香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

INTRODUCTION TO COMPUTER SCIENCE: PROGRAMMING METHODOLOGY

TUTORIAL 4

FLOW CONTROL

Frederick Khasanto
122040014

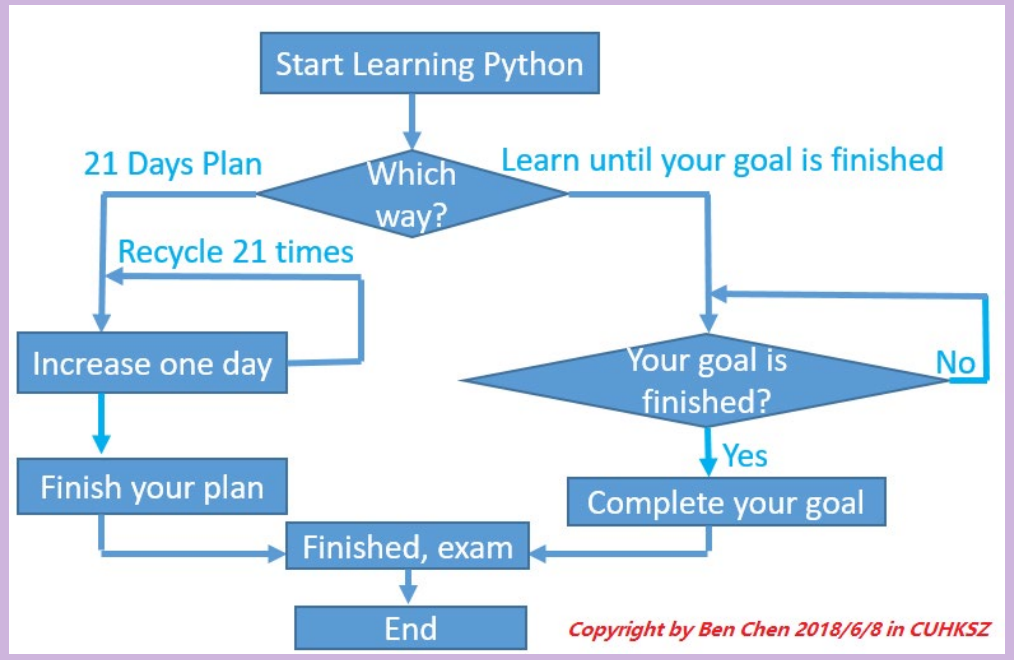
28 September 2023

Outline

- 1. Idea of flow control.
- 2. Conditional flows.
- 3. Repeated flows:
  - i) For loops; ii) While loops.
- 4. Break and Continue.
- 5. Try/except.

I .Idea of flow control

- Flow control: execute individual statements, instructions or function calls in a specific order – one line by one line from up to down.



- Look at the flow chart on the left – imagine a water flow from start to end.

- 1. If statement: conditional flow
- 2. Loops: repeated flow(while loop and for loop)

2. Conditional flow

- Conditional flow (or "Decision flow"):
- 1. One way decision (if);
  2. Two way decision (if, else);
  3. Multi-way decision (if, elif, ..., else).

```
##1.One way decision
weight=float(input('How many pounds does your suitcase weigh?'))
if weight>50:
    print("There is a $25 charge for luggage that heavy.")
print("Thank you for your business.")
```

```
##2.Two way decision
temperature=float(input('What is the temperature?'))
if temperature>70:
    print('Wear shorts.')
else:
    print('Wear long pants.')
print('Get some exercise outside.')
```

```
##3.Multi-way decision
num=3.4
if num>0:
    print('Positive number.')
elif num==0:
    print('Zero.')
else:
    print('Negative number.')
```

3.Repeated flow

- Use loops to execute statements repeatedly.

```
##for loop
for i in loop-list:
    # Loop body
```

```
sumup=0
for num in range(1,101):
    sumup+=num
print('the sum from 1 to 100 is', sumup)
```

```
## while loop
while loop-continuation-condition:
    # Loop body
    Statement(s)
```

```
sumup=0
num=1
while num<=100:
    sumup+=num
    num+=1
print('the sum from 1 to 100 is', sumup)
```

About range()

```
>>> r=range(5)
>>> r
range(0, 5)
>>> type(r)
<class 'range'>
>>> l=list(range(5))
>>> l
[0, 1, 2, 3, 4]
>>> type(l)
<class 'list'>
>>> list(range(1,10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(10,1,-1))
[10, 9, 8, 7, 6, 5, 4, 3, 2]
>>> list(range(1,10,2))
[1, 3, 5, 7, 9]
>>> list(range(10,1,-2))
[10, 8, 6, 4, 2]
>>>
```

** In range(init, end, step), "init" is included but "end" is not.*

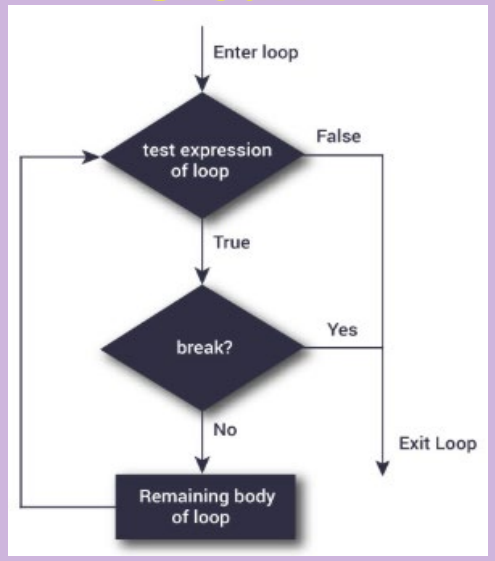
**range() can be changed into list types*

**range(init, end) means from init to end-1;*
**range(init, end, -1) means from init to end+1 step=-1.*

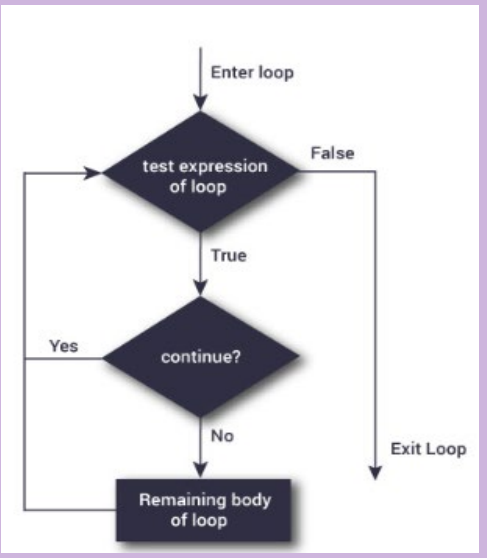
4.Break and Continue

- In Python, break and continue statements can alter the flow of a normal loop.
- The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.
- The continue statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues on with the next iteration.

break



continue




```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```

```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```

5. Try/except

- Use try/except to capture the error in your code.
- The try block lets you test a block of code for errors. If the code in it is good, it will run normally;
  otherwise, if with errors, it can be skipped, without the program being terminated.
- The except block lets you handle the error.

```
try:
    a=2
##    b=3
    print(a+b)
except:
    print('There are some errors in the try block.')
```

** If variable "b" is defined, nothing wrong with try block;*
otherwise except block will be executed as a warning.

```
try:
    num=int(input('Please input an integer:'))
    print(num)
except:
    print("Your input is not a number!")
```

** If an integer is input, the program is fine;*
if, however, what is input is not an integer, like letters "abc", a warning to the user occurs.

Q1: Quadratic equation

- The two roots of a quadratic equation, for example, $ax^2 + bx + c = 0$, can be obtained using the following formula:

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad and \quad r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$b^2 - 4ac$ is called the discriminant of the quadratic equation. If it is positive, the equation has two real roots. If it is zero, the equation has one root. If it is negative, the equation has no real roots. Write a program that prompts the user to enter values for $a, b, and\ c$ and displays the result based on the discriminant. Here are sample runs:

```
Enter coefficients a,b and c in the equation ax^2+bx+c=0:1,3,1
The two roots of the equation are x1=-0.38 and x2=-2.62.
```

```
Enter coefficients a,b and c in the equation ax^2+bx+c=0:1,2,1
There is only one root x=-1.00.
```

```
Enter coefficients a,b and c in the equation ax^2+bx+c=0:1,2,3
The equation has no real roots.
```

Q2: Days in a month

- Write a program that prompts the user to enter the month and year and displays the number of days in the month.
  For example, if the user entered moth 2 and year 2000, the program should display that February 2000 has 29 days.
  If the user entered month 3 and year 2005, the program should display the March 2005 has 31 days.
  Here are sample runs:

```
Enter the month and year:2,2000
February 2000 has 29 days
```

```
Enter the month and year:3,2019
March 2019 has 31 days
```

Q3: Sum the digits

- Previously we write a program that reads an integer between 0 and 1000 and adds all the digits in the integer. For example, if an integer is 932, the sum of all its digit is 14. Now please use while loop to solve this problem. Your code should be able to handle numbers greater than 1000.

```
Enter a number:932
Sum up all the digits as: 14
Do you want to continue? y or n:y
Enter a number:20190221
Sum up all the digits as: 17
Do you want to continue? y or n:n
```

Q4: Count numbers

- Write a program that reads an unspecified number of integers, determines how many positive and negative values have been read, and computes the total and average of the input values (not counting zeros). Your program ends with the input 0. Display the average as a floating-point number. Here is a sample run:

```
Enter an integer, the input ends if it is 0:1 Enter
Enter an integer, the input ends if it is 0:3 Enter
Enter an integer, the input ends if it is 0:-2 Enter
Enter an integer, the input ends if it is 0:5 Enter
Enter an integer, the input ends if it is 0:0 Enter
The number of positives is 3
The number of negative is 1
The sum of numbers is 7
The average of numbers is 1.75
```

Q5: Find factors

- Write a program that reads an integer and displays all its smallest factors, also known as prime factors. For example, if the input integer is 120, the output should be as follows:

  2,2,2,3,5

Q6: Display a pyramid

- Write a program that prompts the user to enter an integer from 1 to 15 and displays a pyramid with height of that integer, as shown in the following sample run: