

CSIT110

# Fundamental Programming with Python

Dictionary

Goh X. Y.



# In this lecture

- Dictionary
- Dictionary methods
- None Type
- Problem solving with Dictionary

# Dictionary – How does it look like

Unordered collection of data values

used to store key-value pairs

keys must be of an immutable data type such as strings, numbers, or tuples.

keys must also be **unique**

```
# this is an empty dictionary
empty = {}
# Example of a dictionary with keys that are string only
variable_name = {
    "key_name1": "each key and value is separated by a colon",
    "key_name2": "value can be a string or a number",
    "key_name3": 20,
    "key_name4": "each key value pair is separated by a comma"
}
```

# Dictionary

used to store key-value pairs

```
person = {  
    "first_name": "Amanda",  
    "last_name": "Smith",  
    "age": 20  
} # information about a person  
  
state_abb = {  
    "NSW": "New South Wales",  
    "ACT": "Australian Capital Territory",  
    "NT": "Northern Territory",  
    "QLD": "Queensland",  
    "SA": "South Australia",  
    "TAS": "Tasmania",  
    "VIC": "Victoria",  
    "WA": "Western Australia"  
} # Australian state abbreviations
```

# Dictionary

## Main-purposes

```
person = {  
    "first_name": "Amanda",  
    "last_name": "Smith",  
    "age": 20  
} # information about a person
```

Grouping data together

```
state_abb = {  
    "NSW": "New South Wales",  
    "ACT": "Australian Capital Territory",  
    "NT": "Northern Territory",  
    "QLD": "Queensland",  
    "SA": "South Australia",  
    "TAS": "Tasmania",  
    "VIC": "Victoria",  
    "WA": "Western Australia"  
} # Australian state abbreviations
```

Mapping

# Dictionary

- Mapping

```
digit_to_word = {  
    0: "zero",  
    1: "one",  
    2: "two",  
    3: "three",  
    4: "four",  
    5: "five",  
    6: "six",  
    7: "seven",  
    8: "eight",  
    9: "nine"  
}
```

```
word_to_digit = {  
    "zero": 0,  
    "one": 1,  
    "two": 2,  
    "three": 3,  
    "four": 4,  
    "five": 5,  
    "six": 6,  
    "seven": 7,  
    "eight": 8,  
    "nine": 9  
}
```

# Dictionary - print

```
person = {  
    "first_name": "Amanda",  
    "last_name": "Smith",  
    "age": 20  
}
```

using function `print` to print out the whole dictionary

```
print(person)
```

# Dictionary – get value

```
person = {  
    "first_name": "Amanda",  
    "last_name": "Smith",  
    "age": 20  
}
```

Values can be retrieved using function **get** with the corresponding keys:

```
name = person.get("first_name")    → "Amanda"  
  
family_name = person.get("last_name")    → "Smith"  
  
age_ = person.get("age")           → 20  
  
address_ = person.get("address")      → None
```



# None

```
person = {  
    "first_name": "Amanda",  
    "last_name": "Smith",  
    "age": 20  
}
```

**None** is equivalent to **null** in other programming languages  
It also means it has no value:

```
email = person.get("email")  
if (email is None):  
    print("User has no email")  
else:  
    print("User email is " + email)
```

→ None

# Dictionary – get with default value

```
person = {  
    "first_name": "Amanda",  
    "last_name": "Smith",  
    "age": 20  
}
```

We can specify a **default value** in the function `get` if the key-value pair is not found:

```
std_type = person.get("student_type", "N/A")    # →  
"N/A"  
  
credit_point = person.get("credit_point", 0)    # → 0
```

# Dictionary – Example

```
digit_to_word = {  
    0: "zero",  
    1: "one",  
    2: "two",  
    3: "three",  
    4: "four",  
    5: "five",  
    6: "six",  
    7: "seven",  
    8: "eight",  
    9: "nine"  
}  
  
print(digit_to_word.get(7))
```

seven

# Dictionary – Example

```
word_to_digit = {  
    "zero": 0,  
    "one": 1,  
    "two": 2,  
    "three": 3,  
    "four": 4,  
    "five": 5,  
    "six": 6,  
    "seven": 7,  
    "eight": 8,  
    "nine": 9  
}  
  
print(word_to_digit.get("eight"))
```

8

# Dictionary – get value

Another way to access the value with the key:

```
person = {  
    "first_name": "Amanda",  
    "last_name": "Smith",  
    "age": 20  
}
```

```
first_name = person["first_name"] → "Amanda"
```

```
last_name = person["last_name"] → "Smith"
```

```
age = person["age"] → 20
```

```
address = person["address"] → TypeError!
```

# Dictionary – update values

```
person = {  
    "first_name": "Amanda",  
    "last_name": "Smith",  
    "age": 20  
}
```

we can change the existing values:

```
person["first_name"] = "Mandy"  
person["last_name"] = "Jones"  
person["age"] = 24
```

```
person = {  
    "first_name": "Many",  
    "last_name": "Jones",  
    "age": 20  
}
```

# Dictionary – add new key-value pair

```
person = {  
    "first_name": "Amanda",  
    "last_name": "Smith",  
    "age": 20  
}
```

we can add new key-value pair:

```
person["email"] = "Mandy.Jones@gmail.com"  
person["gpa_score"] = 3.5
```

```
person = {  
    "first_name": "Many",  
    "last_name": "Jones",  
    "age": 20,  
    "email": "Mandy.Jones@gmail.com",  
    "gpa_score": 3.5  
}
```

# Dictionary – delete a key-value pair

```
person = {  
    "first_name": "Mandy",  
    "last_name": "Jones",  
    "age": 24,  
    "email": "Mandy.Jones@gmail.com"  
}
```

we can delete a key-value pair:

```
del person["email"]
```

```
person = {  
    "first_name": "Many",  
    "last_name": "Jones",  
    "age": 20  
}
```

we can delete **all** key-value pairs, the dictionary becomes empty:

```
person.clear()
```

```
person = {}
```



# Dictionary – get all keys

```
person = {  
    "first_name": "Mandy",  
    "last_name": "Jones",  
    "age": 24,  
    "email": "Mandy.Jones@gmail.com"  
}
```

We can get the list of all keys:

```
all_keys = person.keys()  
  
for key in all_keys:  
    print(key)
```

```
first_name  
last_name  
age  
email
```

# Dictionary – get all values

```
person = {  
    "first_name": "Mandy",  
    "last_name": "Jones",  
    "age": 24,  
    "email": "Mandy.Jones@gmail.com"  
}
```

We can get the list of all values:

```
all_values = person.values()  
  
for value in all_values:  
    print(value)
```

```
Mandy  
Jones  
24  
Mandy.Jones@gmail.com
```

# Dictionary – Example: capitals of cities

```
capital_city = {  
    "Australia": "Canberra",  
    "Denmark": "Copenhagen",  
    "Ireland": "Dublin",  
    "New Zealand": "Wellington",  
    "Nepal": "Kathmandu"  
}  
# ask user to enter country  
country = input("Enter country: ")  
  
# retrieve the capital city  
capital = capital_city.get(country)  
  
# display capital  
print(f"Capital city of {country} is {capital}")
```

Enter country: **Australia**  
Capital city of Australia is Canberra

# Dictionary – Example: capitals of cities

```
capital_city = ...

# ask user to enter country
country = input("Enter country: ")

# retrieve the capital city
capital = capital_city.get(country)

# display capital
if capital is None:
    print("Sorry I don't know the capital city of " + country)
else:
    print(f"Capital city of {country} is {capital}")
```

```
Enter country: Atzovia
Sorry I don't know the capital city of Atzovia
```

# Dictionary – Example: State abbreviation

```
state_abb = {  
    "NSW": "New South Wales",  
    "ACT": "Australian Capital Territory",  
    "NT": "Northern Territory",  
    "QLD": "Queensland",  
    "SA": "South Australia",  
    "TAS": "Tasmania",  
    "VIC": "Victoria",  
    "WA": "Western Australia"  
}
```

Enter state NSW/ACT/NT/QLD/SA/TAS/VIC/WA: **NT**  
The state you entered is Northern Territory

```
# ask user to enter state code  
state_code = input("Enter state NSW/ACT/NT/QLD/SA/TAS/VIC/WA: ")  
  
# retrieve the state name  
state_name = state_abb.get(state_code)  
  
print("The state you entered is " + state_name)
```

# Dictionary – Example: Subject selection

Welcome to subject enrolment

Enter subject code: **MATH111**

Enter credit point: **10**

Add more subjects? Y/N: **Y**

Enter subject code: **CS222**

Enter credit point: **4**

Add more subjects? Y/N: **Y**

Enter subject code: **LOGIC333**

Enter credit point: **5**

Add more subjects? Y/N: **N**

| Subject code | CP |
|--------------|----|
| MATH111      | 10 |
| CS222        | 4  |
| LOGIC333     | 5  |

# Dictionary – Example: Subject selection

Welcome to subject enrolment

Enter subject code: **MATH111**

Enter credit point: **10**

Add more subjects? Y/N: **Y**

Enter subject code: **CS222**

Enter credit point: **4**

Add more subjects? Y/N: **Y**

Enter subject code: **LOGIC333**

Enter credit point: **5**

Add more subjects? Y/N: **N**

| Subject code | CP |
|--------------|----|
| MATH111      | 10 |
| CS222        | 4  |
| LOGIC333     | 5  |

Put subject information into a **dictionary**

```
{  
    "code": "MATH111",  
    "cp": 10  
}
```

```
{  
    "code": "CS222",  
    "cp": 4  
}
```

```
{  
    "code": "LOGIC333",  
    "cp": 5  
}
```

Put all these dictionaries  
into a **list**

# Dictionary – Example: Subject selection

```
# display greeting
print("Welcome to subject enrolment")

# create a list to store subject dictionaries
subject_list = []

while True:
    ... # ask user to enter subject info

    subject = ... # create a dictionary to hold subject info

    subject_list.append(subject) # add subject to list

    # ask user if they want to continue
    more_subject = input("Add more subjects? Y/N: ")
    if (more_subject == "N"):
        break
```



# Dictionary – Example: Subject selection

```
...
while True:
    # ask user to enter subject info
    subject_code = input("Enter subject code: ")
    user_input = input("Enter credit point: ")
    subject_cp = int(user_input)

    subject = { # create a dictionary to hold subject info
        "code": subject_code,
        "cp": subject_cp
    }

    subject_list.append(subject) # add subject to list

    # ask user if they want to continue
    more_subject = input("Add more subjects? Y/N: ")
    if (more_subject == "N"):
        break
```

# Dictionary – Example: Subject selection

```
...  
# display the selected subjects  
print(f"{'Subject code':<15}{'CP':>2}")  
  
for i in range(0, len(subject_list)): # get the ith subject from the list  
    subject = subject_list[i]         # which is a dictionary  
  
    subject_code = subject.get("code") # get subject info from the dictionary  
    subject_cp = subject.get("cp")  
  
# display subject info  
print(f"{subject_code:<15}{subject_cp:>2}")
```

| Subject code | CP |
|--------------|----|
| MATH111      | 10 |
| CS222        | 4  |
| LOGIC333     | 5  |

# Problem solving – Challenge yourself!

Please enter numerical code: **017689**

You have entered: zero-one-seven-six-eight-nine

# Extra info – Type hints

```
from typing import List, Dict, Tuple

n: List[] = [1,2,3,'1223']
o: List[int] = [3,2,1]
p: List[int, str] = [323, 'hi there']
q: Dict[str, str] = {'key': 'value'}
r: Tuple[int, str] = ('text',10001)
```

# Extra info - Sets

Not tested – good to know

# Sets -

```
thisset = {"apple", "banana", "cherry"}  
# construct a set using a list  
thisset = set(["apple", "banana", "cherry"])
```

- Items are **unique**
- unordered and unindexed
- written with curly brackets.
- Useful for removing duplicates

```
myList = ["apple", "banana", "cherry", "apple"]  
unique_list = list(set(myList))
```

# Sets -

```
thisset = {"apple", "banana", "cherry"}  
# construct a set using a list  
thisset = set(["apple", "banana", "cherry"])
```

- Items are **unique** -> Useful for removing duplicates
  - unordered and unindexed
  - written with curly brackets.
- 
- Once a set is created, you cannot change its items, but you can add or remove items.

# Sets – loop through items

```
thisset = {"apple", "banana", "cherry"}
```

```
for x in thisset:  
    print(x)
```



# Sets – add items

```
# to add one item, use .add()
```

```
thisset.add("orange")
```

```
# to add multiple item, use .update()
```

```
thisset.update(["durian", "mango", "grapes"])
```

```
thisset.update({"mangosteen", "duku", "jackfruit"})
```

# Sets – remove items

```
thisset.discard("apple")  
print(thisset)
```

```
thisset -= {"cherry"}  
print(thisset)
```

# Sets -

Can you find out what these methods do?

- `<class 'set'>.pop()`
- `<class 'set'>.clear()`
- `<class 'set'>.union()`
- `<class 'set'>.issubset()`
- `del <class 'set'>.`

Any questions?