01076103, 01076104

Programming Fundamental

Programming Project

Function

Function

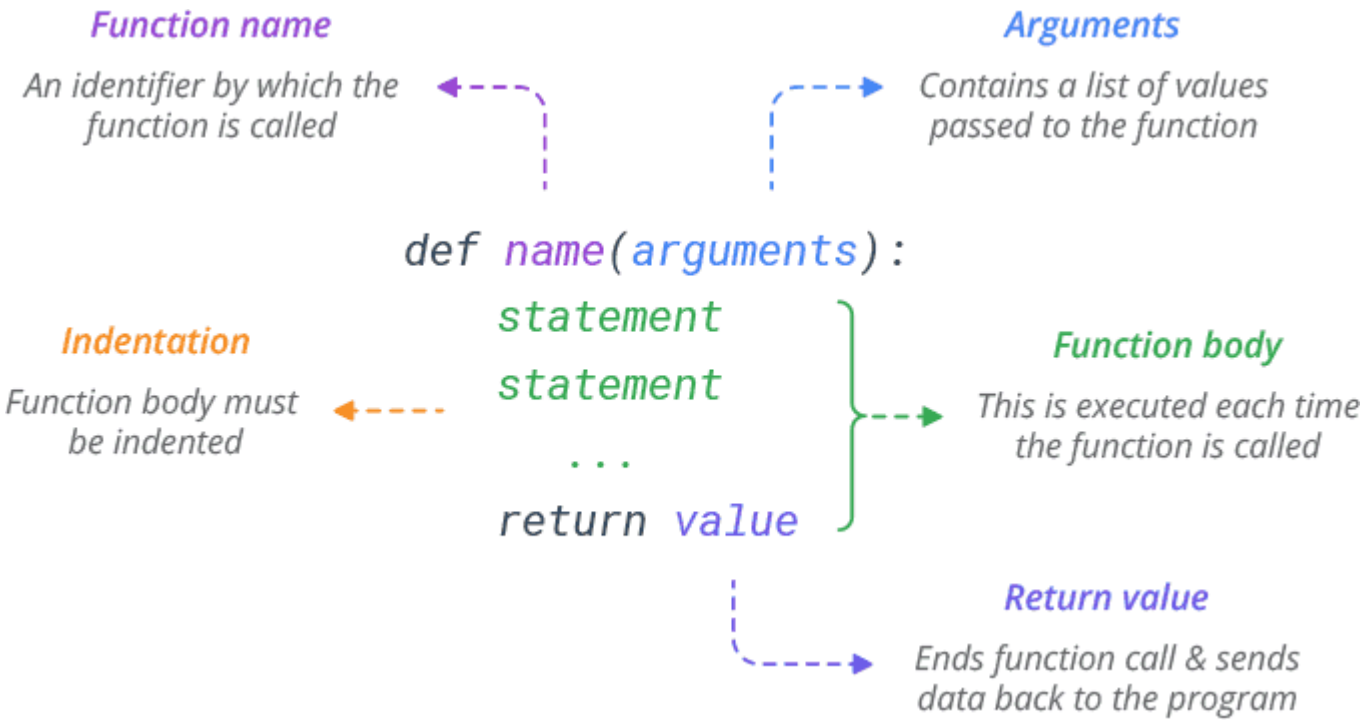
- ในการทำงานบางอย่างที่ต้องการทำงานซ้ำๆ เราสามารถสร้างเป็น Function ได้
- ปกติใน python มีฟังก์ชันให้ใช้งานมากอยู่แล้ว
- https://docs.python.org/3/library/functions.html

Built-in Functions

A abs() aiter() all() any() anext() ascii() **B** bin() bool() breakpoint() bytearray() bytes() **C** callable() chr() classmethod() compile() complex() **D** delattr() dict() dir() divmod()	**E** enumerate() eval() exec() **F** filter() float() format() frozenset() **G** getattr() globals() **H** hasattr() hash() help() hex() **I** id() input() int() isinstance() issubclass() iter()	**L** len() list() locals() **M** map() max() memoryview() min() **N** next() **O** object() oct() open() ord() **P** pow() print() property()	**R** range() repr() reversed() round() **S** set() setattr() slice() sorted() staticmethod() str() sum() super() **T** tuple() type() **V** vars() **Z** zip() **_** __import__()

Function

- แต่เราก็สามารถสร้าง User defined function ขึ้นใช้เองได้
- รูปแบบของการเขียน Function



Function

- ตัวอย่าง จะเห็นว่า function ช่วยเรื่อง reuse ได้ ซึ่งเป็นเรื่องสำคัญของการเขียนโปรแกรม ฟังก์ชันจะรับค่าได้ เรียกว่า argument

main.py

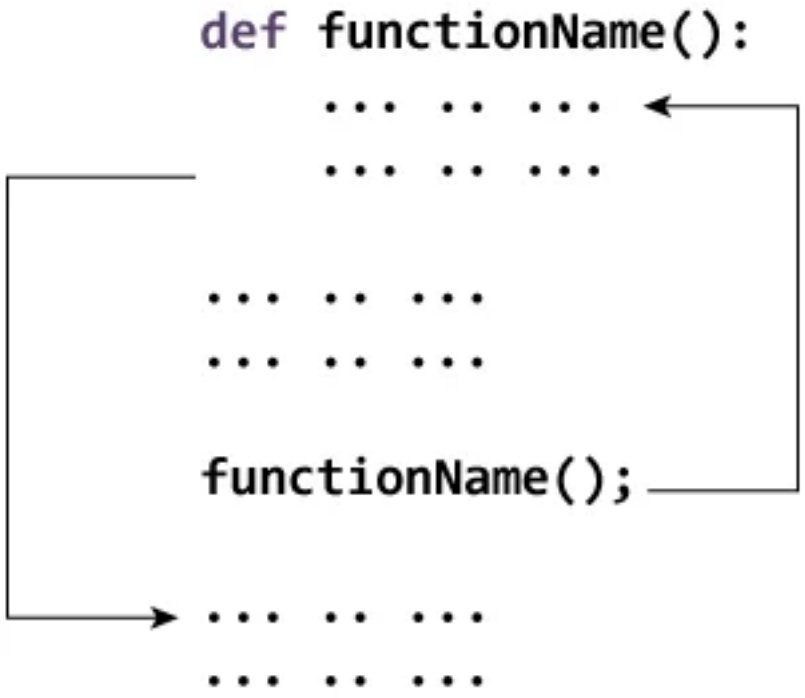
```
1  def hello(name):
2      # code ที่อยู่ใน function ต้อง indent
3      """
4      Function to print ส่วนนี้เรียกว่า
5      docstring เอาไว้อธิบายการทำงานของ
6      function
7      """
8      print ("Good afternoon, " + name)
9
10 hello("John")
11 hello("Bob")
```

Console Shell

```
Good afternoon, John
Good afternoon, Bob
>
```

Function

- การทำงานของ function



Function

- กรณีที่มีการส่งค่ากลับ จะใช้คำสั่ง return

main.py

```
1 def is_even(n):
2     if (n%2==0):
3         return True
4     else:
5         return False
6
7 print("1 is even:",is_even(1))
```

Console Shell

```
1 is even: False
```

- กรณีฟังก์ชันไม่ซับซ้อน จะเขียนย่อๆ ก็ได้

main.py

```
1 def is_even(n): return True if n%2==0 else False
2
3 print("1 is even:",is_even(1))
4
```

Console Shell

```
1 is even: False
```

Function

- สมมติเราจำเป็นต้องเขียน function is_odd() เพิ่มเติมจะเขียนอย่างไร
- เราจะเขียนแบบนี้ก็ได้

main.py ×

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

- แต่โปรแกรมมีลักษณะที่คล้ายกับ is_even ซึ่ง developer ที่ดีพึงหลีกเลี่ยง ดังนั้นเราจะเขียนแบบนี้ ซึ่งจะสั้นกว่า และมีการ reuse

```
7 ▼ def is_odd(n):
8       return not(is_even(n))
```

Function

- การใช้ function จะทำให้โปรแกรมอ่านง่ายขึ้น (clean code) เช่น โปรแกรมที่แสดงเฉพาะเลขคู่ใน list ถ้าแยกส่วนตรวจสอบออกมา โปรแกรมจะดูง่ายขึ้น

main.py ×

```
1 def is_even(n): return True if n%2==0 else False
2
3 def is_odd(n): return not is_even(n)
4
5  lst1 = [1,2,3,4,5,6,7,8,9]
6
7 for num in lst1:
8     if is_even(num):
9         print(num)
10
```

Function

- โปรแกรมรับค่าตัวเลข และบอกว่าเป็น square number หรือไม่ โดยทำเป็นฟังก์ชัน is_square คือ เป็นตัวเลขกำลังสองของเลขอื่นหรือไม่

main.py

```
1  def is_square(n):
2      for i in range(n):
3          if (n==i*i):
4              return True
5      return False
6
7  print(is_square(-1))
8  print(is_square(0))
9  print(is_square(4))
10 print(is_square(5))
11 print(is_square(25))
```

Console Shell

```
False
False
True
False
True
>
```

Function

- บางกรณีต้องมีการ Return หลายค่า
- เช่น จะเขียนโปรแกรมที่ใส่จำนวนเงินแล้ว Return มูลค่าสินค้า กับ ภาษีมูลค่าเพิ่ม

main.py

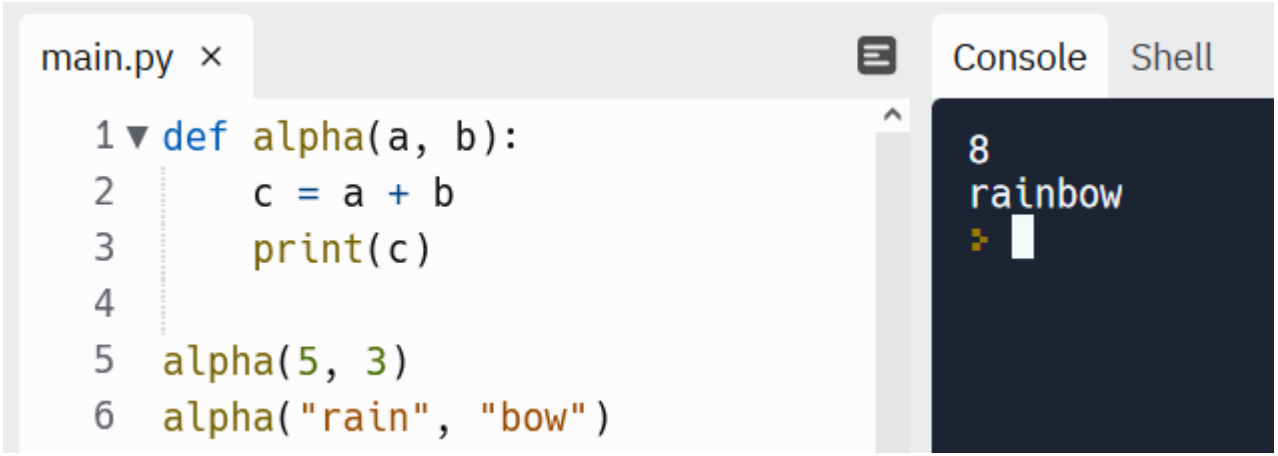
```
1  def price_with_vat(amount):
2      vat = amount * 7 / 107  # 107  * 7 /107
3      price = amount - vat
4      return price, vat
5
6  print(price_with_vat(107))
7  p, v = price_with_vat(214)
8  print("p = ", p)
9  print("v = ", v)
```

Console Shell

```
(100.0, 7.0)
p =  200.0
v =  14.0
>
```

Function

- เนื่องจากตัวแปรของภาษา Python เป็น Duck Typing หรือ Dynamic Typing ดังนั้น ต้องตรวจสอบโปรแกรมให้ดี ไม่เช่นนั้นอาจเกิดผลที่ไม่ต้องการได้



- จะเห็นว่าฟังก์ชันทำงานถูกต้อง ในการเรียกใช้ทั้งสองครั้ง แต่ผลการทำงานต่างกัน โดยการเรียกครั้งแรกเป็นการบวก แต่การเรียกครั้งที่ 2 เป็นการ concatenate

Function

- ในกรณีที่มี parameter หลายตัว มีความเป็นไปได้ว่าอาจใส่สลับกันมา เพื่อป้องกันไม่ให้ใส่สลับกัน อาจใช้วิธีระบุชื่อตัวแปรตอนเรียกก็ได้ (named argument)
- เราสามารถเริ่มใช้ named argument ตอนไหนก็ได้ แต่ argument หลังจากนั้นต้องเป็น named argument ด้วยทั้งหมด (ก่อนหน้านั้น ถือว่าเรียงตามลำดับ)

main.py ×

```
1 def price_with_vat2(amount, vat_rate):
2     vat = amount * vat_rate / (100+vat_rate)  # 107  * 7 /107
3     price = amount - vat
4     return price, vat
5
6 print(price_with_vat2(amount=107, vat_rate=7))
7
```

Console Shell

```
(100.0, 7.0)
>
```

Function

- ในบางกรณีที่พารามิเตอร์บางตัวมักเป็นค่าใดค่าหนึ่งบ่อยๆ อาจกำหนดให้มีค่า default argument ได้
- จากรูปจะเห็นว่า vat_rate มักจะเท่ากับ 7% ดังนั้นจึงกำหนดว่าถ้าไม่ได้ส่งค่าเข้าไป จะถือว่า = 7%
- แต่การใช้ default argument จะต้องเป็นพารามิเตอร์ตัวหลังสุดเท่านั้น

main.py ×

```
1 def price_with_vat2(amount, vat_rate=7):
2     vat = amount * vat_rate / (100+vat_rate)  # 107  * 7 /107
3     price = amount - vat
4     return price, vat
5
6 print(price_with_vat2(107))
```

Console Shell

```
(100.0, 7.0)
```

Function

- บางฟังก์ชัน อาจไม่สามารถระบุจำนวน argument ที่แน่นอนได้ กรณีนี้จะเรียกว่า Arbitrary Arguments
- ภาษา Python มี feature ที่รองรับกรณีนี้ไว้ ตามแสดงในตัวอย่าง

main.py

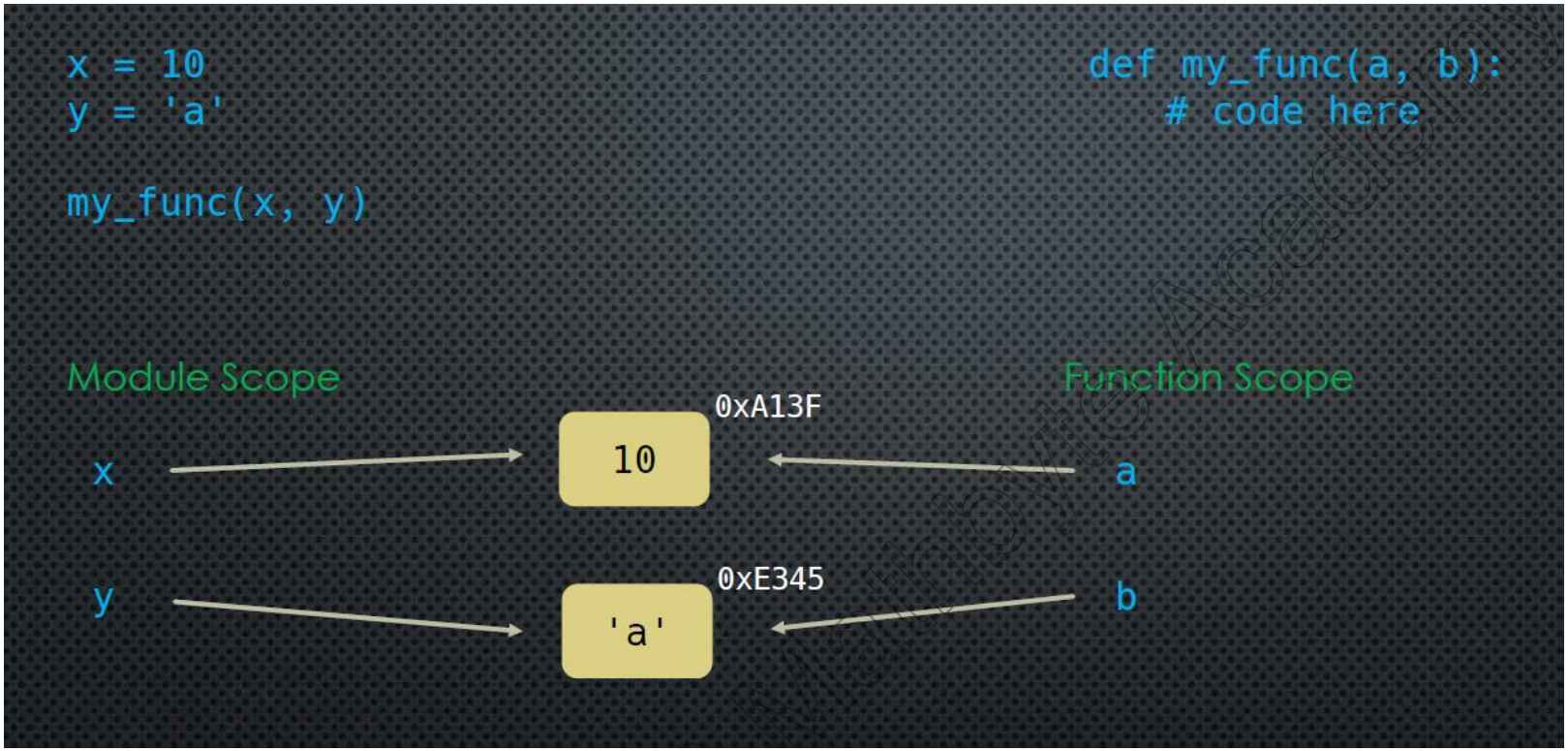
```
1  def greet(*names):
2      """This function greets all
3      the person in the names tuple."""
4
5      # names is a tuple with arguments
6      for name in names:
7          print("Hello", name)
8
9
10 greet("Monica", "Luke", "Steve", "John")
11
```

Console Shell

```
Hello Monica
Hello Luke
Hello Steve
Hello John
```

Function

- ในการส่งพารามิเตอร์เข้าไปในฟังก์ชัน เป็นการส่งตำแหน่งเข้าไปใน function เช่นจากรูปจะเห็นว่า argument ของฟังก์ชัน คือ a กับ b จะถูกกำหนดให้ชี้ไปที่ x และ y



Function

- **Exercise 5.1** ให้เขียน function ชื่อ day_of_year(day, month ,year) โดยมีการคืนค่า คือ day_of_years เป็นวันที่ลำดับที่เท่าใดของปีคริสตศักราช year
  - ปีที่เป็น Leap Year เดือนกุมภาพันธ์จะมี 29 วัน
  - ให้สร้างฟังก์ชัน is_leap เพื่อตรวจสอบ leap year แยกออกมา และให้ฟังก์ชัน day_of_year เรียกใช้ is_leap อีกที

Function

- การออกแบบโปรแกรมที่ดี พยายามให้โปรแกรมแต่ละส่วนสั้นและอ่านง่ายที่สุด ดังนั้นจึงต้องพยายามแยกส่วนโปรแกรมออกเป็น ฟังก์ชันย่อยๆ ให้มากที่สุด
- แต่ละฟังก์ชันควรมีหน้าที่เฉพาะ และ เบ็ดเสร็จในตัวเอง เช่น ฟังก์ชัน is_leap ที่แม้จะเป็นฟังก์ชันเล็กๆ แต่ก็ช่วยให้โปรแกรมอ่านง่ายขึ้น และ ซับซ้อนน้อยลง
- มีผู้ให้แนวทางปฏิบัติว่า แต่ละส่วนของโปรแกรมควรยาวประมาณ 15 บรรทัด จะทำให้โปรแกรมอ่านง่าย

Function

- **Exercise 5.2** จากโปรแกรม 5.1 ให้เขียนฟังก์ชัน เพิ่มเติมเป็น date_diff
  - รับข้อมูลในรูปแบบ “dd-mm-yyyy” เช่น

    date_diff(“1-1-2018”, “1-1-2020”) จะได้ 731 วัน
    date_diff(“25-12-1999”, “9-3-2000”) จะได้ 76 วัน

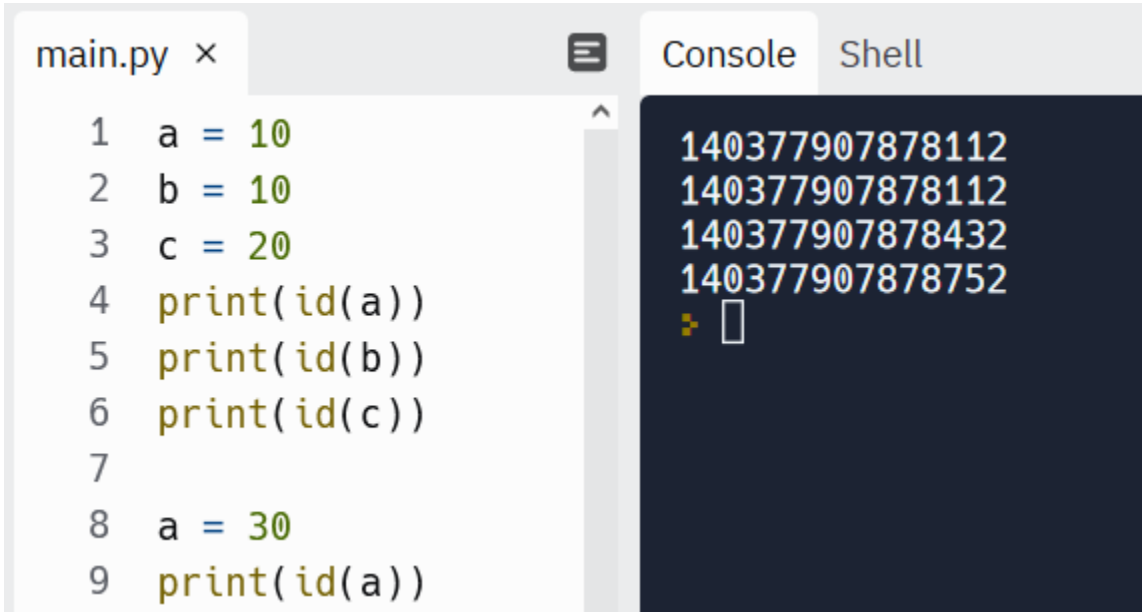
  - ให้เขียนฟังก์ชัน day_in_year โดยจะส่งค่าจำนวนวันของปี (365 หรือ 366) โดยรับข้อมูลเป็น ปี
  - ส่งคืนข้อมูลเป็นจำนวนวันตั้งแต่วันที่แรก จนถึงวันที่สอง โดยรวมทั้ง 2 วันนั้นเข้าไปด้วย
  - ให้สมมติว่าวันแรก จะต้องมาก่อนวันที่สองเสมอ ดังนั้นไม่ต้องตรวจสอบ

Function

- **Exercise 5.3** จากโปรแกรม 5.2 ให้เขียนฟังก์ชัน date_diff เพิ่มเติม โดยให้มีการตรวจสอบ
  - วันที่ต้องเป็นวันที่ถูกต้องของเดือนนั้นๆ
  - เดือนต้องอยู่ระหว่าง 1-12
  - เดือนกุมภาพันธ์ของปีที่มี Leap Year เท่านั้นที่จะมี 29 วันได้
  - หากข้อมูล Input ผิดพลาด ให้ Return -1

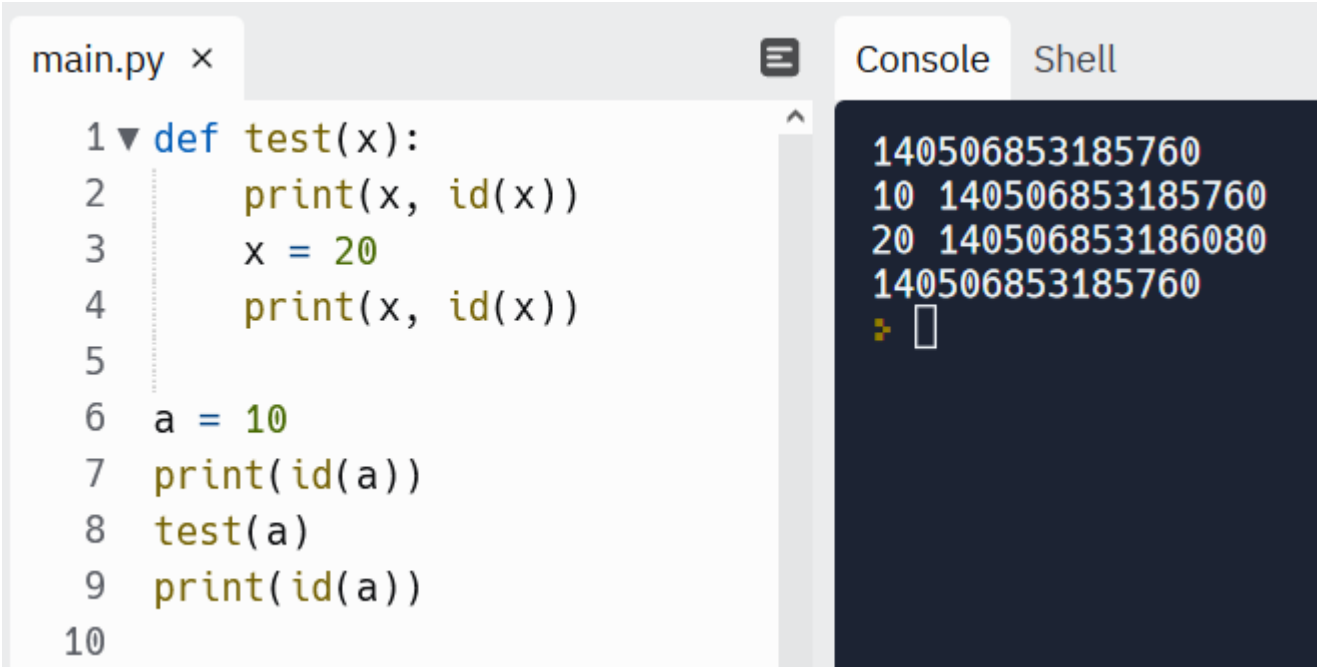
Mutable กับ Immutable

- List เป็นโครงสร้างข้อมูลที่เรียกว่า Mutable (แปลว่า เปลี่ยนแปลงได้)
- ส่วน Int, Float, Bool, String เป็นข้อมูลที่เป็น Immutable โดยหากเราแก้ไขข้อมูลในตัวแปรนั้น Python จะทำลายข้อมูลนั้น และสร้างขึ้นใหม่ โดยไม่ใช้ข้อมูลตำแหน่งเดิม ตามตัวอย่างที่เมื่อเปลี่ยนค่าใน a จะพบว่า a จะไม่ได้อยู่ที่ตำแหน่งเดิม



Mutable กับ Immutable

- จากความรู้ข้างต้น ในกรณีที่เราส่งพารามิเตอร์เข้าไปใน argument ของฟังก์ชัน
- หากพารามิเตอร์นั้นเป็นแบบ Immutable ก็ไม่มีปัญหาอะไร เพราะหากมีการกำหนดค่าใหม่ในฟังก์ชัน ก็จะเหมือนกับเป็นตัวแปรใหม่ ไม่กระทบตัวแปรเดิม



Mutable กับ Immutable

- แต่หากเป็นข้อมูลที่เป็น mutable แล้ว หากมีการแก้ไขข้อมูลตัวแปรภายในฟังก์ชัน อาจทำให้มีปัญหาได้ (ยกเว้น กรณีที่เป็นความตั้งใจ)
- จะเห็นว่า List a ที่ส่งเป็นพารามิเตอร์ มีการแก้ไขไปด้วย ดังนั้นควรระวังกรณีนี้

main.py

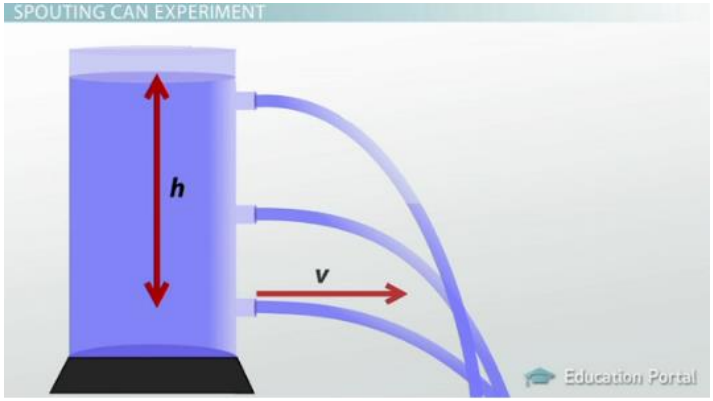
```
1 def test(x):
2     print(id(x), x)
3     x[1] = 50
4     print(id(x), x)
5
6 a = [10, 20, 30]
7 print(id(a), a)
8 test(a)
9 print(id(a),a )
```

Console Shell

```
139658320515584 [10, 20, 30]
139658320515584 [10, 20, 30]
139658320515584 [10, 50, 30]
139658320515584 [10, 50, 30]
```

Problem Solving : Case Study

- มีถังน้ำขนาด 2 x 2 x 2 เมตร มีน้ำอยู่เต็ม เจาะรูเป็นวงกลมที่มีพื้นที่หน้าตัด 2 ตารางเซนติเมตร จำนวน 4 จุด คือ ที่ก้นถัง, ที่ความสูง 50 cm, ที่ความสูง 100 cm และที่ความสูง 150 cm ตามลำดับ จงเขียนโปรแกรมว่าใช้เวลาเท่าใด น้ำจึงไหลออกจากถังจนหมด (ให้คำนวณเป็น รอบ รอบละ 0.01 วินาที)
- ปริมาณน้ำที่ไหลออกจากรู = ความเร็วของน้ำ x เวลา x พื้นที่หน้าตัด
- ความเร็วของน้ำที่ไหลออกจากรู = $(2 \times g \times \text{ความสูงของน้ำจากรู})^{1/2}$
- ดังนั้นปริมาณน้ำไหลออก = $(2 \times g \times \text{ความสูงของน้ำจากรู})^{1/2}$ x เวลา x พื้นที่หน้าตัด (น้ำต้องอยู่เหนือรูเท่านั้น)



Problem Solving : Case Study

- **วิเคราะห์โจทย์**
  - ในชีวิตความเป็นจริง น้ำจะไหลต่อเนื่องไปเรื่อยๆ แต่ในการเขียนโปรแกรมเราจะต้องจำลองการทำงานเป็นรอบๆ
  - เราสามารถมองการทำงานได้ดังนี้
    1. น้ำไหลออกจากรู ตามความเร็ว (ความสูงของน้ำ)
    2. ตรวจสอบปริมาณน้ำที่หายไป
    3. ตรวจสอบระดับน้ำลดลง
    4. วนไปทำข้อ 1 จนกว่าน้ำจะหมดถัง

Problem Solving : Case Study

- **วิเคราะห์โจทย์**
  - เนื่องจากมีรูจำนวน 4 รู ซึ่งมีความสูงไม่เท่ากัน ดังนั้นน้ำที่ไหลออกจากรูจึงไม่เท่ากันด้วย
  - เราจะกำหนดให้ความสูงของน้ำ = h (h ต้องมากกว่า 0)
  - น้ำที่ไหลออกจากรู จำนวน 4 รู = vol1, vol2, vol3, vol4
  - ดังนั้นปริมาณน้ำที่หายไป = vol1+vol2+vol3+vol4
  - คำนวณ h ใหม่ = h เดิม – (ปริมาณน้ำที่ไหลออก / พื้นที่หน้าตัด)
  - นับเวลาเพิ่ม และ ตรวจสอบว่าน้ำหมดหรือยัง

Problem Solving : Case Study

- เขียนเป็นโค้ดเทียม
  - กำหนด h = 2
  - กำหนดค่า dt = 0.01
  - กำหนดค่า area = 0.002
  - กำหนดค่า g = 10
  - กำหนดค่า timer = 0

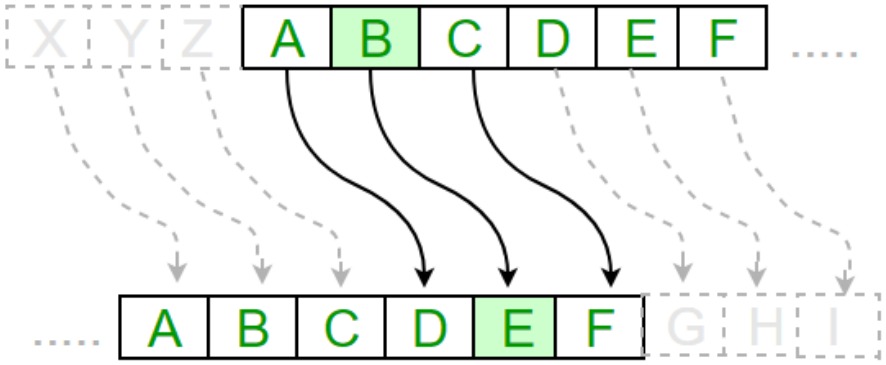
Problem Solving : Case Study

- เขียนเป็นโค้ดเทียม
  - ทำงานต่อไปนี้ หาก h > 0
    - คำนวณค่า vol1 = $(2 \times g \times h)^{1/2} \times dt \times area$
    - ถ้า h > 0.5 คำนวณค่า vol2 = $(2 \times g \times (h-0.5))^{1/2} \times dt \times area$ ถ้าไม่ใช่ vol2 = 0
    - ถ้า h > 1.0 คำนวณค่า vol3 = $(2 \times g \times (h-1.0))^{1/2} \times dt \times area$ ถ้าไม่ใช่ vol3 = 0
    - ถ้า h > 1.5 คำนวณค่า vol4 = $(2 \times g \times (h-1.5))^{1/2} \times dt \times area$ ถ้าไม่ใช่ vol3 = 0
    - ผลรวม total = vol1 + vol2 + vol3 + vol4
    - คำนวณค่า h ใหม่ = h เดิม – (total/4)
    - Timer = timer + dt
  - แสดงผลค่า timer

Case Study : Caesar cipher

- ในสมัยโรมัน มีการส่งข้อมูลแบบเข้ารหัส เพื่อป้องกันการอ่านแอบอ่าน เรียกว่าการเข้ารหัสแบบซีซาร์ โดยใช้วิธีการเลื่อนตัวอักษร

  abcdefghijklmnopqrstuvwxyz
- เช่น ถ้ากำหนดเลื่อน 5 ข้อความ hello -> mkrru
- กรณีที่เป็นตัว z จะวนกลับไปใช้ abc ใหม่



Case Study : Caesar cipher

- **วิเคราะห์โจทย์** ข้อนี้อาจจะเริ่มจากพิจารณาว่าต้องมีฟังก์ชันอะไรบ้าง จากนั้นก็ค่อยๆ เขียนและทดสอบฟังก์ชัน และค่อยนำมารวมกันเป็นโปรแกรม
  - ต้องมีฟังก์ชันเข้ารหัส (encrypt) โดยมีพารามิเตอร์จำนวน 2 ตัว คือ string ข้อความ และ จำนวนที่ shift โดย return เป็น string ข้อความที่เข้ารหัสแล้ว

```
plain_text = "hello"
shift = 5
cipher_text = "mjqqt"
print output: "The encoded text is mjqqt"
```

Case Study : Caesar cipher

- ฟังก์ชัน encrypt สามารถเขียนเป็นโค้ดเทียมได้ดังนี้
  - cipher_text เท่ากับ “ ”
  - สำหรับแต่ละตัวอักษรใน plain_text (string ที่รับเข้ามา)
    - ให้หาว่าเป็นตัวอักษรลำดับเท่าใดใน a-z เก็บค่าไว้ที่ position
    - จากตำแหน่งที่ได้ให้บวกค่า shift amount เข้าไป เพื่อหาตำแหน่งใหม่
    - ให้นำตัวอักษรในตำแหน่งใหม่เพิ่มเข้าไปใน cipher_text
  - สำหรับการหาว่าตัวอักษรเป็นลำดับเท่าใด จะใช้ method ที่ชื่อ index

Case Study : Caesar cipher

- ฟังก์ชัน encrypt สามารถเขียนเป็นโปรแกรม และ ทดสอบดังนี้
- เพื่อป้องกันปัญหา ตัวใหญ่ ตัวเล็ก จึงใช้ method .lower เพื่อแปลงเป็นตัวเล็ก

main.py

```
1  import string
2
3  def encrypt(plain_text, shift_amount):
4      cipher_text = ""
5      for letter in plain_text:
6          position = string.ascii_lowercase.index(letter)
7          new_position = position + shift_amount
8          new_letter = string.ascii_lowercase[new_position]
9          cipher_text += new_letter
10     return cipher_text
11
12 text = input("Type your message: ").lower()
13 shift = int(input("Type the shift number: "))
14 text = encrypt(text, shift)
15 print(f"The encoded text is {text}")
```

Console Shell

```
Type your message: hello
Type the shift number: 5
The encoded text is mjqqt
```

Case Study : Caesar cipher

- **Exercise 5.4 :** ให้นักศึกษาทดลองเขียนโปรแกรม decrypt เพื่อถอดรหัส

```
12 ▼ def decrypt(cipher_text, shift_amount):
13     plain_text = ""
14 ▼   for letter in cipher_text:
15       position = string.ascii_lowercase.index(letter)
16       new_position = position - shift_amount
17       plain_text += string.ascii_lowercase[new_position]
18     return plain_text
19
20   text = input("Type your message: ").lower()
21   shift = int(input("Type the shift number: "))
22   text = encrypt(text, shift)
23   print(f"The encoded text is {text}")
24   text = decrypt(text, shift)
25   print(f"The decoded text is {text}")
```

Case Study : Caesar cipher

- **Exercise 5.5 :** ถ้าดูจากโปรแกรมจะเห็นว่าส่วน encrypt และ decrypt มีโปรแกรมที่คล้ายกัน ซึ่งเมื่อเราเจอแบบนี้ ควรจะหาวิธีนำมารวมกัน
- ให้ นศ. เขียนฟังก์ชัน ชื่อ caesar โดยให้รับพารามิเตอร์เพิ่มอีก 1 ตัว คือ direction เพื่อบอกว่าจะ encrypt หรือ decrypt
- ฟังก์ชัน caesar ต้องรองรับกรณีที่มีเว้นวรรคในคำ เช่น “ce computer”
- ฟังก์ชัน caesar เมื่อพบเครื่องหมายให้ข้ามไปไม่ต้อง shift
- ให้ป้องกันกรณีที่ใส่เลข shift เกิน 26
- กรณีที่เป็นตัว z และมีการ shift ให้กลับไปที่ a ใหม่
- ให้เขียนโปรแกรมถามว่าจะทำงานต่อหรือไม่

Case Study : Caesar cipher

main.py ×

```python
import string

def caesar(start_text, shift_amount, cipher_direction):
    end_text = ""
    if cipher_direction == "decode":
        shift_amount *= -1
    for char in start_text:
        if char in string.ascii_lowercase:
            position = string.ascii_lowercase.index(char)
            new_position = (position + shift_amount) % 26
            end_text += string.ascii_lowercase[new_position]
        else:
            end_text += char
    return end_text

should_end = False
while not should_end:
    direction = input("Type 'encode' to encrypt, type 'decode' to decrypt: ")
    text = input("Type your message: ").lower()
    shift = int(input("Type the shift number: "))

    shift = shift % 26

    text = caesar(start_text=text, shift_amount=shift, cipher_direction=direction)
    print(f"Here's the {direction}d result: {text}")
    restart = input("Type 'yes' if you want to go again. Otherwise type 'no'.")
    if restart == "no":
        should_end = True
        print("Goodbye")
```

For your attention