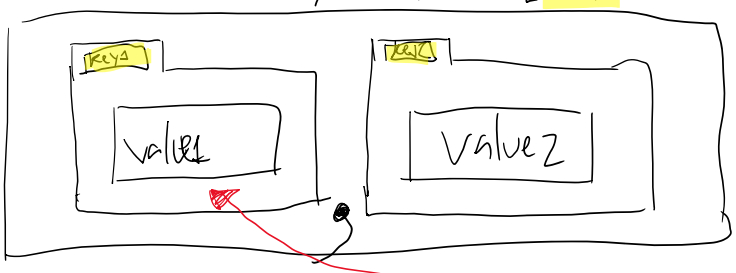
Dictionary

- เป็นที่เก็บของที่ประกอบไปด้วย คู่ ที่เป็น key กับ value
  (คู่ = pair)

- Syntax:
  my_dict = { key1: value1 , key2: value2 , key3: value3 }
  (pair, pair, pair) = 3 pairs

- Access: ดูว่าค่า(value)ของ key : my_dict[key1] → value1

my_dict =



ต้องการของใน key1 → my_dict[key1]

- Key ต้องเป็นของที่เปลี่ยนไม่ได้ (immutable):
  ✳ string, tuple, int, float

- Value เป็นอะไรก็ได้: string, int, list, tuple, dict.

EX1

```
my_dict = {"AAA": 10, "BBB": 20, "CCC": 30}
print(my_dict["AAA"])
```
→ 10

```
total = my_dict["AAA"] + my_dict["CCC"]
```
10 + 30 = 40

```
print(total)
```
→ 90

EX2

```
student = {"name": "Alice", "id": 123, "age": 23,
           "gender": "female"}
print(f"{student['Name']} age = {student['age']}")
```
Alice, 23

→ Alice age = 23

✳ Access: dict[key] → value

⊛ Access: dict[key] → value

(value ↑ ของ key นั้น)

Update: dict[key] = new_value

เปลี่ยน value ของ key นั้น เป็น new_value

EX

```
grades = {"Alice": "A", "Bob": "B", "Chris": "C"}
print(grades["Alice"])  → A
grades["Alice"] = "D"    #update
print(grades["Alice"])  → D
print(grades)  → {"Alice": "D", "Bob": "B", "Chris": "C"}
grades["Bob"] = ['A', 'B', 'C']
print(grade["Bob"])  → ['A', 'B', 'C']
```

⊛ ไม่สามารถเข้าไปเอาด้วย key ที่ไม่อยู่ได้ → Error!

```
my_dict = {"Alice": "A", "Bob": "B"}
          (key : value , key : value)
print(my_dict["Max"])  ← Error!!
```

(Max ↑ key ไม่อยู่ใน my_dict)

- in operator
  - ↳ เช็คว่า key อยู่ใน dict ไหม? → True / False

```
"Alice" in my_dict  → True
"Max" in my_dict  → False
```

- len() function
  - ↳ บอกจำนวนคู่ของ dict

```
len(my_dict)  → 2
```

การสร้าง dictionary จากว่างเปล่า ({})

[illegible]

การสร้าง dictionary จากข้างในว่าง ({ })

```
my_dict = {}
my_dict["Alice"] = 20
print(my_dict)
my_dict["Bob"] = 30
my_dict["Chris"] = 25
print(my_dict)
my_dict["Alice"] = 27
print(my_dict["Alice"])
```

(*) ใช้ = เพื่อสร้างคู่ใหม่ได้ (*)

← สร้างคู่ "Alice":20 ใน dict

→ {"Alice":20}

← สร้างคู่ใหม่

← สร้างคู่ใหม่

→ {"Alice":20, "Bob":30, "Chris":25}

* เรียงตามลำดับที่สร้าง !!

← Update Alice!

→ 27

EX สร้าง dict ใหม่จาก list ของ key และ value

```
names = ["Alice", "Bob", "Chris"]
scores = [27, 32, 25]
students = {}
for i in range(len(names)):
    students[names[i]] = scores[i]
print(students)
```

→ {'Alice': 27, 'Bob': 32, 'Chris': 25}

เป็นแบบ {"Alice":27, "Bob":32, "Chris":25}

Note: ควรจับตาม index

```
student[names[0]] = scores[0]
student["Alice"] = 27
student[names[1]] = scores[1]
⋮
student[names[n]] = scores[n]
```

Loop dictionary

↳ การ loop (for) ตาม key

EX

```
your_dict = {"Max":20, "X":30, "Y":40}
for key in your_dict:
    print(key)
```

① ② ③

→ Max
X
Y

```
for key in your_dict:
```

```
for key in your_dict:
    print(key, your_dict[key])
```

y
Max 20
x 30
y 40

Useful dictionary Methods

- .keys() → ส่ง key ทั้งหมดออกมา เป็น "list" (ใช้ร่วมกับ loop ได้)
- .values() → ส่ง value ทั้งหมดออกมา เป็น "list"
- .items() → ส่ง ทุกๆ key-value ทั้งหมดออกมา เป็น "list" ของ tuple (pair)

Ex

```
my_dict = {"Alice":20, "Bob":30}
print(my_dict.keys())   → ["Alice", "Bob"]
print(my_dict.values()) → [20, 30]
print(my_dict.items())  → [("Alice",20), ("Bob",30)]

for name, score in my_dict.items():
    print(name, score)  → Alice 20
                          Bob 30
```

.get(key, default) method

↳ ใช้ดึง value ของ key นั้น (เหมือน dict[key])
→ แต่ถ้า key ไม่อยู่ใน dict จะส่งค่า default ออกมาแทน

Ex

```
my_dict = {"Alice":1, "Bob":23}
x = my_dict.get("Alice", 0) → 1
y = my_dict.get("alice", 0) → 0
print(x, y)                 → 1 0
```

Ex เขียน function get ของเราเอง

```
def myget(dic, key, default):
```

Ex เขียน function get ของเราเอง

```
def our_get(dic, key, default):
    if key in dic:
        return dic[key]        → (1)
    else:
        return default

print(our_get(my_dict, "Alice", 0))   → 1
```

(dic = {"Alice": 1, "Bob": 23})

---

* นับของด้วย dictionary

โจทย์: นับผลไม้ใน list ว่าแต่ละประเภทมีเท่าไหร่

↳ fruits = ["apple", "apple", "banana", "orange", "orange"]

↳ output → apple 2
banana 1
orange 2

```
c_apple = 0
c_banana = 0
c_orange = 0
for fruit in fruits:
    if fruit == "apple":
        c_apple += 1
    if fruit == "banana":
        c_banana += 1
    if fruit == "orange":
        c_orange += 1
print(...)
```

* painpoint = ถ้าใช้ตัวแปร แยกแต่ละชนิดจะต้องเช็ค if ทุกอัน

idea: . ใช้ dict โดยที่แต่ละคู่จะเป็นเหมือน 1 ตัวแปร เอาไว้นับผลไม้แต่ละชนิด

↳ คู่ = key จะเป็นชื่อผลไม้
value จะเป็นจำนวนผลไม้

(1) ถ้าเรารู้ว่ามีผลไม้อะไรบ้าง

```
c_fruit = {"apple": 0,   (1, 2)
           "banana": 0
           "orange": 0}
for fruit in fruits:
    c_fruit[fruit] += 1
    = c_fruit[fruit] + 1
```

↑ key ไม่มี error!

```
# loop 1
fruit = "apple"
c_fruit["apple"] += 1
   = c_fruit["apple"] + 1
   0 + 1
   = 1

# loop 2
fruit = "apple"
c_fruit["apple"] += 1
   = 1+1 = 2
```

(2) ถ้าไม่รู้ว่ามีผลไม้อะไรบ้าง

② ถ้าไม่รู้ว่าจะมีผลไม้อะไรบ้าง

```
c_fruit = {}
for fruit in fruits:
    if fruit in c_fruit:
        c_fruit[fruit] = 0+1
    else:
        c_fruit[fruit] += 1
print(c_fruit)
```

(ลองครั้งแรก)

```
c_fruit[fruit] = c_fruit.get(fruit, 0)+1
```

EX นับคำในประโยคภาษาอังกฤษ

เช่น "The the the Cat cat meow"

the สามคำ
Cat สองคำ
meow หนึ่งคำ | ไม่สนตัวเล็กตัวใหญ่

```
sent = input("Enter sentence: ")
sent = sent.lower()
sentences = sent.split(" ") → ["the", "the", "the", "cat", "cat", "meow"]
counts = {}
for word in sentences:
    counts[word] = counts.get(word, 0) +1
for word in counts:
    print(word, counts[word])
```

Nested dict

↳ dict of dict (JSON)

```
users = {"Alice": {"age": 25, "gender": "f"},
         "Bob": {"age": 30, "gender": "m"},
         "Chris": {"age": 27, "gender": "m"}}
```

"Chris" : {"age": 27, "gender": "m"}}

print(users["Alice"]) → {"age": 25, "gender": "f"}

print(users["Alice"]["age"]) → 25

print(users["Bob"]["gender"]) → m

↳ dict of list

my_dict = {"Alice": [1, 2, 3],
"Bob": [2, 3, 4]}

print(my_dict["Alice"][1]) → 2

[1, 2, 3]
0 1 2

my_dict["Alice"].append(20)

[1, 2, 3].append(20)

print(my_dict) → {"Alice": [1, 2, 3, 20],
"Bob": [2, 3, 4]}