

CS 1340 Introduction to Computing Concepts

Instructor: Xinyi Ding
Feb 13 2020, Lecture 8

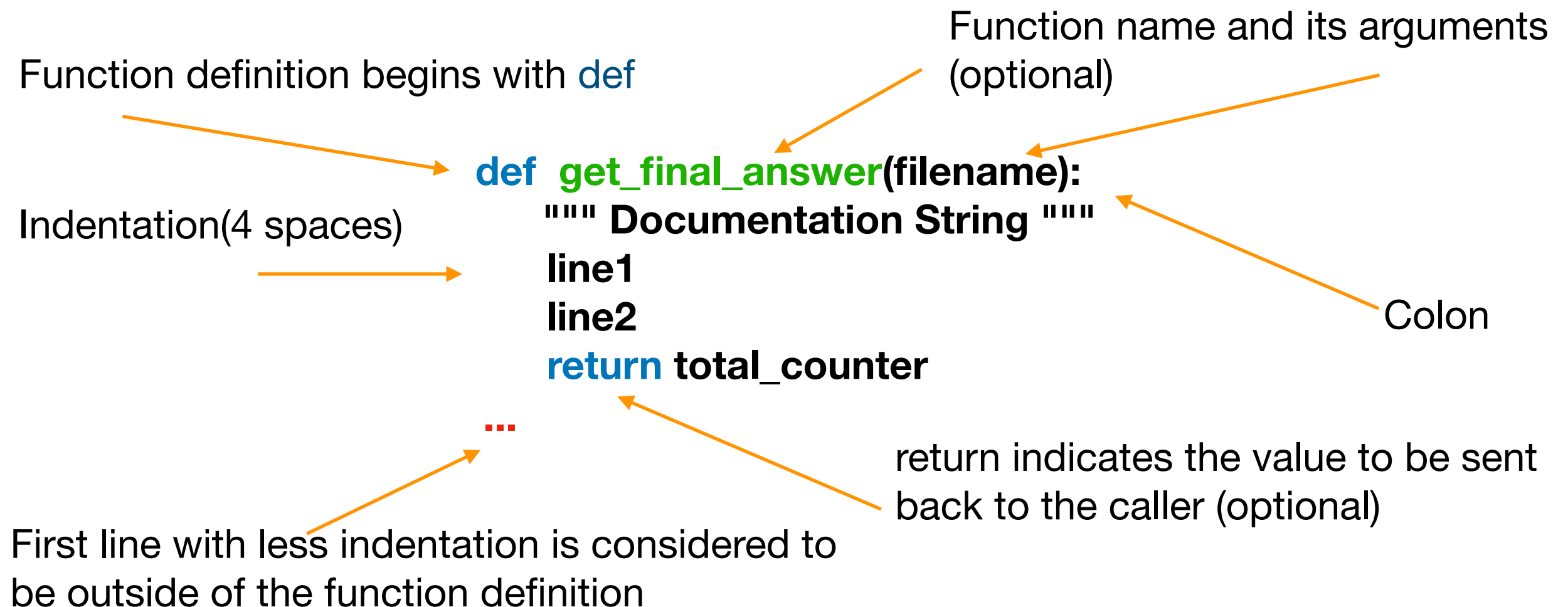


Agenda

- 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



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

```
1 def build_person(first_name, last_name):
2     """Return a dictionary of information about a person"""
3     person = {"first": first_name, "last": last_name}
4     print(person)
5
6
7 player = build_person("kobe", "bryant")
8 print(player)
9
10
```

functions x

/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**

```
1 x = lambda a : a + 10
2 print(x(5))
3
4 x = lambda a, b : a * b
5 print(x(5, 6))
6
7
```

functions ×

```
/Users/xinyi/anaconda/envs/mlern/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 x making_pizza.py x
1 def make_pizza(size, *toppings):
2     """Summarize the pizza we are about to make"""
3     print("\nMaking a " + str(size) +
4           "-inch pizza with the following toppings:")
5     for topping in toppings:
6         print("- " + topping)
```

```
pizza.py x making_pizza.py x
1 import pizza
2
3 pizza.make_pizza(16, "pepperoni", "extra cheese")
4
```

```
making_pizza x
/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



The screenshot shows a code editor with two tabs: 'pizza.py' and 'making_pizza.py'. The 'making_pizza.py' tab is active and contains the following code:

```
1 import pizza as pz
2
3 pz.make_pizza(16, "pepperoni", "extra cheese")
4
```

Below the code editor, the output of the script is displayed in a terminal window. The output shows the command to run the script and the resulting message:

```
making_pizza x
/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



```
1 from pizza import make_pizza
2
3 make_pizza(16, "pepperoni", "extra cheese")
4
```

making_pizza x

/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



The screenshot shows a code editor with two tabs: 'pizza.py' and 'making_pizza.py'. The 'making_pizza.py' tab is active and contains the following code:

```
1 from pizza import make_pizza as mp
2
3 mp(16, "pepperoni", "extra cheese")
4
```

Below the code editor is a terminal window titled 'making_pizza'. It shows the command to run the script and its output:

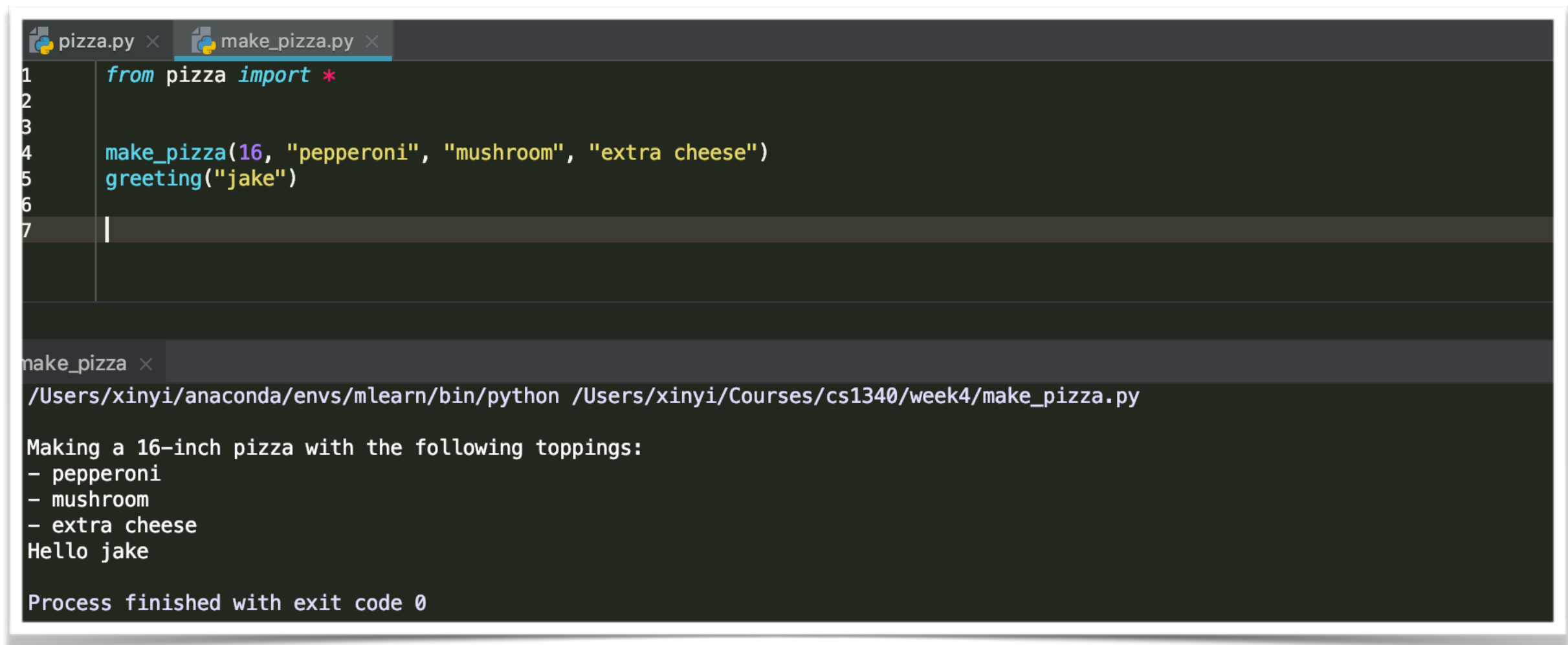
```
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week3/making_pizza.py

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



```
1 from pizza import *
2
3
4 make_pizza(16, "pepperoni", "mushroom", "extra cheese")
5 greeting("jake")
6
7
```

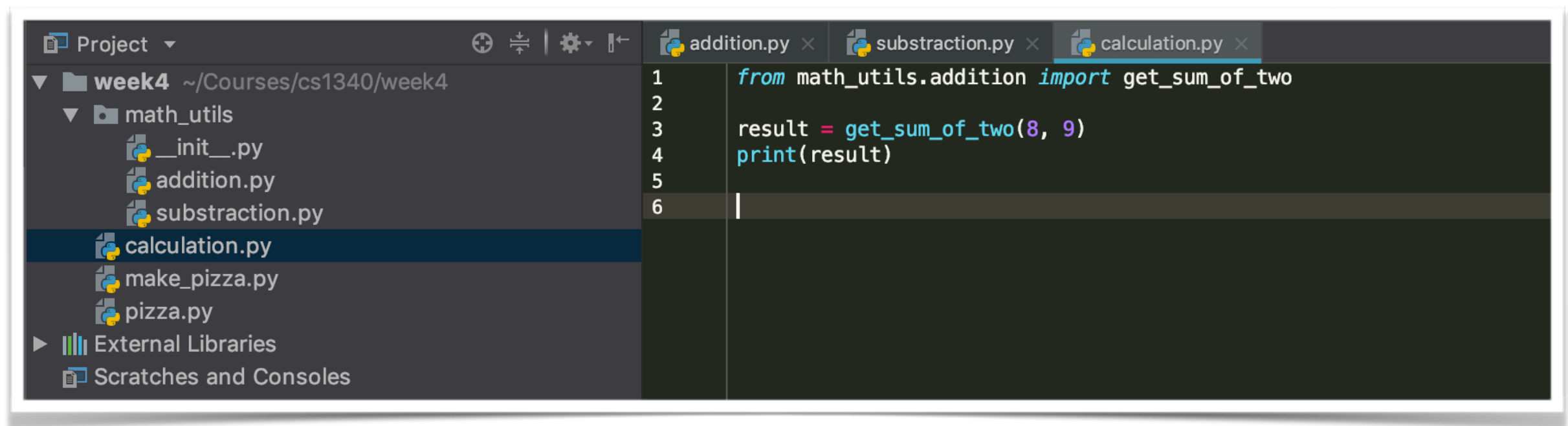
```
make_pizza x
/Users/xinyi/anaconda/envs/mlearn/bin/python /Users/xinyi/Courses/cs1340/week4/make_pizza.py

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

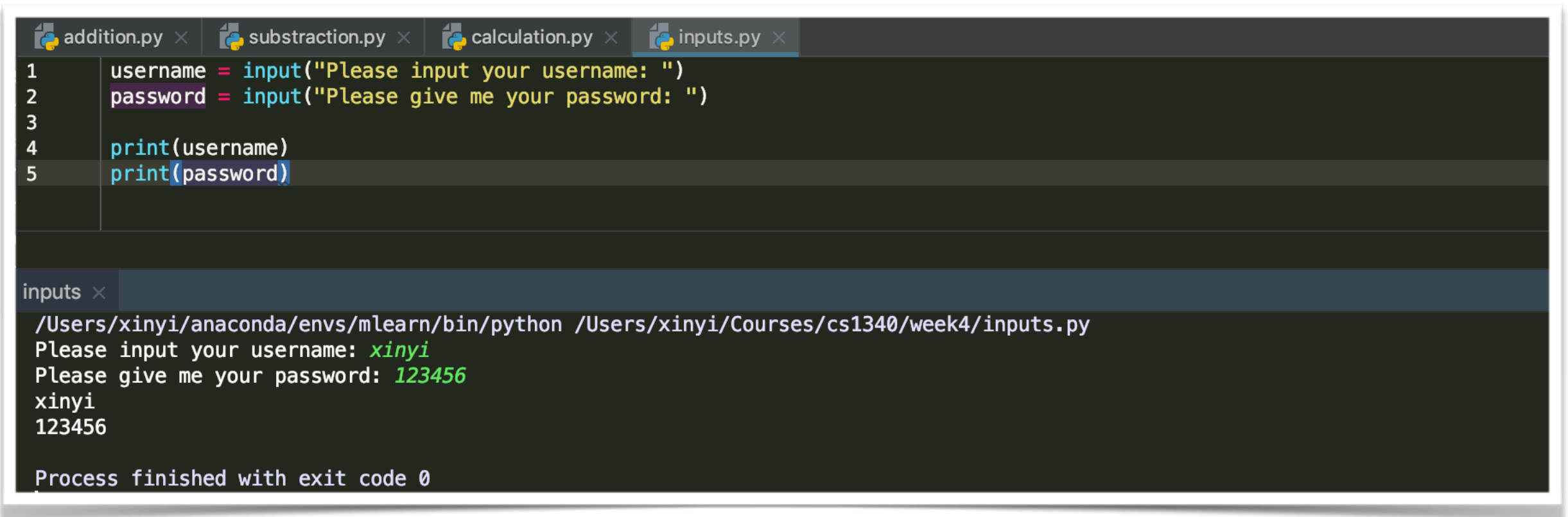
- Up till now, our program were static. The value of variables were defined or hard coded into the source code.
- To Allow flexibility we might want to take the input from the user. We use `input()` function in python

`input([prompt])`

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

Python Input/Output

- An simple example



The screenshot shows a Python IDE with four tabs: `addition.py`, `subtraction.py`, `calculation.py`, and `inputs.py`. The `inputs.py` tab is active, displaying the following code:

```
1 username = input("Please input your username: ")
2 password = input("Please give me your password: ")
3
4 print(username)
5 print(password)
```

Below the code editor, the terminal output for the `inputs.py` script is shown:

```
inputs x
/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.stdout, 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 formatting

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

```
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
```

```
1 print("I love {0} and {1}".format("bread", "butter"))
2 print("I love {1} and {0}".format("bread", "butter"))
3
```

```
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/>

```
1 pi = 3.1415926
2 print("The value of pi is %.2f"%pi)          # This is the old style formatting
3 print("The value of pi is {:.2f}".format(pi)) # This is the new and preferred one
4
```

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