01076103, 01076104

Programming Fundamental

Programming Project

Dictionary, Set

Dictionary

- Dictionary เป็นโครงสร้างข้อมูลอีกชนิดหนึ่ง ใช้ { } ในการกำหนด
- Dictionary มีลักษณะของการจับคู่ ซึ่งอ้างอิงโดย key : value การใช้งานจะเป็นรูปแบบ { key : value, key : value }
- Dictionary เป็นโครงสร้างแบบไม่มีลำดับ ดังนั้นจะใช้ index ในการระบุตำแหน่งที่มีลักษณะเป็นตัวเลขเหมือนกับ list ไม่ได้ ในการอ้างถึงข้อมูลจะใช้ค่า key ในการอ้าง
- Dictionary เป็นข้อมูลแบบ Mutable

Dictionary

- ตัวอย่างการสร้าง dictionary

main.py

```python
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}

# using dict()
my_dict = dict({1:'apple', 2:'ball'})

# from sequence having each item as a pair
my_dict = dict([(1,'apple'), (2,'ball')])
```

Dictionary

- การอ้างถึงสมาชิกใน dictionary จะใช้ key เป็นหลัก
- สามารถอ้างโดยใช้ Index [ ] (ถ้าไม่พบจะ Error) , หรือใช้ method get (ถ้าไม่พบ -> None)

main.py

```
1  # get vs [] for retrieving elements
2  my_dict = {'name': 'Jack', 'age': 26}
3
4  # Output: Jack
5  print(my_dict['name'])
6
7  # Output: 26
8  print(my_dict.get('age'))
9
10 # Trying to access keys which doesn't exist throws error
11 # Output None
12 print(my_dict.get('address'))
13
14 # KeyError
15 print(my_dict['address'])
```

Console Shell

```
Jack
26
None
Traceback (most recent call last):
  File "main.py", line 15, in <module>
    print(my_dict['address'])
KeyError: 'address'
>
```

Dictionary

- การเปลี่ยน หรือ เพิ่มข้อมูล
- การเปลี่ยนให้ใช้ในรูปแบบ dict[key] = value
- การเพิ่มก็เช่นเดียวกัน โดย dictionary จะตรวจสอบ key ที่มีอยู่เดิม ถ้ามีอยู่เดิม ก็จะเป็นการเปลี่ยน แต่ถ้าไม่มี key นั้น ก็จะเป็นการเพิ่ม

main.py ×

```
1  # Changing and adding Dictionary Elements
2  my_dict = {'name': 'Jack', 'age': 26}
3
4  # update value
5  my_dict['age'] = 27
6  print(my_dict)
7
8  # add item
9  my_dict['address'] = 'Downtown'
10 print(my_dict)
```

Console Shell

```
{'name': 'Jack', 'age': 27}
{'name': 'Jack', 'age': 27, 'address': 'Downtown'}
```

Dictionary

- การลบข้อมูล 1) ทำโดยใช้ method pop() จะคืนค่าเป็น value และลบ 2) ทำโดยใช้ method popitem() จะคืนค่า (key, value) 3) method clear() เป็นการลบทั้งหมด

main.py

```
1  squares = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
2
3  # remove a particular item, returns its value
4  print(squares.pop(4))
5  print(squares)
6
7  # remove an arbitrary item, return (key,value)
8  print(squares.popitem())
9  print(squares)
10
11 # remove all items
12 squares.clear()
13 print(squares)
14
15 # delete the dictionary itself
16 del squares
17 print(squares)
```

Console Shell

```
16
{1: 1, 2: 4, 3: 9, 5: 25}
(5, 25)
{1: 1, 2: 4, 3: 9}
{}
Traceback (most recent call last):
  File "main.py", line 17, in <module>
    print(squares)
NameError: name 'squares' is not defined
>
```

Dictionary

- fromkeys() เป็น method ที่เพิ่มข้อมูลเข้าไปใน dictionary โดยใช้กรณีที่มีหลาย key แต่มี value เดียวกัน เช่น การเพิ่มวิชา โดยมีคะแนนเป็น 0 ทุกวิชา
- items() เป็น method ที่ส่งคืน key, value ของแต่ละสมาชิกใน dictionary กลับคืนมา เราสามารถแยก key กับ value ได้ โดยเอาตัวแปร 2 ตัวมารับ เช่น
  for k, v in marks.items()
- เราสามารถจะดึงเฉพาะค่า key หรือ value ได้โดยใช้ method keys() หรือ values()

main.py

```
1  # Dictionary Methods
2  marks = {}.fromkeys(['Math', 'English', 'Science'], 0)
3  print(marks)
4
5  for item in marks.items():
6      print(item)
7
8  print(list(sorted(marks.keys())))
```

Console Shell

```
{'Math': 0, 'English': 0, 'Science': 0}
('Math', 0)
('English', 0)
('Science', 0)
['English', 'Math', 'Science']
```

Dictionary

- ตัวอย่างการใช้งาน dictionary โดยการสร้างเครื่องคิดเลขอย่างง่าย
- จะเริ่มจากสร้าง ฟังก์ชัน บวก ลบ คูณ หาร
- และสร้าง dictionary เพื่อเก็บ
  - key : เครื่องหมาย
  - value : function

```
13 operations = {'+' : add,
14                '-' : subtract,
15                '*' : multiply,
16                '/' : devide}
17
```

main.py ×

```
1  def add(n1, n2):
2      return n1+n2
3
4  def subtract(n1, n2):
5      return n1-n2
6
7  def multiply(n1, n2):
8      return n1*n2
9
10 def devide(n1, n2):
11     return n1/n2
12
```

Dictionary

- จากนั้นก็เขียนโปรแกรมรับตัวเลขมาคำนวณ จะเห็นว่าในบรรทัดที่ 27 เราสามารถ assign ตัวแปรให้มาชี้ฟังก์ชันที่ตรงกับเครื่องหมายที่เลือก และ สั่งให้ทำงานในบรรทัดที่ 26

main.py

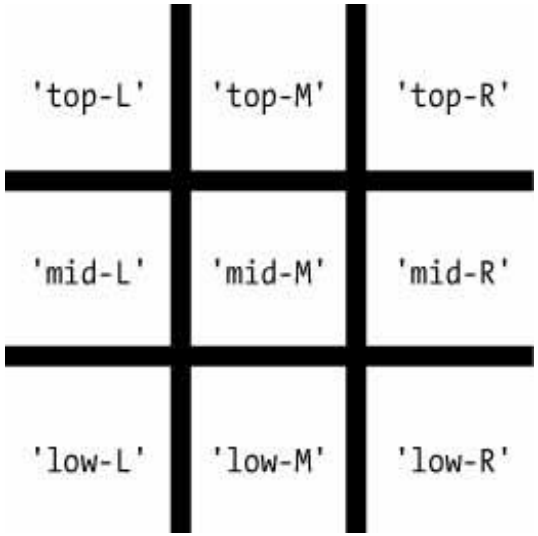
```
18 num1 = float(input("What's the first number?: "))
19 for symbol in operations:
20     print(symbol)
21
22 should_continue = True
23
24 while should_continue:
25   operation_symbol = input("Pick an operation: ")
26   num2 = float(input("What's the next number?: "))
27   calculation_function = operations[operation_symbol]
28   answer = calculation_function(num1, num2)
29   print(f"{num1} {operation_symbol} {num2} = {answer}")
30
31   if input(f"Type 'y' to continue calculating with {answer}, or
type 'n' to start a new calculation: ") == 'y':
32     num1 = answer
33   else:
34     should_continue = False
```

Console Shell

```
What's the first number?: 25
+
-
*
/
Pick an operation: *
What's the next number?: 25
25.0 * 25.0 = 625.0
Type 'y' to continue calculating with 625.0,
art a new calculation:
```

Dictionary

- ดูอีกตัวอย่าง คือ โปรแกรม X-O แต่ต่างประเทศเรียก Tic-Tac-Toe เกมนี้จะมีตารางตามรูป




```
board = {'top-L': ' ', 'top-M': ' ', 'top-R': ' ',
         'mid-L': ' ', 'mid-M': ' ', 'mid-R': ' ',
         'low-L': ' ', 'low-M': ' ', 'low-R': ' '}
```

- ในการใช้ dictionary เราจะใช้วิธีการกำหนดตำแหน่งให้กับแต่ละช่องเป็นชื่อ ตามรูป โดยจะใช้ชื่อนี้เป็น key และ value คือ o หรือ x
- เริ่มต้น เราก็สร้างบอร์ดตามโปรแกรม โดยมีค่าเป็นว่าง

Dictionary

- จากนั้นทำฟังก์ชัน print ดังรูป
- คำถาม : เราสามารถใช้ for loop ในการแสดงบอร์ดได้หรือไม่

main.py ×

```
1  board = {'top-L': ' ', 'top-M': ' ', 'top-R': ' ',
2           'mid-L': ' ', 'mid-M': ' ', 'mid-R': ' ',
3           'low-L': ' ', 'low-M': ' ', 'low-R': ' '}
4
5  def print_board(board):
6      print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-R'])
7      print('-+-+-')
8      print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-R'])
9      print('-+-+-')
10     print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-R'])
11
12 print_board(board)
```

Console

```
  |  |
-+-+-
  |  |
-+-+-
  |  |
```

Dictionary

- **Ex 7.1** จงเขียนโปรแกรมรับข้อมูลคะแนนของนักศึกษา 1 คน โดยจะเรียนได้หลายวิชา แต่ละวิชาจะเก็บค่าคะแนนเพียง 1 ค่า โดยใช้ dictionary
  - ให้สร้าง subject_dict เป็น dictionary แบบ global
  - function add_score(subject, score) จะได้ dictionary ที่มีวิชาของนักศึกษาพร้อมคะแนน
    - add_score (‘python’, 80) => {‘python’ :80 }
    - add_score (‘calculus’, 60) => {‘python’ :80, ‘calculus’ :60 }

Dictionary

- **Ex 7.1** จงเขียนโปรแกรมรับข้อมูลคะแนนของนักศึกษา 1 คน โดยจะเรียนได้หลายวิชา แต่ละวิชาจะเก็บค่าคะแนนเพียง 1 ค่า โดยใช้ dictionary
  - function show_student_score() โดยนำข้อมูลมาแสดง ในรูปแบบ

    python : 50

    calculus : 55 (ส่วน วิชา กำหนดพื้นที่ 15 ส่วนคะแนน 10)
  - เขียนฟังก์ชันหาค่าเฉลี่ยของคะแนนในทุกรายวิชาของนักเรียนคนนั้น

    function calc_average_score() โดยส่งค่าคืนมาเป็น คะแนนเฉลี่ย ทศนิยม 2 ตำแหน่ง

Dictionary

- การสร้าง Dictionary 2 มิติ
- จาก Exercise ที่ผ่านมา เราสร้าง Dictionary แบบ 1 มิติ

  sdict1 = {'python’: 40, 'calculus’: 45}

  มี 2 Element คือ sdict[0] = 'python': 50 และ sdict[1] = 'calculus': 55

  หากเรามี sdict2 = {'python’: 50, 'calculus’: 55}
- ถ้าเรากำหนด อีก dictionary เป็น

  st_score = {‘65015001’: sdict1, {‘65015002’: sdict2 }
- ก็จะกลายเป็น Dictionary 2 มิติ หรือ Nested Dictionary ทันที
- ในการอ้างถึง ถ้าอ้างถึง st_score[‘65015001’] จะหมายถึง sdict1 หรือ {'python’: 40, 'calculus’: 45} ถ้าอ้าง st_score[‘65015001’]['python’] ก็จะหมายถึง 40

Dictionary

- สำหรับการเพิ่มข้อมูลใน dictionary 2 มิติ อาจทำได้ดังนี้
  - st_score[‘65015003’] = {}
  - st_score[‘65015003’][‘python’] = 30
  - st_score[‘65015003’][‘calculus’] = 35

- หรือ

  st_score[‘65015004’] = {'python’: 42, 'calculus’: 25}

- ในการเขียนโปรแกรมให้แยกให้ถูกว่า อันไหนเป็น Key และ Value ของอันไหน

Dictionary

- Ex 7.2 ให้ขยายความสามารถของโปรแกรมตามข้อ 7.1 โดยให้รองรับนักศึกษาหลายคน
  - function add_score(student_id, subject, score) โดยถ้าเป็นวิชาเดิมจะถือเป็นการ update score

```
add_score('64015001', 'python', 40)
add_score('64015002', 'python', 50)
add_score('64015001', 'calculus', 45)
add_score('64015002', 'calculus', 55)
```

  - จะได้ dict เป็น
  - **{'64015001': {'python': 40, 'calculus', 45} ,**
    '64015002': {'python': 50, 'calculus', 55} }

Dictionary

- **Ex 7.2** ให้ขยายความสามารถของโปรแกรมตามข้อ 7.1 โดยให้รองรับนักศึกษาหลายคน
  - function show_student_score(65015001) โดยนำข้อมูลมาแสดง ในรูปแบบ

    65015001 : python : 40

    65015001 : calculus : 50

    (ส่วน วิชา กำหนดพื้นที่ 10 ส่วนคะแนน 5)
  - เขียนฟังก์ชันหาค่าเฉลี่ยของคะแนนในทุกรายวิชาของนักเรียนคนนั้น function calc_average_score(65015001) โดยส่งค่าคืนมาเป็น คะแนนเฉลี่ย ทศนิยม 2 ตำแหน่ง

Dictionary comprehension

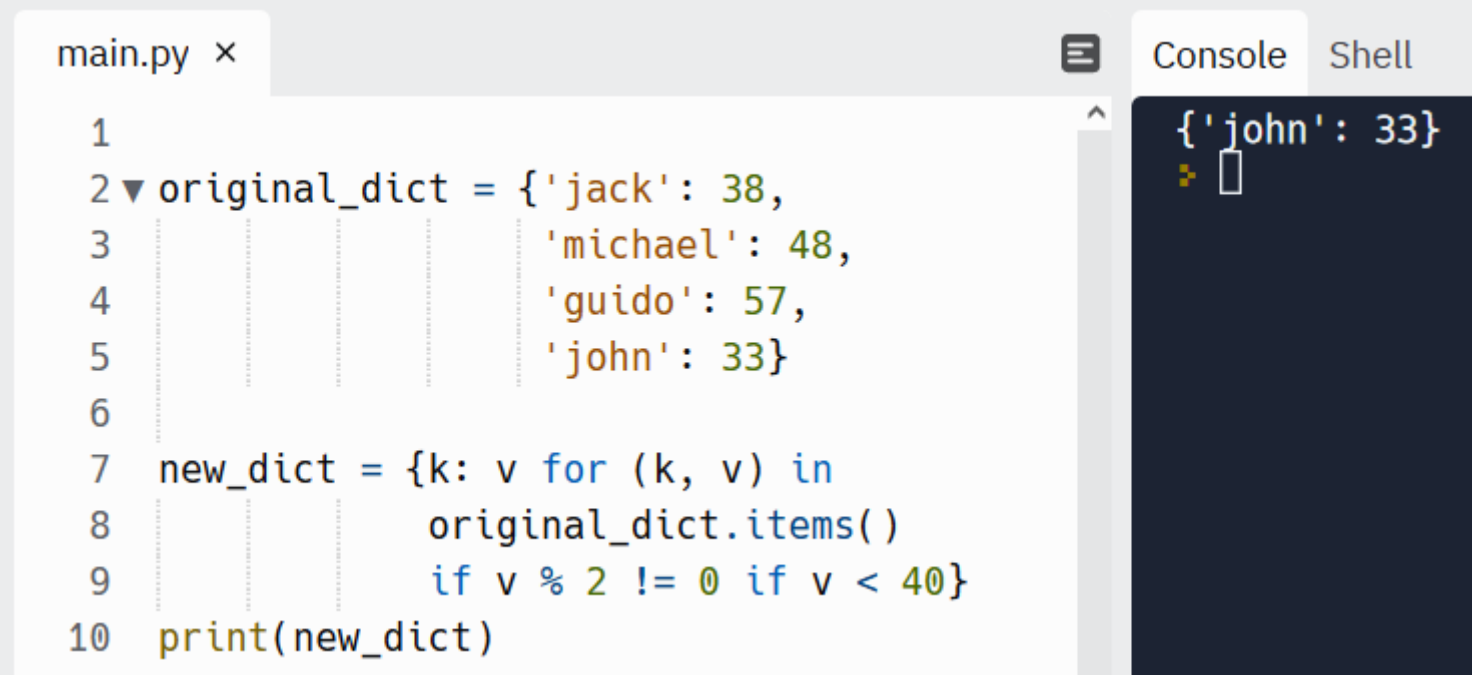
- กรณีของ dictionary comprehension ก็เช่นเดียวกัน แต่เป็นการสร้างใส่ dictionary แทนที่จะเป็น list แต่เนื่องจาก dictionary มีความซับซ้อนมากกว่า ดังนั้นจึงทำให้อ่านยากขึ้นไปอีก
- รูปแบบการทำงานของ dictionary comprehension มีดังนี้

```
dictionary = {key: value for vars in iterable}
```

```
{ key: value for vars in iterable }
  |     |       |         |
{ num: num*num for num in range(1, 11) }
```

Dictionary comprehension

- ตัวอย่าง กรณีมี 2 เงื่อนไข คือ ต้องเป็นเลขคี่ และ ต้องมี value น้อยกว่า 40



Dictionary comprehension

- ตัวอย่างกรณี ใช้ if else

main.py

```
original_dict = {'jack': 38,
                 'michael': 48,
                 'guido': 57,
                 'john': 33}

new_dict_1 = {k: ('old' if v > 40
                  else 'young')
                  for (k, v) in
                     original_dict.items()}

print(new_dict_1)
```

Console Shell

```
{'jack': 'young', 'michael': 'old', 'guido': 'old', 'john': 'young'}
```

Dictionary comprehension

- **ตัวอย่าง** เป็นการสร้าง dictionary ใหม่ ที่เก็บราคาเครื่องดื่ม จากเดิมเป็น dollar เป็นปอนด์

main.py ×

```
1
2 old_price = {'milk': 1.02,
3               'coffee': 2.5,
4               'bread': 2.5}
5
6 dollar_to_pound = 0.76
7 new_price = {item:
  value*dollar_to_pound for (item, value)
  in old_price.items()}
8 print(new_price)
```

Console Shell

```
{'milk': 0.7752, 'coffee': 1.9, 'bread': 1.9}
```

Set

- Set เป็นโครงสร้างข้อมูลอย่างหนึ่งใน Python
- คำว่า "เซต" หมายถึง กลุ่ม หมู่ เหล่า กอง ฝูง ชุด และเมื่อกล่าวถึงเซตของสิ่งใดๆ จะทราบได้ทันทีว่าในเซตนั้นมีอะไรบ้าง เราเรียกสิ่งที่อยู่ในเซตว่า 'สมาชิก' โดย Set จะไม่มีสมาชิกที่ซ้ำกัน
- Set เป็นโครงสร้างแบบไม่มีลำดับเช่นเดียวกับ Dictionary และเป็น Mutable
- Set จะใช้เครื่องหมาย { } ในการกำหนดขอบเขต และ สมาชิกของ Set (อย่าสับสนกับ Dict)

Set

- ใน Set จะไม่มีสมาชิกซ้ำกัน หากเรากำหนดซ้ำ จะถูกตัดออก
- เราสามารถสร้าง Set จาก List ได้ แต่ใน Set ต้องไม่มีสมาชิกที่เป็น Mutable

main.py

```
1  # set cannot have duplicates
2  my_set = {1, 2, 3, 4, 3, 2}
3  print(my_set)
4
5  # we can make set from a list
6  my_set = set([1, 2, 3, 2])
7  print(my_set)
8
9  # set cannot have mutable items
10 # here [3, 4] is a mutable list
11 # this will cause an error.
12 my_set = {1, 2, [3, 4]}
```

Console Shell

```
{1, 2, 3, 4}
{1, 2, 3}
Traceback (most recent call last):
  File "main.py", line 15, in <module>
    my_set = {1, 2, [3, 4]}
TypeError: unhashable type: 'list'
```

Set

- ไม่สามารถใช้ index กับ set ได้
- การเพิ่มข้อมูลอาจใช้ add (1 สมาชิก) หรือ update (หลายสมาชิก)

main.py

```
1  my_set = {1, 3}
2  print(my_set)
3
4  #my_set[0] # Error set don't support index
5
6  # add an element
7  my_set.add(2)
8  print(my_set)
9
10 # add multiple elements
11 my_set.update([2, 3, 4])
12 print(my_set)
13
14 # add list and set
15 my_set.update([4, 5], {1, 6, 8})
16 print(my_set)
```

Console Shell

```
{1, 3}
{1, 2, 3}
{1, 2, 3, 4}
{1, 2, 3, 4, 5, 6, 8}
```

Set

- การนำสมาชิกออก สามารถใช้ discard และ remove โดย discard หากไม่มีสมาชิกนั้นอยู่จะไม่แสดง error แต่หากใช้ remove จะแสดง KeyError

main.py ×

```
1  my_set = {1, 3, 4, 5, 6}
2  print(my_set)
3
4  # discard an element
5  my_set.discard(4)
6  print(my_set)
7
8  # remove an element
9  my_set.remove(6)
10 print(my_set)
11
12 # discard an element, not present in my_set
13 my_set.discard(2)
14 print(my_set)
15
16 # remove an element
17 # not present in my_set, you will get an error.
18 my_set.remove(2)
```

Console Shell

```
{1, 3, 4, 5, 6}
{1, 3, 5, 6}
{1, 3, 5}
{1, 3, 5}
Traceback (most recent call last):
  File "main.py", line 18, in <module>
    my_set.remove(2)
KeyError: 2
```

Set

- สามารถใช้ pop ได้เช่นเดียวกับ list แต่ตัวที่ถูกนำออก จะไม่แน่นอน เพราะ set เป็นแบบไม่มีลำดับ และสามารถใช้ clear ในการล้าง set ทั้งหมดได้

main.py

```
1  my_set = set("HelloWorld")
2  print(my_set)
3
4  # pop an element, random element
5  print(my_set.pop())
6
7  # pop another element
8  my_set.pop()
9  print(my_set)
10
11 # clear my_set
12 my_set.clear()
13 print(my_set)
```

Console Shell

```
{'o', 'r', 'H', 'd', 'l', 'e', 'W'}
o
{'H', 'd', 'l', 'e', 'W'}
set()
```

Set

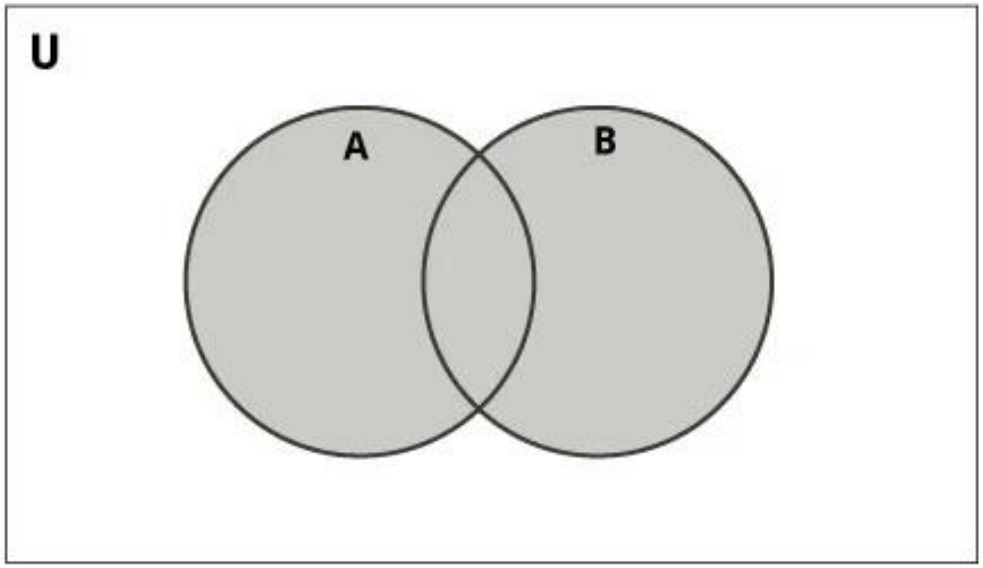
- การทำงานแบบ set : Union คือ การรวมสมาชิกของ set ทั้งสอง (ถ้ามีซ้ำจะนับแค่ 1)

main.py ×

```
1  A = {1, 2, 3, 4, 5}
2  B = {4, 5, 6, 7, 8}
3  print(A | B)
```

Console Shell

```
{1, 2, 3, 4, 5, 6, 7, 8}
```



Set

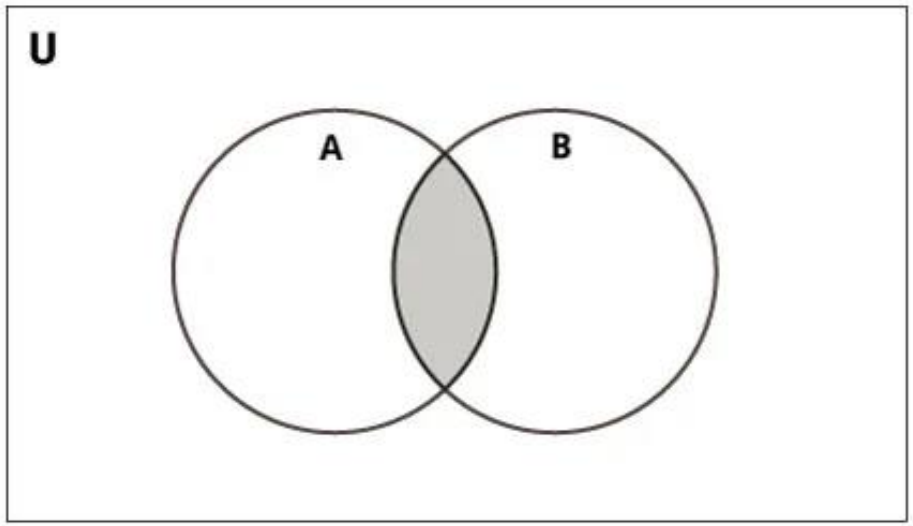
- การทำงานแบบ set : Intersection คือ การเอาเฉพาะสมาชิกที่เป็นสมาชิกของ set ทั้งสอง

main.py ×

```
1  A = {1, 2, 3, 4, 5}
2  B = {4, 5, 6, 7, 8}
3  print(A & B)
```

Console Shell

```
{4, 5}
>
```



Set

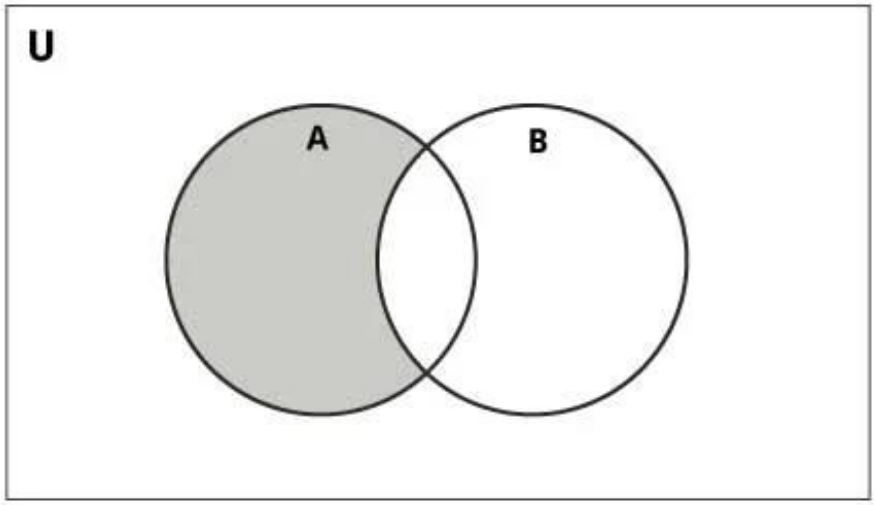
- การทำงานแบบ set : Difference คือ การเอาเฉพาะสมาชิกที่ไม่ได้เป็นสมาชิกของอีก set (A-B ไม่เท่ากับ B-A)

main.py ×

```
1  A = {1, 2, 3, 4, 5}
2  B = {4, 5, 6, 7, 8}
3  print(A - B)
```

Console Shell

```
{1, 2, 3}
>
```



Set

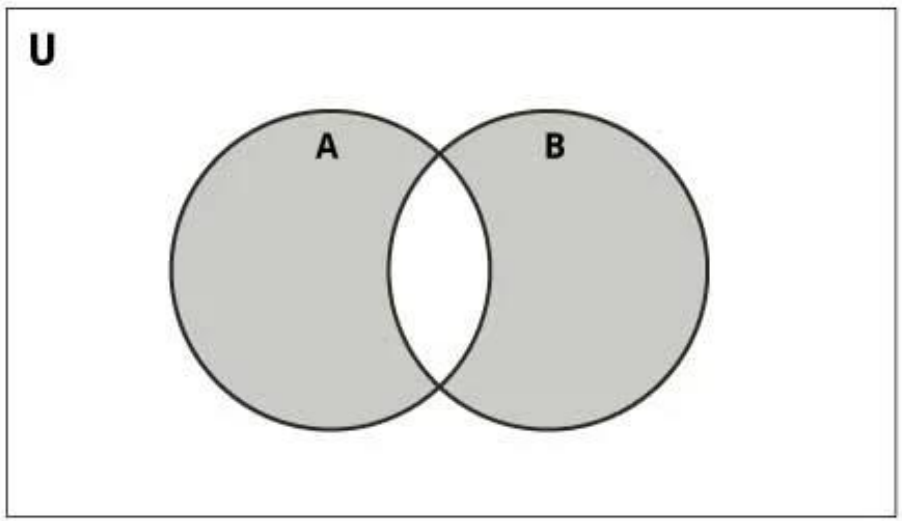
- การทำงานแบบ set : Symmetric Difference คือ การเอาเฉพาะสมาชิกไม่ Intersection

main.py ×

```
1  A = {1, 2, 3, 4, 5}
2  B = {4, 5, 6, 7, 8}
3  print(A ^ B)
```

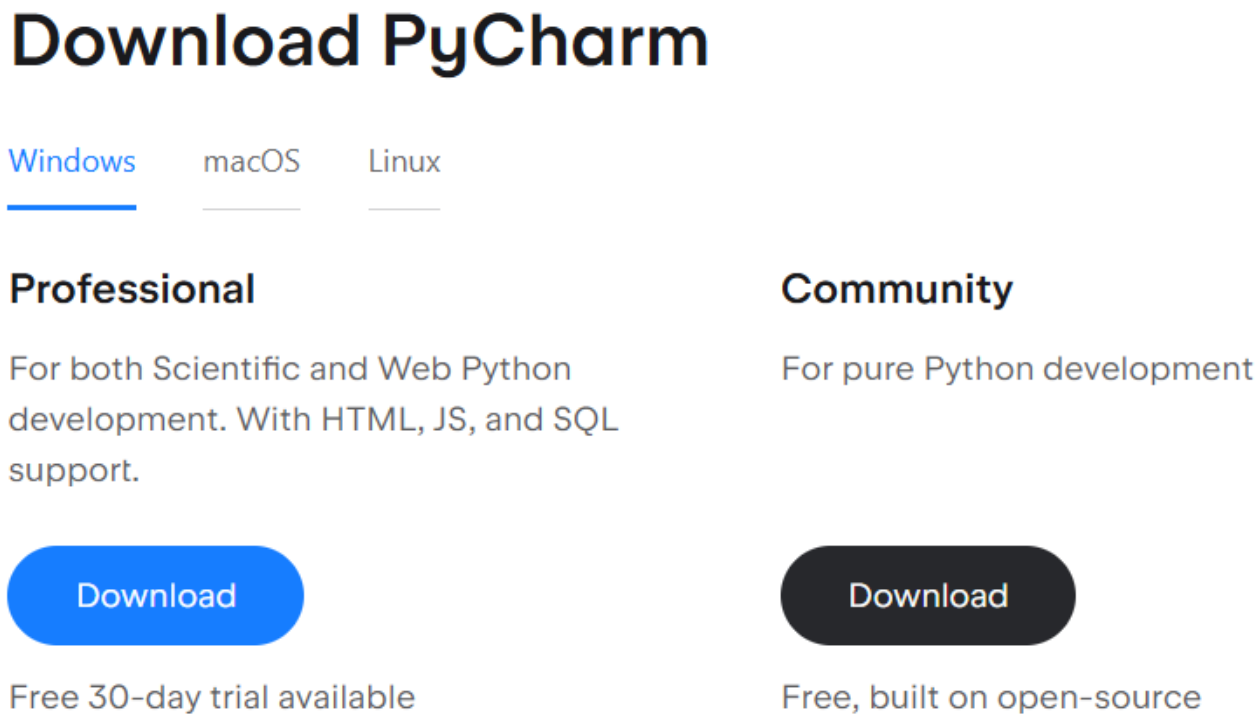
Console Shell

```
{1, 2, 3, 6, 7, 8}
```



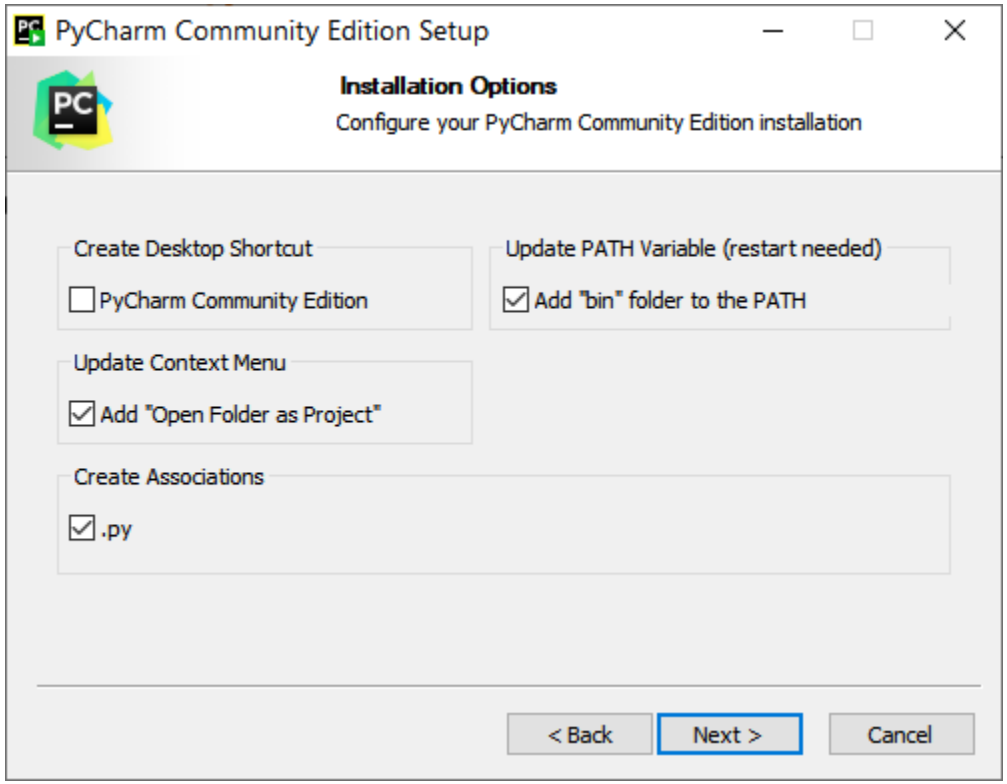
การติดตั้ง Pycharm

- เข้าไปที่เว็บ https://www.jetbrains.com/pycharm/download/ แล้ว download community edition



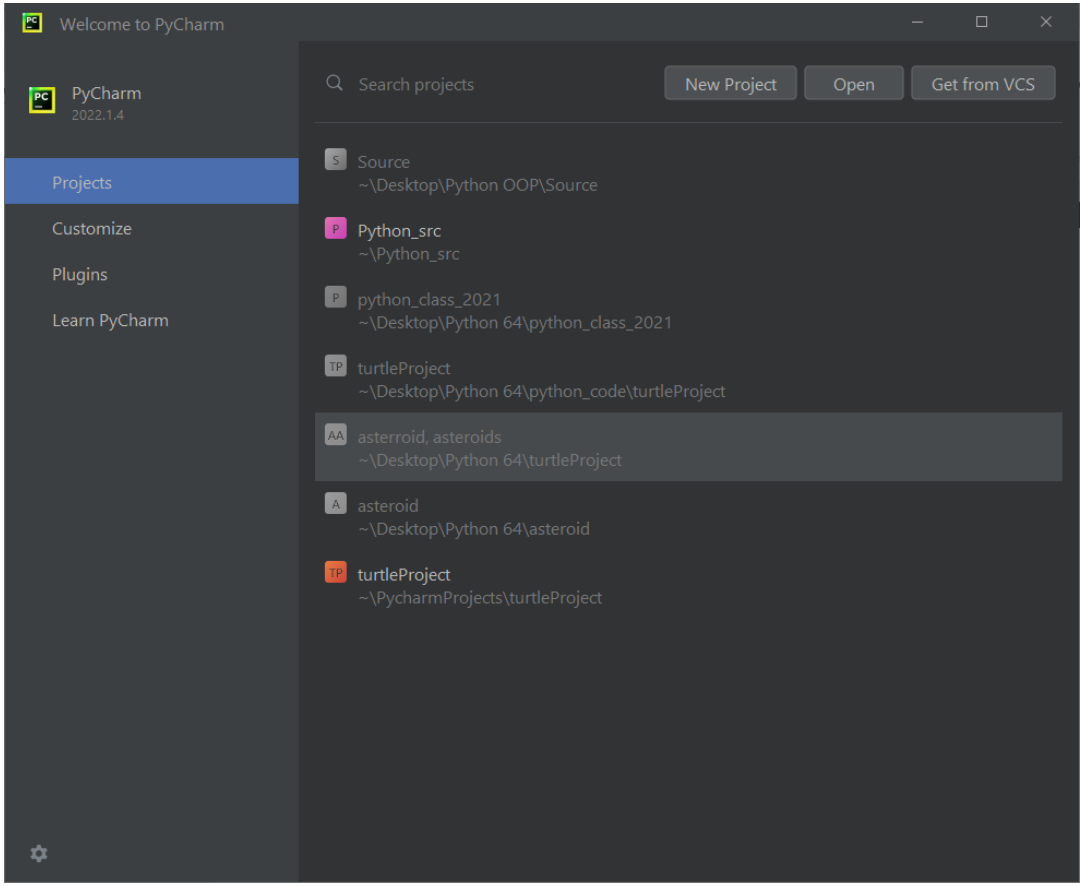
การติดตั้ง Pycharm

- ดำเนินการติดตั้งจนเสร็จ



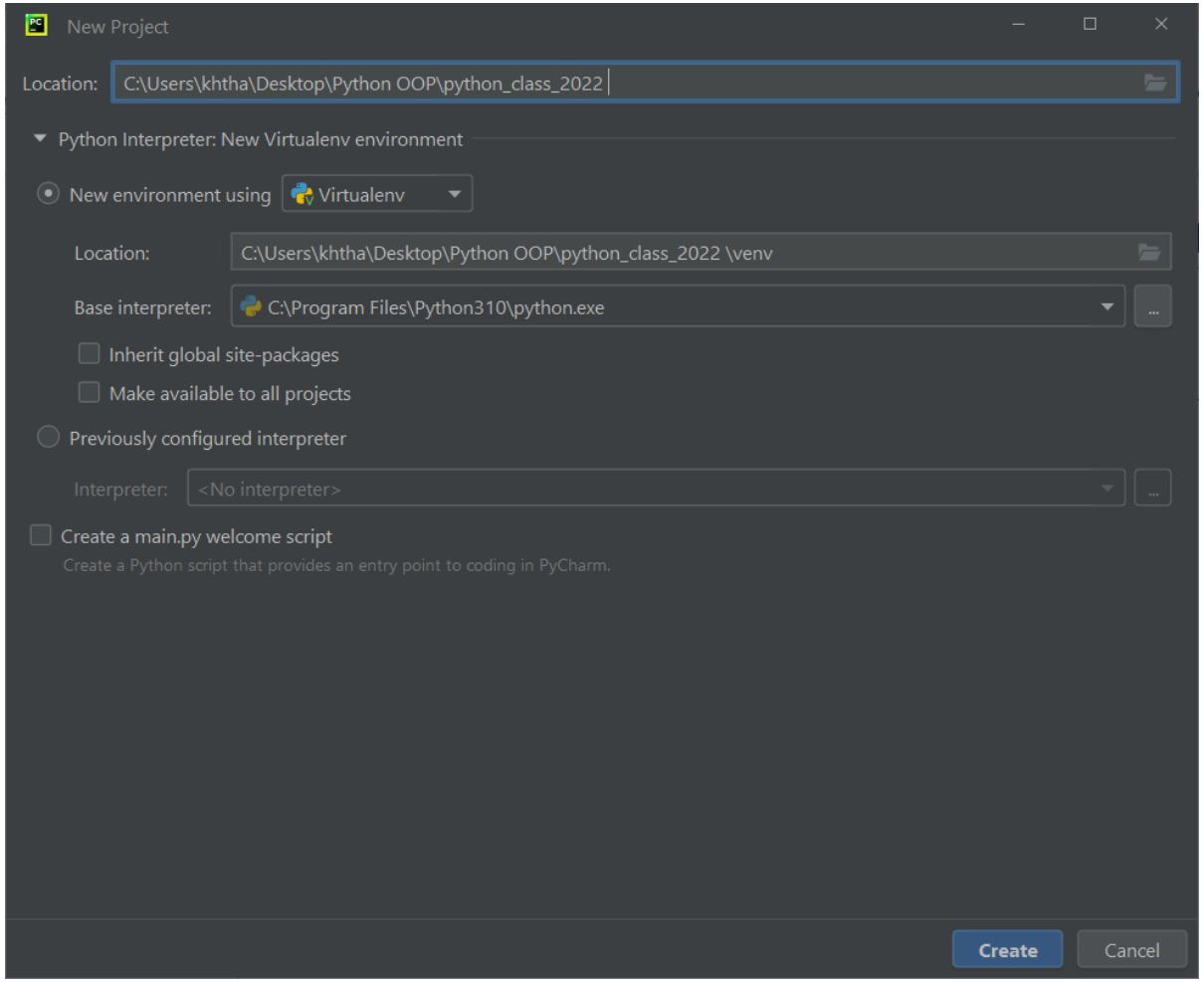
การติดตั้ง Pycharm

- เมื่อเปิดจะแสดงหน้าจอนี้ ถ้ายังไม่เคยใช้มาก่อนให้เลือก New Project แต่ถ้ามี Project อยู่แล้ว ก็เลือก Open



การติดตั้ง Pycharm

ตั้งชื่อ Project แล้วกด Create



Debugging

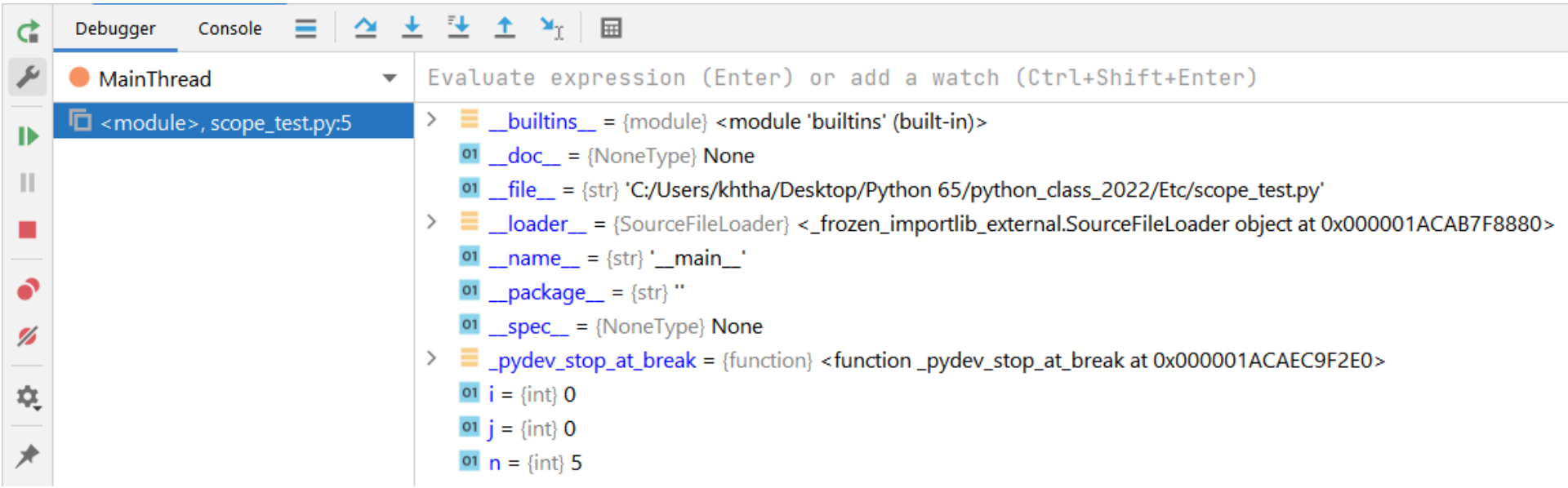
- เป็นทักษะที่สำคัญมากของ Developer
- เมื่อโปรแกรมเริ่มมีขนาดใหญ่ขึ้น และ ซับซ้อนขึ้น เมื่อมีความผิดพลาด (Bug) เกิดขึ้นในการเขียนโปรแกรม ก็จะยากในการหาจุดผิดพลาดมากขึ้นเรื่อยๆ
- ดังนั้นคนที่มีทักษะในการหาข้อผิดพลาดเก่งๆ ก็จะลดเวลาในการทำงานได้
- โปรแกรมเมอร์ที่เก่ง จะต้อง debug เก่งด้วย

Debugging

- ใน PyCharm จะมี Debugger โดยมีความสามารถดังนี้
  - กำหนด Breakpoint คือ จุดที่ต้องการให้โปรแกรมหยุดทำงานเมื่อทำงานถึงจุดนี้ วิธีการ คือ คลิกบริเวณเลขบรรทัด จะมีจุดแดงเพิ่มเข้ามา เมื่อเรา debug (กด icon รูปแมลง) โปรแกรมจะหยุดทำงานที่จุดนี้

```
1   n = 5   n: 5
2   for i in range(n):   i: 0
3       for j in range(n):   j: 0
4           if i==0 or i==n-1 or j==0 or j==n-1:
5               print('*',end='')
6           else:
7               print(' ',end='')
8       print()
```

Debugging



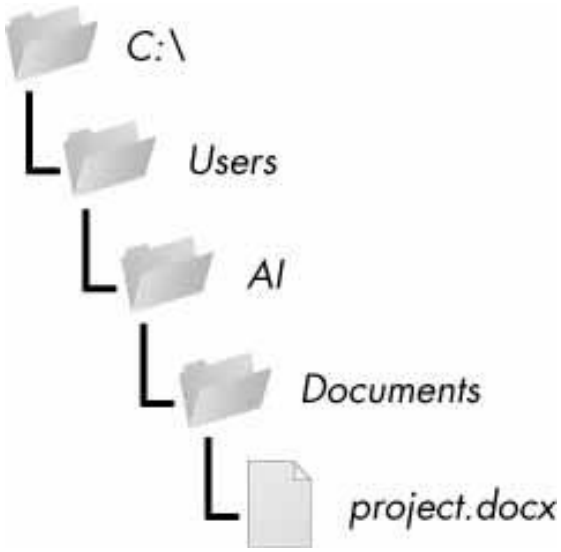
- หมายถึงทำงานต่อ หยุดโปรแกรม step over รันบรรทัดนี้ ถ้ามี fn ให้ทำทั้ง fn รันบรรทัดนี้ แต่ถ้ามี fn ให้เข้าไปใน fn รันจนถึง cursor
- ที่หน้าจอด้านขวา จะเห็นว่ามีตัวแปรแสดงอยู่ ทำให้เรารู้ว่า ในขั้นตอนนี้ตัวแปรมีค่าใด

Debugging

- ก่อนที่เราจะใช้ความสามารถในการ debug ของโปรแกรม เราต้องคาดการณ์ได้ก่อนว่าโปรแกรมจะมีพฤติกรรม หรือ การทำงานอย่างไร
- การ debug คือ การยืนยันพฤติกรรมว่าโปรแกรมทำงานตามที่คิดหรือไม่
- โดยในแต่ step หรือ แต่ละ loop ตัวแปรแต่ละตัวแปรมีการเปลี่ยนแปลงไปตามที่คิดหรือไม่
- ถ้าตัวแปรเปลี่ยนไม่เปลี่ยนไปตามที่คาดไว้ เราจะสามารถทราบปัญหาที่เกิดขึ้นและสามารถแก้ไขได้เร็วขึ้น

File & Directory

- การเก็บข้อมูลลงในตัวแปร จะเก็บได้แค่ชั่วคราวเท่านั้น หากเราต้องการเก็บข้อมูลถาวร เราจะต้องเก็บข้อมูลลงในไฟล์
- องค์ประกอบของชื่อไฟล์ จะประกอบด้วย filename และ path



File & Directory

- กรณีทำงานกับไฟล์หลายไฟล์ใน folder หรือ path เดียวกัน จะแยกเก็บระหว่างชื่อไฟล์และ path เช่น สมมติว่าใน c:\work\excel มีไฟล์ excel1.xlsx , excel2.xlsx และ excel3.xlsx
- หากเราจะสร้าง path + filename จะใช้ library pathlib ตามรูป

```
1  from pathlib import Path
2
3  myFiles = ['accounts.txt',
4             'details.csv',
5             'invite.docx']
6
7  for filename in myFiles:
8      print(Path('C:\work\excel', filename))
9
```

```
"C:\Users\khtha\Desktop\Python OOP\
C:\work\excel\accounts.txt
C:\work\excel\details.csv
C:\work\excel\invite.docx

Process finished with exit code 0
```

File & Directory

- กรณีที่การทำงานประกอบด้วย directory และ subdirectory หลากหลาย โดยมีความจำเป็นจะต้องสร้าง path จากคำต่างๆ ก็สามารถสร้างได้เช่นกัน
- จะใช้ / ในการต่อระหว่างแต่ละส่วนของ path

```
1  from pathlib import Path
2
3  for n in range(1,13):
4      a = Path('c:\work') / 'excel' / str(n)
5      print(a)
6
7
8
```

```
"C:\Users\khtha\Desktop\Python
c:\work\excel\1
c:\work\excel\2
c:\work\excel\3
c:\work\excel\4
c:\work\excel\5
c:\work\excel\6
c:\work\excel\7
c:\work\excel\8
c:\work\excel\9
c:\work\excel\10
c:\work\excel\11
c:\work\excel\12
```

File & Directory

- ยังมีการทำงานอื่นๆ ที่เกี่ยวข้องกับ directory
  - คำสั่ง cwd ใช้แสดง path ปัจจุบัน
  - os.chdir จะใช้ในการเปลี่ยน path

main.py ×

```
1  from pathlib import Path
2  import os
3  path = Path.cwd()
4  print(path)
5  os.chdir('..')
6  path = Path.cwd()
7  print(path)
```

Console Shell

```
/home/runner/file
/home/runner
>
```

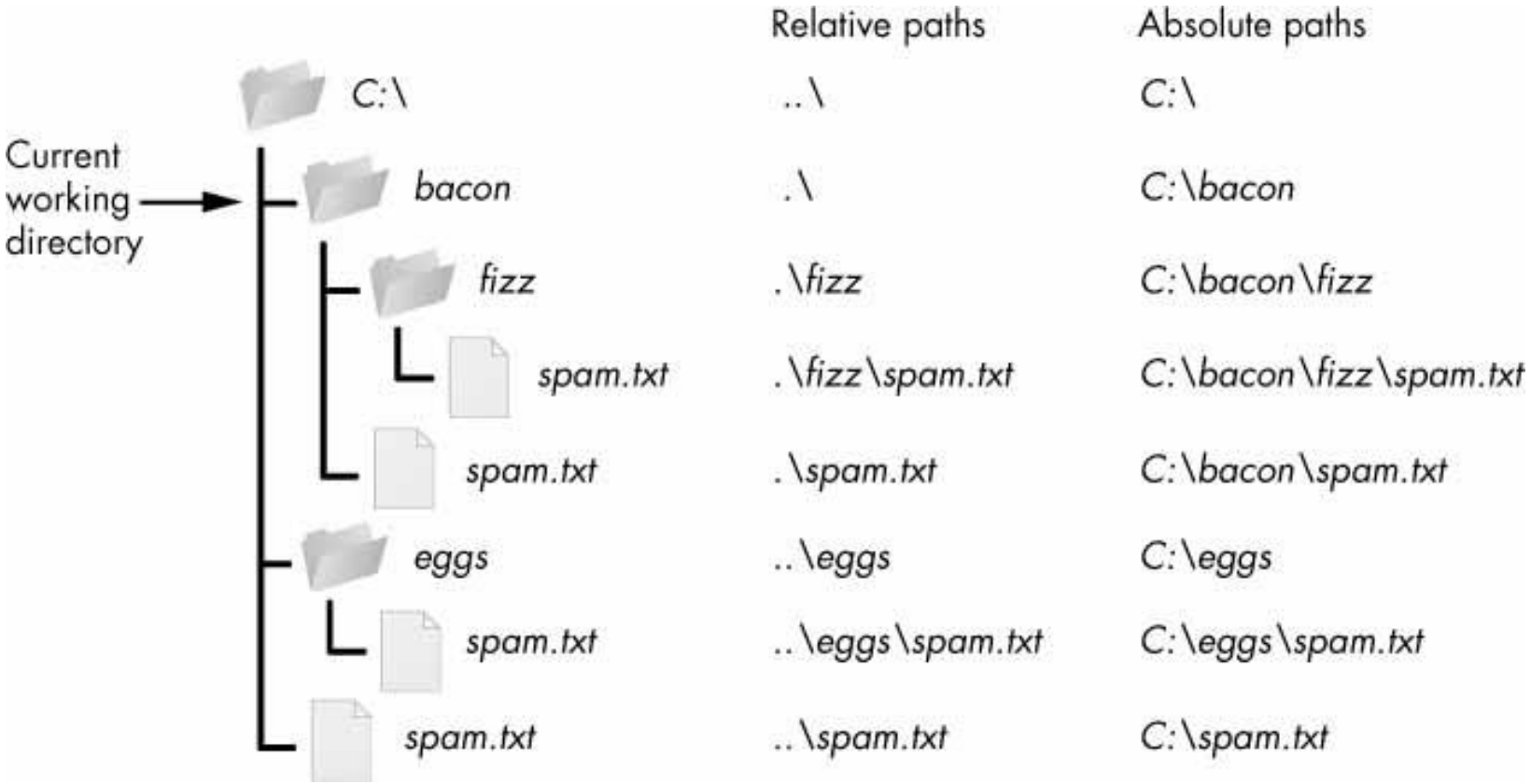
File & Directory

- การอ้าง path มี 2 แบบ
  - Absolute path คือ การอ้าง path ตั้งแต่ชั้นบนสุด เช่น

    C:\Windows\Containers\serviced
  - Relative Path คือ การอ้าง path นับจาก directory ปัจจุบัน เช่น

    Python 65\assignment
  - การอ้าง path แบบ Relative จะมีสัญลักษณ์พิเศษ ได้แก่ . หมายถึง directory ปัจจุบัน เช่น ถ้าเขียน .\assignment หมายถึง directory assignment ที่อยู่ใต้ directory ปัจจุบัน อีกสัญลักษณ์หนึ่ง คือ .. หมายถึง directory สูงขึ้นไป 1 ลำดับ

File & Directory



File & Directory

- ยังมีการทำงานอื่นๆ ที่เกียวกับ directory ได้แก่
  - การสร้าง directory จากคำสั่งข้างล่าง จะเป็นการสร้าง directory ชื่อ work ไว้ใน drive D และสร้าง directory python เอาไว้ด้านใน และสร้าง directory file_and_dir เอาไว้ข้างในอีกที

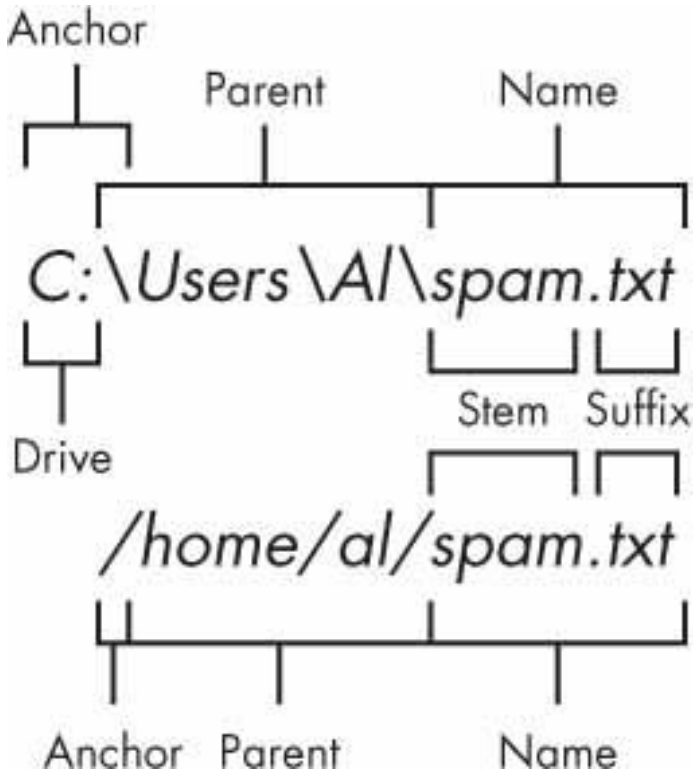
```
import os
os.makedirs('D:\\work\\python\\file_and_dir')
```

  - หรือใช้อีกวิธี

```
from pathlib import Path
Path(r'd:\work\python\file_and_dir2').mkdir()
```

File & Directory

- ใน path ของ directory จะแบ่งออกเป็นส่วนๆ ตามรูป



File & Directory

- anchor เป็น root directory จากในรูป คือ c:\
- parent คือส่วนที่เป็น directory ทั้งหมด ซึ่งอาจจะซ้อนกันหลายชั้นก็ได้
- name เป็นชื่อไฟล์ ซึ่งประกอบด้วย 2 ส่วนย่อย คือ stem คือ ชื่อไฟล์ และ suffix ซึ่งเป็นนามสกุล

```
from pathlib import Path

p = Path('d:/work/python/prog1.txt')
print(p.anchor)
print(p.parent)
print(p.name)
print(p.suffix)
```

```
d:\
d:\work\python
prog1.txt
.txt
```

File & Directory

- ในส่วนของ parent จะมีข้อมูลอีกส่วนซึ่งอยู่ในคำสั่ง Path.cwd().parents โดยจะเป็น list ของลำดับของ directory

```python
from pathlib import Path

l = len(Path.cwd().parents)
for i in range(l):
    print(Path.cwd().parents[i])
```

```
C:\Users\khtha\Desktop\Python 65\python_class_2022
C:\Users\khtha\Desktop\Python 65
C:\Users\khtha\Desktop
C:\Users\khtha
C:\Users
C:\
```

File & Directory

- สามารถหาขนาดไฟล์ และ รายชื่อไฟล์ที่อยู่ใน directory
- os.path.getsize(path) ใช้ในการหาขนาดไฟล์
- os.listdir(path) จะส่งกลับ list ของไฟล์ที่อยู่ใน directory นั้น
- กรณีที่ไม่ได้ต้องการไฟล์ทั้งหมดที่อยู่ใน directory นั้น แต่ให้มี filter สามารถทำได้โดยใช้ glob()
  - p = Path(‘d:/work/python’)
  - list(p.glob('*.txt') จะส่ง list ของชื่อไฟล์ที่ลงท้ายด้วย .txt
  - list(p.glob('project?.docx')

File & Directory

- ต่อไปจะเป็นการอ่านและเขียนไฟล์
- เริ่มต้นต้อง open ไฟล์ก่อน โดยใช้ฟังก์ชัน open ตามตัวอย่าง

```
from pathlib import Path

p = Path('d:/work/python/prog1.txt')
prog_file = open(p)
```

- สำหรับ พารามิเตอร์ ของ open จะเป็น string ธรรมดาก็ได้

```
prog_file = open('d:/work/python/prog1.txt')
```

File & Directory

- สำหรับการอ่านไฟล์จะใช้ฟังก์ชัน read

```
prog_file = open('d:/work/python/prog1.txt')

content = prog_file.read()
print(content)
```

- ฟังก์ชัน read จะอ่านมาทีเดียวทั้งไฟล์ แต่หากต้องการอ่านทีละบรรทัดต้องใช้ฟังก์ชัน readline() (ตรวจสอบหมดไฟล์ด้วย len)

```
content = prog_file.readline()
```

- จะมีคำสั่งคล้ายกัน คือ readlines() โดยจะส่งคืนเป็น list ของแต่ละบรรทัด

File & Directory

- สำหรับการเขียนไฟล์ จะยุ่งยากกว่าเล็กน้อย เนื่องจากต้อง open แบบมี mode

```
txt_file = open('test.txt', 'w')
txt_file.write('Hello, world!\n')
txt_file.close()

txt_file = open('test.txt', 'a')
txt_file.write('CE Computer Engineer.')
txt_file.close()

txt_file = open('test.txt')
content = txt_file.read()
txt_file.close()
print(content)
```

โหมด w คือ write เขียนไฟล์ใหม่ ผลคือข้อมูลเดิมจะล้างออกหมด

โหมด a คือ append จะเขียนไฟล์ต่อจากของเดิม

File & Directory

- **Ex 7.3** กำหนดให้ไฟล์ grade.txt ประกอบด้วยข้อมูลรูปแบบดังนี้

  student_id, name, subject, test_x, score, test_y, score, ...

  ...

- คือ แต่ละบรรทัดจะประกอบด้วย (แต่ละส่วนคั่นด้วย ,)
  - รหัสนักศึกษา (นักศึกษาแต่ละคน รหัสไม่ซ้ำ)
  - ชื่อนักศึกษา (ชื่อนักศึกษา ซ้ำได้) (ไม่เกิน 20 ตัวอักษร)
  - วิชา
  - ชื่อการทดสอบ (ซ้ำไปเรื่อยๆ)
  - คะแนนการทดสอบ (ซ้ำไปเรื่อยๆ) (อยู่ระหว่าง 0-100)
- เช่น

  65010123, john, python, assignment, 10, midterm, 20, final, 50

  65010123, john, calculus, homework, 20, midterm, 20, final, 50

File & Directory

- **Ex 7.3** จงเขียนโปรแกรมรับข้อมูลคะแนนของนักศึกษา จากไฟล์ โดยนักศึกษาแต่ละคน จะเรียนได้หลายวิชา โดยใช้ dictionary
  - ให้ทำเป็น function ชื่อ build_score โดยรับข้อมูลเป็นชื่อไฟล์ และส่งคืนค่ากลับเป็น dictionary ที่มีโครงสร้างดังนี้

{‘student_id’ : {‘name’: [‘subject’, score1, score2, ..., sum], [...], ...}

เช่น

{‘65015123’ :

{‘john’ : [[‘python’, 10, 20, 50, 80], [‘calculus’, 20, 20, 50, 90]] },

{‘65015124’ :

{ ‘jame’ : [[‘python’, 10, 20, 50, 80], [‘calculus’, 20, 20, 50, 90]] }

}

Dictionary

- **Ex 7.3** จงเขียนโปรแกรมรับข้อมูลคะแนนของนักศึกษา จากไฟล์ โดยนักศึกษาแต่ละคน จะเรียนได้หลายวิชา โดยใช้ dictionary
  - กำหนดให้ชื่อการทดสอบ ในวิชาเดียวกันจะไม่ซ้ำกัน
  - จำนวนคะแนนในแต่ละวิชาของนักศึกษาแต่ละคนจะเท่ากัน
  - จำนวนการทดสอบในแต่ละวิชาของนักศึกษาแต่ละคนจะเท่ากัน
  - ชื่อการทดสอบในไฟล์ของนักศึกษาแต่ละคนอาจจะเรียงไม่เหมือนกัน
  - ในการแสดงผลลำดับคะแนน จะใช้ลำดับของชื่อการทดสอบของนักศึกษาคนแรกในไฟล์เป็นเกณฑ์

Dictionary

- **Ex 7.4** จงสร้างฟังก์ชัน print_score โดยใช้ dictionary ที่ได้จากข้อ 7.3 โดยมีหน้าตาแบบนี้ (กำหนดให้คะแนนมี 4 ส่วน ตามรูป)

ID	Name	Subject	Test1	Test2	Test3	Test4	Sum.
65015001	john	python	8	18	25	35	86
65015001	john	english	8	18	25	35	86
65015002	tom	python	10	15	27	30	82
65015002	tom	english	10	15	27	30	82
65015003	lisa	python	10	20	30	40	100
65015003	lisa	english	10	20	30	40	100

For your attention