที่เพิ่มขึ้นจาก 1 หา 10.
- ปะกาดตีอป่วม s เพื่อเกับถ่าเฟ้านี้ และ กำมิดถ่าเลิ่มต้นให้เป้นสูม. code สำลับกามบอก s ใมกามนับสามาดยวมได้ดั่งต่ำไปนี้.

```
count = 1
s = 0 # s is variable that contains values accumulated and added, reset it as 0
while count <= 10 :
    s = s + count # add count value to s each time
    count = count + 1 # increase count value by 1 every time
print("Sum is", s)
```

```
Sum is 55
```

1. ຄຳສັ່ງ while

1.4. ຄິດໄລ່ຜົນບວກຂອງຕົວເລກທີ່ຜູ້ໃຊ້ປ້ອນເຂົ້າ

```
1 total = 0
2 answer = 'yes'
3 while answer == 'yes':
4     number = int(input('Enter the number: '))
5     total = total + number
6     answer = input('Continue?(yes/no): ')
7 print('Sum is : ', total)
```

```
Enter a number: 5
Continue?(yes/no): yes
Enter a number: 6
Continue?(yes/no): no
Sum is : 11
```

Line 2~3

- ໃຫ້ຄຳຕອບເລີ່ມຕົ້ນກັບຂໍ້ຄວາມທີ່ເປັນ string “yes”.
- Line 3: ໃນຂະນະທີ່ເງື່ອນໄຂເປັນຈິງ, ສະນັ້ນ block ທາງລຸ່ມຈະຖືກປະຕິບັດ.

1. ຄຳສັ່ງ while

1.4. ຄິດໄລ່ຜົນບວກຂອງຕົວເລກທີ່ຜູ້ໃຊ້ປ້ອນເຂົ້າ

```
1 total = 0
2 answer = 'yes'
3 while answer == 'yes':
4     number = int(input('Enter the number: '))
5     total = total + number
6     answer = input('Continue?(yes/no): ')
7 print('Sum is : ', total)
```

```
Enter a number: 5
Continue?(yes/no): yes
Enter a number: 6
Continue?(yes/no): no
Sum is : 11
```

Line 4~6

- ຕົວເລກແມ່ນຖືກກຳນົດຢູ່ໃນຕົວປ່ຽນ number.
- ເພີ່ມຄ່າໃຫ້ກັບ ຈຳນວນລວມ (total) ໂດຍການສະສົມ.
- ເມື່ອຕົວປ່ຽນ answer ມີຄ່າເປັນ 'yes', loops ຈະວົນຊ້ຳຄືນ ແລະ ເມື່ອຄ່າເປັນ 'no' ກໍຈະອອກຈາກການວົນຊ້ຳ.

Pair programming

ແບບຝຶກຫັດການຂຽນໂປຣແກຣມ

ແນວທາງ, ກົນໄກ ແລະ ແຜນສຸກເສີນ

ການກະກຽມໂປຣແກຣມຈັບຄູ່ກ່ຽວຂ້ອງກັບການກຳນົດແນວທາງ ແລະ ກົນໄກ ເພື່ອຊ່ວຍໃຫ້ນັກຮຽນຈັບຄູ່ໄດ້ຢ່າງຖືກຕ້ອງ ແລະ ເພື່ອໃຫ້ພວກເຂົາຈັບຄູ່ກັນ. ຕົວຢ່າງເຊັ່ນ ນັກຮຽນຄວນຜັດປ່ຽນກັນ "ບັງຄັບ mouse" ການກະກຽມທີ່ມີປະສິດຕິພາບ ຈຳເປັນຕ້ອງໃຫ້ມີແຜນສຸກເສີນ ໃນກໍລະນີທີ່ຄູ່ຮ່ວມງານຄົນໃດຄົນໜຶ່ງບໍ່ຢູ່ ຫຼື ຕັດສິນໃຈທີ່ຈະບໍ່ເຂົ້າຮ່ວມດ້ວຍເຫດຜົນໃດໜຶ່ງ ຫຼື ເຫດຜົນອື່ນ. ໃນກໍລະນີເຫຼົ່ານີ້, ສິ່ງສຳຄັນຄືຕ້ອງໄດ້ລະບຸໃຫ້ຊັດເຈນວ່ານັກຮຽນທີ່ມີການປະກອບສ່ວນຈະບໍ່ຖືກລົງໂທດຍ້ອນວ່າການຈັບຄູ່ບໍ່ໄດ້ຜົນດີ.

ການຈັບຄູ່ຄວາມສາມາດທີ່ຄ້າຍຄືກັນ, ບໍ່ຈຳເປັນຕ້ອງມີຄວາມສາມາດເທົ່າທຽມກັນ ໃນຖານະເປັນຄູ່ຮ່ວມງານກັນ

ການຂຽນໂປຣແກຣມຈັບຄູ່ ທີ່ມີປະສິດທິພາບເມື່ອນັກຮຽນທີ່ມີຄວາມສາມາດຄ້າຍຄືກັນ, ເຖິງວ່າບໍ່ຈຳເປັນຕ້ອງມີຄວາມເທົ່າທຽມກັນຖືກຈັບຄູ່ເປັນຄູ່ຮ່ວມງານກັນ. ການຈັບຄູ່ນັກຮຽນທີ່ບໍ່ກົງກັນມັກຈະເຮັດໃຫ້ການມີສ່ວນຮ່ວມທີ່ບໍ່ສົມດຸນກັນ. ຄູສອນຕ້ອງເນັ້ນໜັກວ່າການຂຽນໂປຣແກຣມຈັບຄູ່ບໍ່ແມ່ນກົນລະຍຸດ "ແບ່ງປັນ ແລະ ເອົາຊະນະ", ແຕ່ເປັນຄວາມພະຍາຍາມຮ່ວມມືກັນທີ່ແທ້ຈິງ ໃນທຸກໆຄວາມພະຍາຍາມສຳລັບໂຄງການທັງໝົດ. ຄູຄວນຫຼີກເວັ້ນການຈັບຄູ່ນັກຮຽນທີ່ອ່ອນຫຼາຍກັບນັກຮຽນທີ່ເກັ່ງຫຼາຍ..

ສ້າງແຮງຈູງໃຈໃຫ້ກັບນັກຮຽນໂດຍການສະເໜີສິ່ງຈູງໃຈເພີ່ມເຕີມ

ການສະເໜີແຮງຈູງໃຈເພີ່ມເຕີມສາມາດຊ່ວຍກະຕຸ້ນນັກຮຽນໃຫ້ຈັບຄູ່, ໂດຍສະເພາະກັບນັກຮຽນທີ່ຮຽນເກັ່ງ. ຄູບາງຄົນພົບວ່າການກຳນົດໃຫ້ນັກຮຽນຈັບຄູ່ພຽງແຕ່ໜຶ່ງ ຫຼື ສອງວຽກນັ້ນຈະມີຜົນດີ.

Pair Programming Practice

ປ້ອງກັນການຂີ້ຕົວະໃນການເຮັດວຽກຮ່ວມກັນ

ສິ່ງທ້າທາຍສໍາລັບຄູແມ່ນການຊອກຫາວິທີທີ່ຈະປະເມີນຜົນໄດ້ຮັບຂອງແຕ່ລະບຸກຄົນ, ໃນຂະນະທີ່ນໍາໃຊ້ປະໂຫຍດຈາກການເຮັດວຽກຮ່ວມກັນ. ເຈົ້າຈະຮູ້ໄດ້ແນວໃດວ່ານັກຮຽນຕັ້ງໃຈຮຽນ ຫຼື ຂີ້ຕົວະ? ຜູ້ຊ່ຽວຊານແນະນໍາໃຫ້ທົບທວນຄືນການອອກແບບຫຼັກສູດ ແລະ ການປະເມີນອີກຄັ້ງໜຶ່ງ, ພ້ອມທັງໂອ້ລົມກັບນັກຮຽນຢ່າງຈະແຈ້ງ ແລະ ຊັດເຈນ ກ່ຽວກັບພຶດຕິກໍາທີ່ຈະຖືກຕີຄວາມໝາຍວ່າ ຂີ້ຕົວະ. ຜູ້ຊ່ຽວຊານຊຸກຍູ້ໃຫ້ຄູມອບໝາຍວຽກງານໃຫ້ມີຄວາມໝາຍສໍາລັບນັກຮຽນ ແລະ ອະທິບາຍຄຸນຄ່າຂອງສິ່ງທີ່ນັກຮຽນຈະຮຽນຮູ້ໂດຍການປະຕິບັດວຽກງານສໍາເລັດ.

ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນ

ສະພາບແວດລ້ອມການຮຽນຮູ້ຮ່ວມກັນເກີດຂຶ້ນໄດ້ທຸກເວລາທີ່ຜູ້ສອນມອບໃຫ້ນັກຮຽນເຮັດວຽກຮ່ວມກັນໃນກິດຈະກໍາການຮຽນຮູ້. ສະພາບແວດລ້ອມການຮຽນຮູ້ແບບຮ່ວມມືກັນ ສາມາດມີສ່ວນກ່ຽວຂ້ອງກັບກິດຈະກໍາທີ່ເປັນທາງການ ແລະ ບໍ່ເປັນທາງການ ແລະ ອາດຈະລວມເຖິງ ຫຼື ບໍ່ລວມເຖິງການປະເມີນໂດຍກົງ. ຕົວຢ່າງ: ນັກສຶກສາຄູ່ໜຶ່ງຖືກມອບໝາຍໃຫ້ຂຽນໂປຣແກຣມ; ນັກສຶກສາກຸ່ມນ້ອຍໆສົນທະນາຄໍາຕອບທີ່ເປັນໄປໄດ້ຕໍ່ກັບຄໍາຖາມຂອງອາຈານໃນລະຫວ່າງການບັນລະຍາຍ ແລະ ນັກຮຽນເຮັດວຽກຮ່ວມກັນນອກຫ້ອງຮຽນເພື່ອຮຽນຮູ້ແນວຄວາມຄິດໃໝ່ໆ. ການຮຽນຮູ້ຮ່ວມກັນແຕກຕ່າງຈາກໂຄງການທີ່ນັກຮຽນ "ແບ່ງປັນ ແລະ ເອົາຊະນະ." ເມື່ອນັກຮຽນແບ່ງວຽກກັນ ແຕ່ລະຄົນຈະໄດ້ຮັບຜິດຊອບພຽງແຕ່ສ່ວນໜຶ່ງຂອງການແກ້ໄຂບັນຫາ ແລະ ມີໂອກາດຈໍາກັດຫຼາຍສໍາລັບການເຮັດວຽກຜ່ານບັນຫາກັບຄົນອື່ນ. ໃນສະພາບແວດລ້ອມທີ່ມີການເຮັດວຽກຮ່ວມກັນ, ນັກຮຽນມີສ່ວນຮ່ວມໃນການສົນທະນາທາງປັນຍາເຊິ່ງກັນ ແລະ ກັນ.

Q1. ຕົວເລກ palindrome ໝາຍເຖິງຕົວເລກຈຳນວນຖ້ວນທີ່ມີຄ່າດຽວກັນກັບຄ່າເດີມຂອງມັນ, ເຖິງແມ່ນວ່າຈະຂຽນສະຫຼັບຈາກຫຼັງມາໜ້າ ເຊັ່ນ: 121 ຫຼື 3443. ຂຽນໂປຣແກຣມຕໍ່ໄປນີ້ເພື່ອກຳນົດວ່າຕົວເລກນັ້ນເປັນຈຳນວນ palindrome ຫຼື ບໍ່, ໂດຍໄດ້ຮັບຈຳນວນ n ຈາກຜູ້ໃຊ້.

Output Example

```
Enter an integer: 135
135 is not a palindrome number
```

```
Enter an integer: 3443
135 is a palindrome number
```

Q2. ຄອມພິວເຕີຈະສຸ່ມຈຳນວນຖ້ວນລະຫວ່າງ 1 ເຖິງ 100 ເປັນຄ່າຄຳຕອບທີ່ຖືກຕ້ອງຕໍ່ໄປນີ້. ເມື່ອຜູ້ໃຊ້ສະແດງຄຳຕອບທີ່ຖືກຕ້ອງ, ໂປຣແກຣມຈະແຈ້ງພຽງວ່າ ຈຳນວນຖ້ວນທີ່ສະແດງນັ້ນສູງກວ່າ ຫຼື ຕ່ຳກວ່າ ເມື່ອທຽບກັບຄຳຕອບທີ່ຖືກຕ້ອງທີ່ເກັບໄວ້. ເກມນີ້ຖືກປະຕິບັດຊ້ຳອີກຈົນກ່ວາຜູ້ໃຊ້ຕອບຄຳຖາມໄດ້ຢ່າງຖືກຕ້ອງ.

Output Example

```
Guess a number between 1 to 100
Enter a number: 50
Lower!
Enter a number: 40
Higher!
Enter a number: 51
Higher!
Enter a number: 45
Lower!
Enter a number: 4
Congratulations. Total try = 5
```


End of
Document
Samsung Innovation Campus