COMP1021
Introduction to Computer Science

Dictionaries

David Rossiter and Gibson Lam

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Explain the difference between a dictionary and a list
  2. Create a dictionary
  3. Retrieve, add, delete, and change the content of a dictionary
  4. Go through all the keys and values of a dictionary

A Quick Reminder - Lists

- You have used Python lists many times e.g.

`mylist = [` *first thing* `,` *second thing* `,` … `]`

- You can put almost anything in a list e.g.

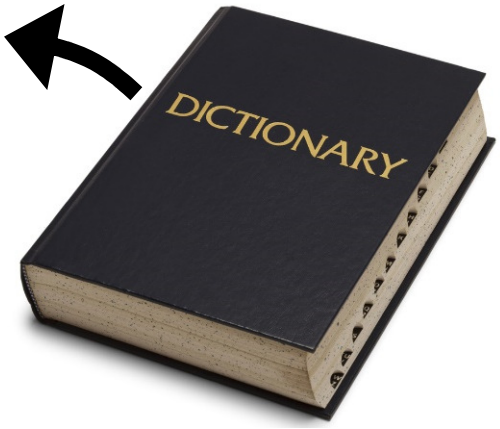
```
mylist = [2000, "apples", [True, 5.5]]
```

- Other programming languages have something similar to a list called an *array*

Dictionaries

- The basic idea is the same as a dictionary you probably had when you were in primary school:
- The idea is that one thing is *mapped to* another thing
- For example, the word 'cat' could be mapped to 'an animal with 4 legs that goes miaow'

cashier noun a person responsible for paying out and receiving money in a shop, bank, etc. • verb (**cashiers**, **cashiering**, **cashiered**) dismiss someone from the armed forces.
cashmere noun fine, soft wool from a breed of Himalayan goat.
casing noun a cover that protects or encloses something.
casino noun (plural **casinos**) a public building or room for gambling.
cask noun a large barrel for storing alcoholic drinks.
casket noun **1** a small ornamental box or chest for holding valuable objects. **2** chiefly N. Amer. a coffin.
cassava /kuh-sah-vuh/ noun the root of a tropical American tree, used as food.



Using a Dictionary

The name of the dictionary

- We create a Python dictionary like this:

```
animals = {
 "cat":"an animal with 4 legs that goes miaow"
}
```

A dictionary uses braces { }

- Then you can use the dictionary to find out what a cat is, like this:

```
result=animals["cat"]
print(result)
```

The left side

- The answer (the right side) will be printed:

```
an animal with 4 legs that goes miaow
```

Using a Dictionary

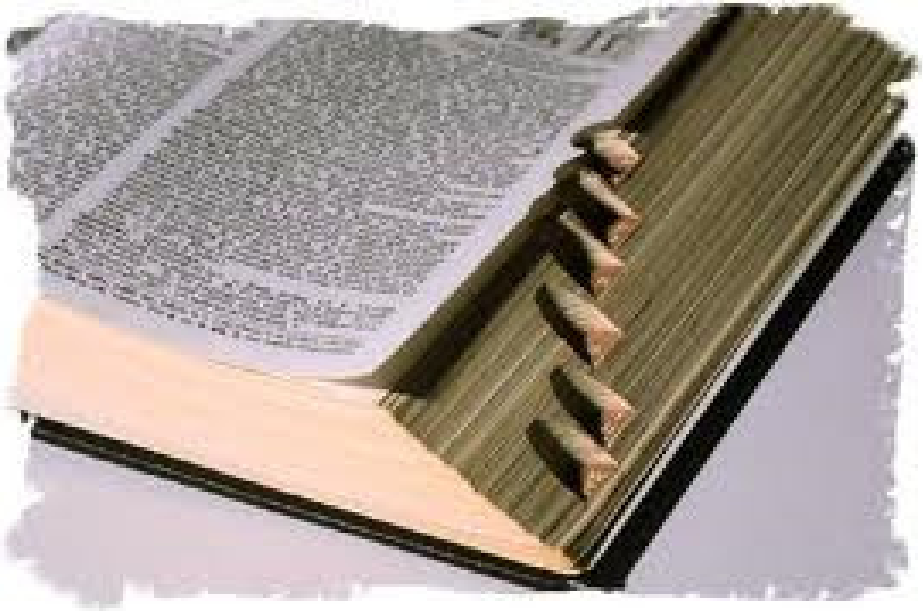
- The dictionary may have lots of entries e.g.

```
animals = {
 "ant":"a small insect with 6 legs",
 "dog":"an animal with 4 legs that goes woof",
 "cat":"an animal with 4 legs that goes miaow"
}
result=animals["ant"]
print(result)
```

The dictionary doesn't have to be in alphabetical order!

- The result:

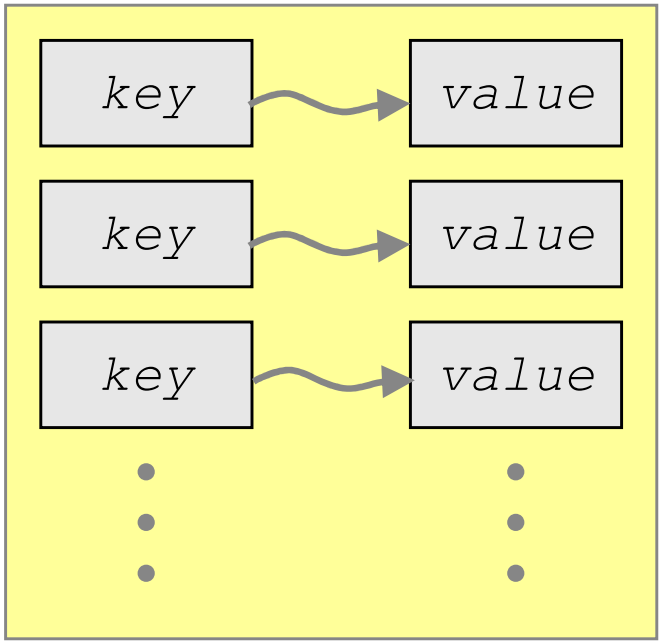
```
a small insect with 6 legs
```



A Dictionary

- The left side is called the *key*
- The right side is called the *value*
- We say that the key *is mapped to* the value

A dictionary



A Python Dictionary

- The general idea is that one single thing (the key, on the left) is mapped to another single thing (the value, on the right)
- At the moment you may think that a Python dictionary uses text, like a paper dictionary
- However, you can use almost anything for the left side/ right side of a dictionary
- You don’t need to use any text in the left side or the right side, if you don’t want to

An Example with No Text

HKUST student ID

```
credits = {
 20865052 : 20,
 29500042 : 22,
 24742797 : 18
}
```

- This dictionary stores how many credits students are currently taking

The data in a dictionary doesn't have to be in any specific order

- Examples of using the dictionary:

```
result1=credits[20865052]
print(result1)
result2=credits[24742797]
print(result2)
```

```
20
18
```

- This means the student is taking 20 credits
- This means the student is taking 18 credits

A More Advanced Dictionary

- This dictionary stores how many lecture sections courses have:

```
courseinfo={
  1021:15,
  2011:6
}
```

- Here the left side and the right side are integers

- *This means there are 15 lecture sections in COMP1021 and 6 lecture sections in COMP2011*

- Here is an extended version:

```
courseinfo = {
 (1021, "f2022"):(15, 1563),
 (2011, "f2022"):(6, 506)
}
```

- *Here extra data has been added: the semester and the number of students taking the course*

- Here the left side and the right side are tuples

- *Quick reminder – a tuple is basically the same as a list, but you can't change anything in it*

Creating a Dictionary

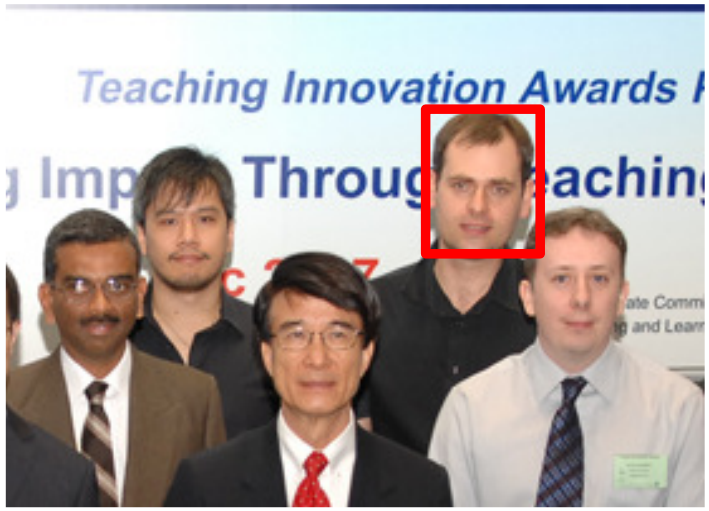
- Let us use a dictionary to store the position and size of three heads in this image:


```
heads = {"David":  (589, 106, 48, 63),
         "Gibson": (474, 102, 44, 58),
         "Paul":  (522, 162, 55, 68)
}
```

x position of top left corner — *y position of top left corner* — *width* — *height*

- In this example the left side (the key) is a string, and the right side (the value) is a tuple

Retrieving Items



- As shown before, use the key to get the right side:

```
david_data = heads["David"]
```

- After running the above code `david_data` contains a tuple of 4 numbers `(589, 106, 48, 63)`
- You could then use further code if you want e.g.

```
width = david_data[2]
print(width)
```

48

Adding New Items

- You can add a new item to a dictionary any time you want
- To add a new item to the dictionary, simply assign something to a new key:

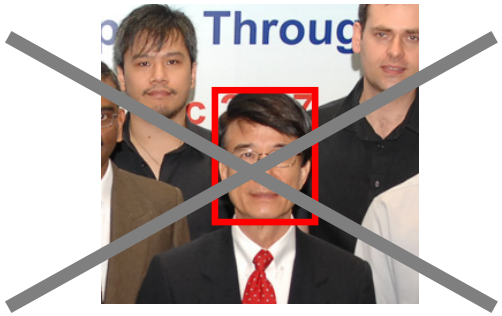

```
heads["Sean"] = (628, 146, 46, 58)
```

```
>>> print(heads)
{'David': (589, 106, 48, 63), 'Gibson':
(474, 102, 44, 58), 'Paul': (522, 162,
55, 68)}
>>> heads["Sean"] = (628, 146, 46, 58)
>>> print(heads)
{'David': (589, 106, 48, 63), 'Gibson':
(474, 102, 44, 58), 'Paul': (522, 162,
55, 68), 'Sean': (628, 146, 46, 58)}
```

An example

Deleting Items

- You can delete an item in a dictionary any time you want
- To delete an item, use `del` e.g. `del heads["Paul"]`



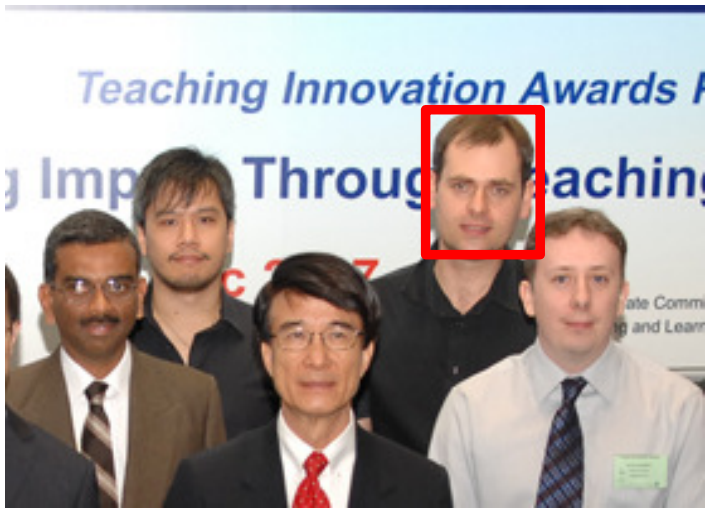
Paul Chu used to run HKUST but he left, so let's dump him

```
>>> print(heads)
{'David': (589, 106, 48, 63), 'Gibson':
(474, 102, 44, 58), 'Paul': (522, 162,
55, 68)}
>>> del heads["Paul"]
>>> print(heads)
{'David': (589, 106, 48, 63), 'Gibson':
(474, 102, 44, 58)}
```

'Paul' data is gone!

An example

Changing Items



- To change something in the dictionary, you just re-create it with the revised value e.g.

```
heads["David"] = (588, 104, 48, 57)
```

Changed

```
>>> print(heads)
{'David': (589, 106, 48, 63), 'Gibson':
(474, 102, 44, 58), 'Paul': (522, 162,
55, 68)}
>>> heads["David"]=(588, 104, 48, 57)
>>> print(heads)
{'David': (588, 104, 48, 57), 'Gibson':
(474, 102, 44, 58), 'Paul': (522, 162,
55, 68)}
```

Changed

An example

Going Through the Keys

- You may want to go through all of the left side (the keys)
- To do that you can use `.keys()` like this:

```
for key in heads.keys():
    print(key)
```

Example of Going Through the Keys

```
>>> print(heads)
{'David': (588, 104, 48, 57), 'Gibson':
(474, 102, 44, 58), 'Sean': (628, 146,
46, 58)}
>>> for key in heads.keys():
...     print(key)
...
...
David
Gibson
Sean
```

Going Through the Values

- If you want to go through all of the right side (the values) you can use `.values()` like this:

```
for value in heads.values():
    print(value)
```

Example of Going Through the Values

```
>>> print(heads)
{'David': (588, 104, 48, 57), 'Gibson':
(474, 102, 44, 58), 'Sean': (628, 146,
46, 58)}
>>> for value in heads.values():
...     print(value)
...
...
(588, 104, 48, 57)
(474, 102, 44, 58)
(628, 146, 46, 58)
```

Going Through Everything

- If you want to, you can use `.items()` to access all of the dictionary content
- For example, to print all the left side (the keys) and the right side (the values):

```
for key, value in heads.items():
    print(key, value)
```

Example of Going Through Everything

```
>>> print(heads)
{'David': (588, 104, 48, 57), 'Gibson':
(474, 102, 44, 58), 'Sean': (628, 146,
46, 58)}
>>> for key, value in heads.items():
...     print(key, value)
...
...
David (588, 104, 48, 57)
Gibson (474, 102, 44, 58)
Sean (628, 146, 46, 58)
```

You Cannot Use a List as a Key

- Although almost anything can be used as a key in a dictionary, you cannot use a list as a key, e.g.

```
heads = { [474, 102, 44, 58]: "Gibson" }
```

```
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    heads = { [474, 102, 44, 58]: "Gibson" }
TypeError: unhashable type: 'list'
```

- That's because a list can change; if it was changed that would be very confusing, use a tuple instead:

```
heads = { (474, 102, 44, 55): "Gibson" }
print(heads)
```

```
{(474, 102, 44, 55): 'Gibson'}
```