CS 1340 Introduction to Computing Concepts

Instructor: Xinyi Ding

Sep 16 2019, Lecture 9

Agenda

- Agenda:
  - Quick review of concepts from last lecture
  - Organize code using Modules
  - Inputs/outputs
  - File Handling

Functions

- A function
  - A block of code which only runs when it is called
  - One way to organize and reuse code
  - You can pass information to a function
  - You can ask a function to return data

Function definition begins with def

Function name and its arguments (optional)

```
def  get_final_answer(filename):
    """ Documentation String """
    line1
    line2
    return total_counter
...
```

Indentation(4 spaces)

Colon

return indicates the value to be sent back to the caller (optional)

First line with less indentation is considered to be outside of the function definition

Functions

- Passing arguments
  - A function definition can have multiple parameters, a function call may need multiple arguments
  - Ways of passing arguments
    - positional arguments
      - need to be in the same order the parameters were written
    - keyword arguments
      - where each argument consists of a variable name and a value

Functions

- All functions in Python have a return value
  - even if no return line inside the code
- Functions without a return return the special value None
  - None is a special constant in the language.
  - None is also logically equivalent to False

```python
def build_person(first_name, last_name):
    """Return a dictionary of information about a person"""
    person = {"first": first_name, "last": last_name}
    print(person)


player = build_person("kobe", "bryant")
print(player)
```

functions

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
{'first': 'kobe', 'last': 'bryant'}
None

Process finished with exit code 0
```

Lambda function

- A lambda function is a small anonymous function
- A lambda function can take any number of arguments, but can only have one expression.
- Syntax: **lambda arguments : expression**

```python
x = lambda a : a + 10
print(x(5))

x = lambda a, b : a * b
print(x(5, 6))
```

functions

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/functions.py
15
30

Process finished with exit code 0
```

Modules

- Consider a module to be the same as a code library
- A file containing a set of functions you want to include in your application
- To create a module just save the code you want in a file with the extension .py
- Use import statement to include the code you want to use

Modules

- Importing an Entire Module

pizza.py | making_pizza.py

```python
def make_pizza(size, *toppings):
    """Summarize the pizza we are about to make"""
    print("\nMaking a " + str(size) +
          "-inch pizza with the following toppings:")
    for topping in toppings:
        print("- " + topping)
```

pizza.py | making_pizza.py

```python
import pizza

pizza.make_pizza(16, "pepperoni", "extra cheese")

```

making_pizza

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/making_pizza.py

Making a 16-inch pizza with the following toppings:
- pepperoni
- extra cheese

Process finished with exit code 0
```

Modules

- Using as to Give a Module an Alias

pizza.py | making_pizza.py

```python
import pizza as pz

pz.make_pizza(16, "pepperoni", "extra cheese")
```

making_pizza

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/making_pizza.py

Making a 16-inch pizza with the following toppings:
- pepperoni
- extra cheese

Process finished with exit code 0
```

Modules

- Importing Specific Functions

pizza.py | making_pizza.py

```python
from pizza import make_pizza

make_pizza(16, "pepperoni", "extra cheese")
```

making_pizza

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/making_pizza.py

Making a 16-inch pizza with the following toppings:
- pepperoni
- extra cheese

Process finished with exit code 0
```

Modules

- Using as to Give a Function an Alias

pizza.py | making_pizza.py

```python
from pizza import make_pizza as mp

mp(16, "pepperoni", "extra cheese")

```

making_pizza

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/making_pizza.py

Making a 16-inch pizza with the following toppings:
- pepperoni
- extra cheese

Process finished with exit code 0
```

Modules

- Importing all functions in a Module

pizza.py | make_pizza.py

```python
from pizza import *


make_pizza(16, "pepperoni", "mushroom", "extra cheese")
greeting("jake")

```

make_pizza

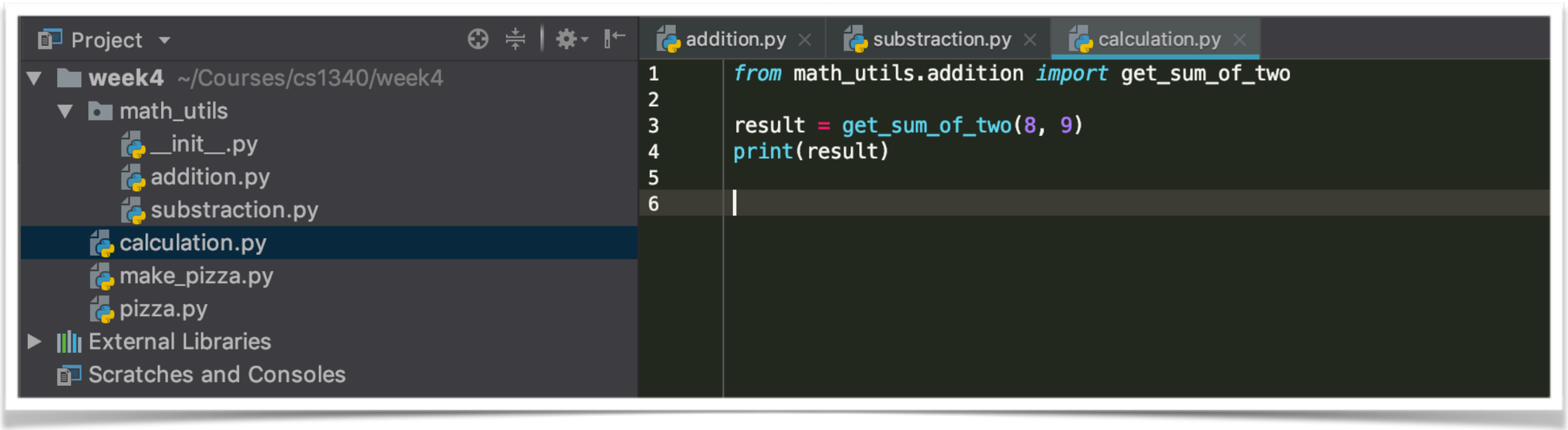
```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week4/make_pizza.py

Making a 16-inch pizza with the following toppings:
- pepperoni
- mushroom
- extra cheese
Hello jake

Process finished with exit code 0
```

Packages

- Put related modules in a package
  - A directory with a __init__.py file
  - Organize related modules



Demo

Python Input/Output

- Up till now, our program were static. The value of variables were defined or hard coded into the source code.
- To Allow flexibility we might want to take the input from the user. We use input() function in python

input([prompt])

Where prompt is the string we wish to display on the screen, it is optional

Python Input/Output

- An simple example

addition.py | substraction.py | calculation.py | inputs.py

```python
username = input("Please input your username: ")
password = input("Please give me your password: ")

print(username)
print(password)
```

inputs

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week4/inputs.py
Please input your username: xinyi
Please give me your password: 123456
xinyi
123456

Process finished with exit code 0
```

Python Input/Output

- Number guessing game
  - Each around
    - It generates a random number between 1 and 100
    - Take one input from the user
    - Give a prompt if the guess is too low or too high

Python Input/Output

- Input() takes input from the standard device(screen)
- print() outputs data to the standard output device (screen)
- The actual syntax of the print() function is

```
print(*objects, sep=' ',  end='\n', file=sys.std, flush=False)
```

```
1  print(1, 2, 3, 4,)
2  print(1, 2, 3, 4, sep="*")
3  print(1, 2, 3 ,4, sep="#", end="&")
4
5
```

number_guessing × prints ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week4/prints.py
1 2 3 4
1*2*3*4
1#2#3#4&
Process finished with exit code 0
```

Python Input/Output

- Output formating

```
x = 5
y = 10
print("The value of x is " + str(5) + " and y is " + str(y))
print("The value of x is {} and y is {}".format(x, y))
```

number_guessing × prints ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week4/prints.py
The value of x is 5 and y is 10
The value of x is 5 and y is 10

Process finished with exit code 0
```

```
print("I love {0} and {1}".format("bread", "butter"))
print("I love {1} and {0}".format("bread", "butter"))
```

number_guessing × prints ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week4/prints.py
I love bread and butter
I love butter and bread

Process finished with exit code 0
```

Python Input/Output

- Output formatting, more info https://pyformat.info/

```
pi = 3.1415926
print("The value of pi is %.2f"%pi)          # This is the old style formating
print("The value of pi is {:.2f}".format(pi)) # This is the new and preferred one

```

number_guessing × prints ×

```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week4/prints.py
The value of pi is 3.14
The value of pi is 3.14

Process finished with exit code 0
```