# Python Programming

## 03. Iteration Statements

Evan Chang


# What's iteration?

- Do stuff repeatedly.
  - Spamming emojis.
    - 轟隆隆隆🤣🤣隆隆隆隆衝衝衝衝😏😏😏拉風😎😎😎引擎發動🔑🔑🔑引擎發動+🚗+👉+🚗
  - Wash hands every 5 minutes.
  - Keep drinking until passing out.

# `while`

- Syntax

```
while statement:
    indented_statements
```

  - if `statement` is `True`, execute `indented_statements` repeatedly.

- Example

```
n = int(input('Input a positive integer: '))
while n > 0:
    print(n)
    n = n - 1
# n↵ n-1↵ ... 1↵
```

# while

- More examples - `n!`

```
n = 10
factorial = 1
while n > 0: # n, n-1 ... 1
    factorial *= n
    n -= 1
print('n! is', factorial)
```

# while

- More examples - the sum of non-negative integers (1/2)

```
n = 0
total = 0
while n >= 0: # n, n-1 ... 1
    n = int(input('Input a non-negative integer: '))
    if n >= 0:
        total += n
print('total is', total)
```

# while

- More examples - the sum of non-negative integers (2/2)

```
n = 0
total = 0
while n >= 0: # n, n-1 ... 1
    total += n
    n = int(input('Input a non-negative integer: '))
print('total is', total)
```

# while

- More examples - String appending

```
tmp = ''
final = ''
while tmp != '.':
    tmp = input('Input a word: ')
    if tmp == '.':
        final = final + tmp
    else:
        final = final + ' ' + tmp
print(final)
```

# Infinite loop

- What's wrong?

```
n = 5
while n > 0:
    print(n)
print('Done!')
```

- This is an **infinite loop**.
  - Since `n` is always greater than 0
  - The `while` statement is always `True`.

# Infinite loop

- What's wrong?

```
n = 5
while n > 0: # can be written as `while True:`
    print(n)
print('Done!')
```

- How to stop?
  - Click the `Stop` button.
  - `Ctrl+C` to send a `KeyboardInterrupt` to the program.

# Useful infinite loop

- Sometimes we need an infinite loop.
  - A server that keeps running.
  - A game that keeps running.
- What if we need to stop the loop without closing the program?
  - `break`

# `while` - `break`

- `break` statement stops the loop immediately.
  - Syntax

```
while statement1:
    indented_statements
    if statement2:
        break # stop the while loop immediately
```

# `while` - `break`

- Example - chatbot

```
while True:
    line = input('Repeat after me: ')
    if line == 'done':
        break # -----------
    print(line) #          \
print('Done!') # <----------
```

```
Repeat after me: hello, world!
hello, world!
Repeat after me: done
Done!
```

# `while` - `continue`

- `continue` statement skips the rest of the loop and goes to the next iteration.
  - Syntax

```
while statement1:
    indented_statements
    if statement2:
        continue # skip the rest of the loop
    indented_statements # may not be executed
```

# `while` - `continue`

- Example - chatbot

```
while True: # <-----------------------------------
    line = input('Repeat after me: ') #           \
    if line[0] == '#': #                          \
        continue # ------------------------------
    if line == 'done':
        break
    print(line)
print('Done!')
```

```
Repeat after me: # hello, world!
Repeat after me:
```

# Nested Loops

- A loop can be inside another loop.
  - Example - multiplication table

```
x = 1
while x <= 9:
    y = 1
    while y <= 9:
        print(x, ' * ', y, ' = ', x * y)
        y = y + 1
    x = x + 1
```

# Nested Loops

- More example

```
x = 1
while x > 0:
    x = int(input('Input a positive integer: '))
    n = x
    fac = 1
    while n > 0:
        fac = fac * n
        n = n - 1
    print(x, '! is', fac)
```

- But `0! is 1` will be printed, too.

# Nested Loops

- Use `break` to stop the outer loop.

```
while True:
    n = int(input('Input a positive integer: '))
    if n <= 0:
        break
    fac = 1
    while n > 0:
        fac = fac * n
        n = n - 1
    print(n, '! is', fac)
```

# Nested Loops with `break`

```
while True: # <-------------------------------------
    n = int(input('Input a positive integer: ')) #  \
    if n <= 0: #                                    \
        break # ------------------------------------
    fac = 1
    while True: # <--------------------
        fac = fac * n #                \
        n = n - 1  #                   \
        if n == 0: #                   \
            break  # -------------------
    print(n, '! is', fac)
```

# Indefinite Loops vs Definite Loops

- Indefinite loops
  - Loops that don't have a definite number of iterations.
  - Hard to predict how many times the loop will be executed.
  - e.g. `while` loops
- Definite loops
  - Loops that have a definite number of iterations.
  - e.g. `for` loops

# for

- Syntax

```
for variable in sequence:
    indented_statements
```

- Example

```
for i in [1, 2, 3, 4, 5]:
    print(i) # 1↵ 2↵ 3↵ 4↵ 5↵
```

# for

- More example - friends

```
friends = ['Stanly', 'Tozy', 'Power', 'NL'] # a list
for friend in friends:
    print('Hi,', friend)
print('Done!')
```

```
Hi, Stanly
Hi, Tozy
Hi, Power
Hi, NL
Done!
```

# for

- Note: the **iteration variable** is a copy of the element in the sequence.

```
friends = ['Stanly', 'Tozy', 'Power', 'NL']
for friend in friends:
    friend = 'Mr. ' + friend
print(friends)
```

```
['Stanly', 'Tozy', 'Power', 'NL']
```

# `for` 's best friend - `range`

- Syntax

```
range(start, stop, step)
```

  - return a sequence of numbers from `start` to `stop` (exclusive) with `step` size.

# `for` 's best friend - `range`

- Examples

```
for i in range(5):
    print(i) # 0↵ 1↵ 2↵ 3↵ 4↵

for i in range(3, 6):
    print(i) # 3↵ 4↵ 5↵

for i in range(10, 0, -2):
    print(i) # 10↵ 8↵ 6↵ 4↵ 2↵
```

# `for` 's best friend - `range`

- With `len()`

```
friends = ['Stanly', 'Tozy', 'Power', 'NL']
for i in range(len(friends)):
    friend = friends[i]
    print('Hi,', friend)
```

# for

- Final example - how to find the maximum value of a list?
- But we can only check one value at a time.

```
l = [9, 41, 12, 3, 74, 15]

      value   maximum
i=0:    9         9
i=1:   41        41
i=2:   12        41
i=3:    3        41
i=4:   74        74
i=5:   15        74 <- the maximum value
```

# for

- Final example - how to find the maximum value of a list?

```
l = [9, 41, 12, 3, 74, 15]
maximum = -999 # why this number?
for value in l:
    if value > maximum:
        maximum = value
print(maximum)
```

# Exercises

# 1. 學妹模擬器

- 上了大學的阿群發現學妹都不想理他，
- 於是你決定幫他寫一個學妹模擬器：
  1. 讓阿群可以一直輸入訊息
  2. 學妹每次都會回覆「恩恩、呵呵、是喔...」
     但為了讓阿群不要發現，每次回覆都會跟上次不同
  3. 如果阿群輸入「我喜歡你」，
     學妹就會回覆「哈哈學長我先去洗澡囉」，程式就結束了

# 2. 可以忘記學妹的質數

- 被學妹打槍的阿群還是持續朝思暮想著學妹，
- 於是他決定開始在腦中數質數，讓自己冷靜冷靜，
- 但是他發現他不停數錯，
- 於是你決定寫一個程式幫他確定一個數字是不是質數：
  1. 讓阿群可以輸入一個整數
  2. 印出他是不是質數

# 3. 全婆俠

- 阿群雖然表面上是個熱愛學妹的紳士,
- 但是其實私底下他還是個不折不扣的全婆俠,
- 他每天都需要在寢室朝向偶像現在的方向大喊:

```
「娜璉我婆我婆!」「定延我婆我婆!」「Momo我婆我婆!」
「Sana我婆我婆!」「志效我婆我婆!」「Mina我婆我婆!」
「彩瑛我婆我婆!」「子瑜我婆我婆!」
```

十遍,於是你決定要用 `for` 迴圈幫助他印出這些講稿

# 4. 學妹的告白

- 每天阿群都會被學妹告白，
- 但都只會持續幾秒，他就會醒來了，
- 而最近持續的時間越來越不穩定，
- 讓小群很擔心有一天學妹就不會再出現了，
- 於是你決定寫一個程式幫他計算這一個禮拜的持續時間的**最大值**、**最小值**以及**標準差**：
  1. 阿群已經將秒數都放在一個 list 中
  2. 用 `for` 迴圈計算**最大值**、**最小值**以及**標準差**

# 5. 愛心蠟燭

- 阿群最近在練習排愛心蠟燭，但是怎麼排都不好看，
- 於是你決定幫他寫一個程式排出好看的愛心圖形:
  1. 使用兩層 `while` 迴圈遍歷：

```
y:  1.5 ~ -1.5, 間距:0.05
x: -1.5 ~  1.5, 間距:0.025
```

  2. 對 $(x^2+y^2-1)^3-x^2y^3<=0$ 的點畫出「*」，其餘點以空格畫出
  3. 注意換行

# 5. 愛心蠟燭

# Acknowledgment


- Prof. Chang-Chieh Cheng. National Yang Ming Chiao Tung University, Taiwan
- Python for Everybody