MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT9133 Semester 1 2019
Programming Foundations in Python

Week 4:
Collective Data Types, I/O and Comments

Chunyang Chen
Gavin Kroeger

GROUP OF EIGHT AUSTRALIA

Module 2 Synopsis

- Module 2 is aimed to provide you with:
  - Concepts of data structure and data type
  - Collective data types in Python:
    - Strings
    - Sequences: List, Tuple, Set and Dictionary
    - Built-in methods
  - Control flow structures: selection and iteration

Module 2 Learning Objectives

- Upon completing this module, you should be able to:
  - Recognise key differences between data structure and data type
  - Deploy a suitable Python built-in data type for the representation of a particular form of data
  - Control the flow of program execution with selective and iterative structures

MONASH University

Collective Data Types: Set, Dictionary

Set Data Type

- Set (Python type `set`):
  - A data type for representing a collection of **unique**, unordered items

- Creating a set:
  - An empty set: `a_set = set()`
  - A set with a number of items: `a_set = {"one", 2}`
  - Build a set from a list: `a_set = set([1,2,2,3])`

Heterogeneous type

- Accessing individual items in a set:
  - Note: Set does not support *indexing* and *item assignment*.

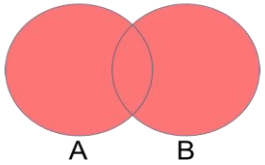
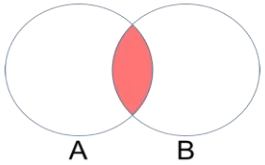
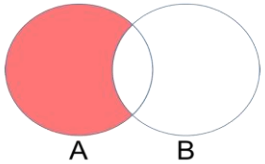
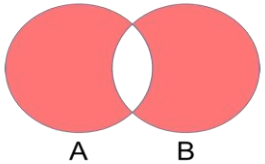
(More on) Set Data Type

- Adding new items to a set:
  - Create a new set with the new items added
  - Syntax: `a_set.add('c')`

- Removing items from a set:
  - `remove()`: accept one argument indicating the item to be removed; raise *KeyError* if the item does not exist
  - `discard()`: similar to `remove()`; does not raise *KeyError* if item does not exist
  - `pop()`: select an arbitrary item to remove and return it
  - `clear()`: remove all the items from a set in place

(More on) Set Data Type

- Set operations
  - Union:
    - `a_set.union(b_set)`
    - `a_set | b_set`
  - Intersection:
    - `a_set.intersection(b_set)`
    - `a_set & b_set`
  - Difference:
    - `a_set.difference(b_set)`
    - `a_set - b_set`
  - Symmetric difference:
    - `a_set.symmetric_difference (b_set)`
    - `a_set ^ b_set`

Set Operation	Venn Diagram	Interpretation
Union		$A \cup B$, is the set of all values that are a member of A, or B, or both.
Intersection		$A \cap B$, is the set of all values that are members of both A and B.
Difference		$A \setminus B$, is the set of all values of A that are not members of B
Symmetric Difference		$A \triangle B$, is the set of all values which are in one of the sets, but not both.

(More on) Set Data Type

- Examples on manipulating a Python set:

```
>>> first_set = set()
>>> len(first_set)
>>> 0
>>> first_set.add('a')
>>> first_set.add('b')
>>> first_set.add('b')
>>> print(first_set)
>>> {'a','b'}
>>> char_list = ['b','c','c','d','d','e']
>>> second_set = set(char_list)
>>> print(second_set)
>>> {'e','d','c','b'}
>>> intersection = first_set & second_set
>>> print(intersection)
>>> {'b'}
>>> union = first_set | second_set
>>> print(union)
>>> {'e','d','c','a','b'}
```

Sets are unordered

https://docs.python.org/3/tutorial/datastructures.html#sets

Review Question 1

What would be the output for the given program?

```
num_list = [4, 2, 2, 1, 3, 4, 1]
tmp_set = set(num_list)
num_list = list(tmp_set)
print(num_list)
```

A. [4, 2, 2, 1, 3, 4, 1]

B. {1, 2, 3, 4}

C. [1, 2, 3, 4]

D. Some error occurred

E. Not sure

Review Question 2

What would be the output for the given program?

```
a_set = {1, 2, 3, 4}
a_set[0] = 5
print(a_set)
```

A. {1, 2, 3, 4}

B. {5, 2, 3, 4}

C. {5, 1, 2, 3, 4}

D. Some error occurred

E. Not sure

Difference between set and list

- Order
  - List is ordered while set is unordered
- Duplicate
  - Sets can't contain duplicates
- In Statement
  - set is much more efficient than list

Dictionary Data Type

- Dictionary (Python type **`dict`**):
  - A *mapping* data type that associates a key with a value
  - Keys must be immutable types; values can be of any data type
  - Each data item is represented as **`key:value`**

- Creating a dictionary:
  - An empty dictionary: **`a_dict = {}`**
  - A dictionary with a number of items:
    `a_dict = {"one":1, "two":2}`
  - Build a dictionary from a list of tuples:
    `a_dict = dict([('a',1),('b',1)])`

Dictionary Data Type

- Adding new items to a dictionary:
  - Syntax: `a_dict[new_key] = new_value`
  - If `new_key` presents in the dictionary, the existing value associated to this key is updated to `new_value`
- Removing items from a dictionary:
  - Syntax: `del a_dict[a_key]`
- Checking for a key in a dictionary:
  - Syntax: `a_key in a_dict` or `a_key not in a_dict`

(More on) Dictionary Data Type

- Examples on manipulating a Python dictionary:

```
>>> simple_dict = {}
>>> len(simple_dict)
>>> 0
>>> simple_dict = {'a':0, 'b':0, 'c':0}
>>> print(simple_dict['b'])
>>> 0
>>> simple_dict['b'] = 1
>>> print(simple_dict['b'])
>>> 1
>>> simple_dict['d'] = 2
>>> print(simple_dict)
>>> {'a':0, 'b':1, 'c':0, 'd':2}
>>> item_list = list(simple_dict.items())
>>> print(item_list[1])
>>> ('b', 1)
>>> keys_list = list(simple_dict.keys())
>>> print(key_list[3])
>>> d
>>> del simple_dict['d']
>>> print(simple_dict)
>>> {'a':0, 'b':1, 'c':0}
```

(More on) Dictionary Data Type

- Accessing individual data items (**`key`**, **`value`**) in a Python dictionary:

```
simple_dict = {'a':0, 'b':1, 'c':2}
for (key, value) in simple_dict.items():
    print(key, value)
```

Review Question 3

What would be the output for the given program?

```
key_list = ['a', 'b', 'c']
value_list = [0, 1, 2]
combine_list = list(zip(key_list, value_list))
new_dict = dict(combine_list)
print(new_dict)
```

A. [('a',0), ('b',1), ('c',2)]

B. { 'a':0, 'b':1, 'c':2 }

C. Some error occurred

D. Not sure

https://docs.python.org/3/library/functions.html#zip

Termination of the loop: Continue, Break

Continue

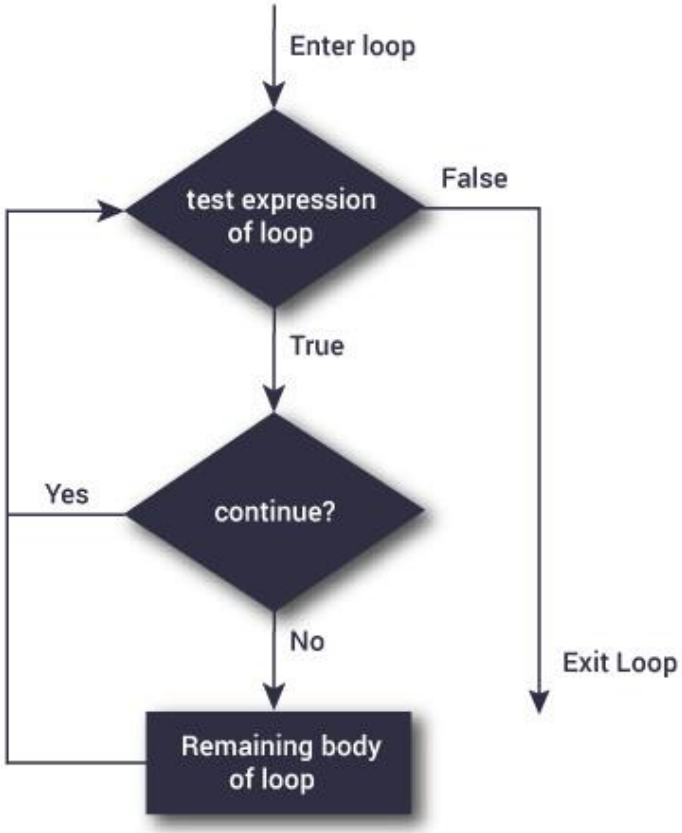
- The **continue** statement skips the current iteration of a <u>for</u> or <u>while</u> loop.

```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```



Continue

- Example about the **continue**:
  - Given a string and a target character, we want to remove the character of the string.

```
a_str = "helloWorld"
charToRemove = "e"
new_str = ""
for char in a_str:
    if char == charToRemove:
        continue
    new_str += char
print(new_str)
```

Break

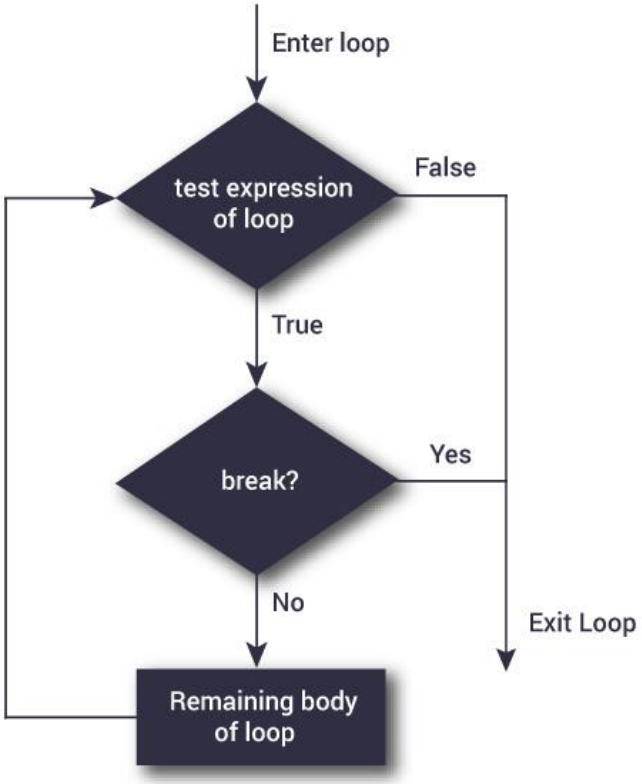
- The break statement terminates the loop containing it.
  - Create a new set with the new items added
  - Syntax: `a_set.add('c')`

```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```



Break

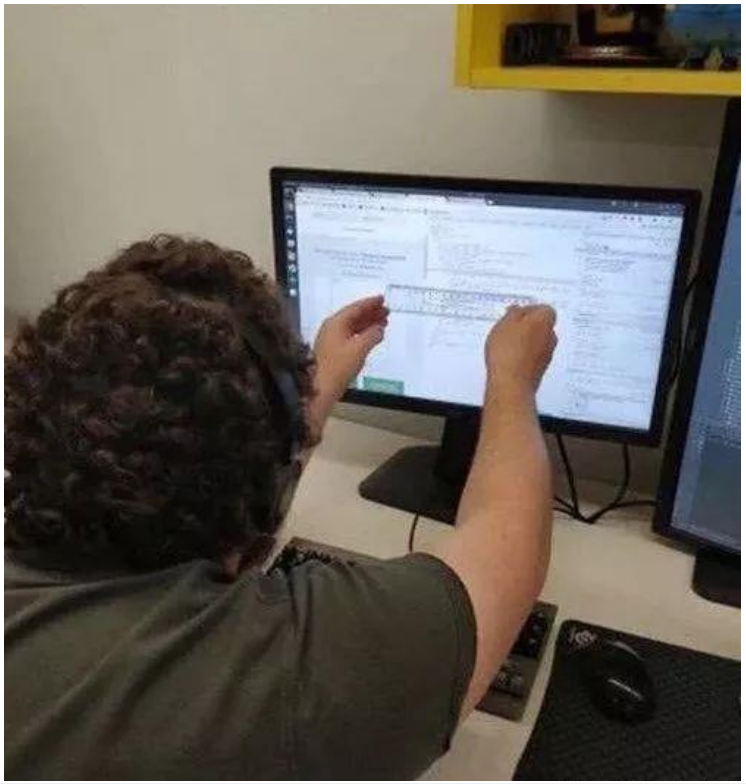
- Example about the **break**:
  - Given a list of numbers and a target number, we want to find that if the target number is in the list.
  - You can use **in** statement, but we want to use the for-loop and break in this example

```
number_list = [3, 11, 9, 7, 6, 5, 100, 20, 9, 6, 3, 1, 0]
target = 9
for number in number_list:
    if number == target:
        print("The target number is in the list")
        break
```

Pay attention to Indentation

- When there are nested loops, please pay special attention to the usage of **continue** and **break** statement.
- Indentation can help you understand which loop you want to **continue** or **break**.

```
a_list = [1, 2, 3]
b_list = [2, 5, 6]
for itemA in a_list :
    for itemB in b_list:
        if itemA == itemB:
            break
        print(itemA, itemB)
```



Comments

Comments

- Purpose of documentation:
  - To enhance the readability of your programs
  - To explain the meaning or functionality of your programs

- Comments in Python:
  - Denoted by the symbol '**#**'
  - Any parts of your code begin with # are ignored by the interpreter

```
a = 1     # first number
b = 2     # second number
result = a + b

# print the result
print("The addition of a and b is", result)
```

More on Comments

- Inline comments:
  - Comments on the same line as the statements

```
# Program description: ...
# Author: ...
# First created: ...
# Last modified: ...
```

- Block comments:
  - Multiple-line comments
  - Each line starts with '**#**'
  - Useful for program header comments
  - Or with ' """ '

```
"""
This code is used for …
"""
```

- Python's style guide on comments:
  - https://www.python.org/dev/peps/pep-0008/#comments
  - http://google.github.io/styleguide/pyguide.html#38-comments-and-docstrings

Comments

- Comments for emphasize or understanding, not verbose or amiguous translation

```python
# Create the neural network
def conv_net(x_dict, n_classes, dropout, reuse, is_training):
    # Define a scope for reusing the variables
    with tf.variable_scope('ConvNet', reuse=reuse):
        # TF Estimator input is a dict, in case of multiple inputs
        x = x_dict['images']

        # MNIST data input is a 1-D vector of 784 features (28*28 pixels)
        # Reshape to match picture format [Height x Width x Channel]
        # Tensor input become 4-D: [Batch Size, Height, Width, Channel]
        x = tf.reshape(x, shape=[-1, 28, 28, 1])
```


```python
# student number of class A
studentNumberClassA = 1
# second number of class B
studentNumberClassB = 2
# add the student number
studentNumberTotal = a + b

# print the result
print("The addition of a and b is", result)
```

```python
# drunk, fix later
# magic, do not touch
```

- Two things developers hate most:
  - Hate writing the comments/documents to their code
  - Hate others' code without comments/documents

MONASH University

Standard Input and Output in Python

Input and Output

- Input and output:
  - Two essential components of a program
- Input:
  - Data needed for solving a specific computational problem
- Output:
  - Presentation of the computational results

```
a = 1
b = 2
result = a + b
print("The addition of a and b is", result)
```

Standard Input

```
a = int(input("Enter the first number: "))
b = int(input("Enter the second number: "))
result = a + b
print("The addition of a and b is", result)
```

- To obtain data values through the standard input (i.e. the keyboard)

- The **`input`** function:
  - **`input("prompt statement:")`** or **`input()`**
  - Python built-in function to obtain data externally
  - Input values are returned as objects of type **`str`**
  - Convert input values into type **`int`** using the conversion function **`int()`** in order to perform arithmetic operations

type casting or type conversion

Standard Output

- To display any information and or computational results on the standard output (i.e. the terminal screen or console)

- The **`print`** function:
  - **`print("output string")`**
  - By default a newline character ('**`\n`**') is appended at the end
  - Each print statement will display the output on a separate line
  - Output arguments to be displayed must of of type **`str`**

```
print("This is just a print statement.")
print("This is another print statement.")
```

What would be the output on the screen?

Question 4

What would be the output on the screen?

```
print("This is just a print statement.", end = " ")
print("This is another print statement.")
```

A. Two separate lines
B. One single line
C. Not sure

Standard Output

- Two ways of displaying multiple output arguments with **`print()`**:
  - With the comma '**,**' to separate each output argument
  - With the operator '**+**' to concatenate multiple output arguments

```
print("The addition of a and b is", result)
print("The addition of a and b is " + str(result))
```

type conversion

- Note: When '**+**' is used with output arguments of Python built-in types (**`int`** or **`float`**), explicit type conversion to **`str`** is required.

Week 4 Summary

- We have discussed:
  - Data structures vs data types
  - Collective data types (String, List, Tuple, Set, Dictionary)
  - Control structures (selection, iteration)

- Next week:
  - File I/O
  - Module 3 Concepts of Decomposition

Reminder: Assignment 1 due on 8th September 11:55pm.