Module

Dr. Dongchul Kim

What are Python Modules?

Definition: A Python module is a file containing Python definitions and statements. The file name is the module name with the suffix '.py' appended.

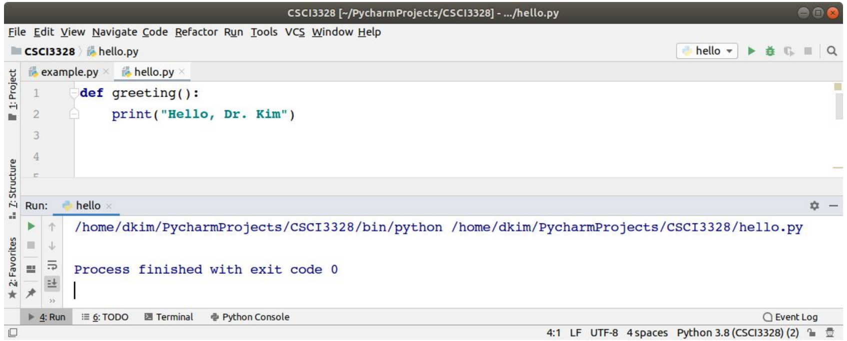
Role in Programming: Modules are a critical part of Python programming for organizing and reusing code. They help in breaking down complex processes into manageable pieces.

Create a Module

To create a module just save the code you want in a file with the file extension `.py`:

For example, save this code in a file named `hello.py`

```
def greeting():
    print("Hello, Dr. Kim")
```


CSCI3328 [~/PycharmProjects/CSCI3328] - .../hello.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 〉 hello.py
hello
example.py
hello.py
1: Project
Z: Structure
2: Favorites
Run:
hello
4: Run
6: TODO
Terminal
Python Console
Event Log
4:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

```
def greeting():
    print("Hello, Dr. Kim")
```

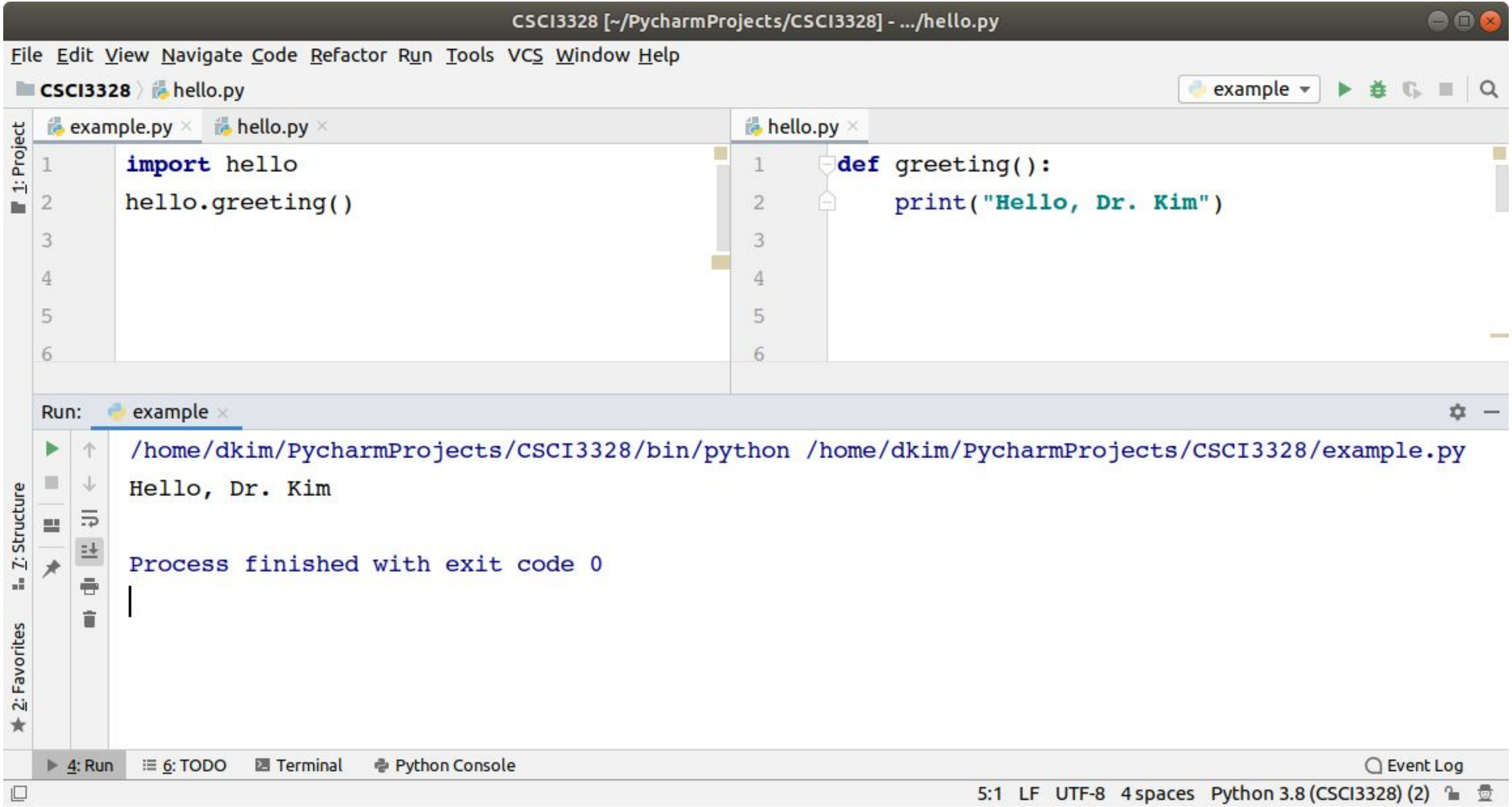
```
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/hello.py

Process finished with exit code 0
```

Use a Module

Now we can use the module we just created, by using the `import` statement:

For example, `import` the module named `hello`, and call the `greeting` function:

```
import hello

hello.greeting()
```


CSCI3328 [~/PycharmProjects/CSCI3328] - .../hello.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 〉 hello.py
example
example.py
hello.py
hello.py
1: Project
7: Structure
2: Favorites
Run:
example
4: Run
6: TODO
Terminal
Python Console
Event Log
5:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

```
import hello
hello.greeting()
```

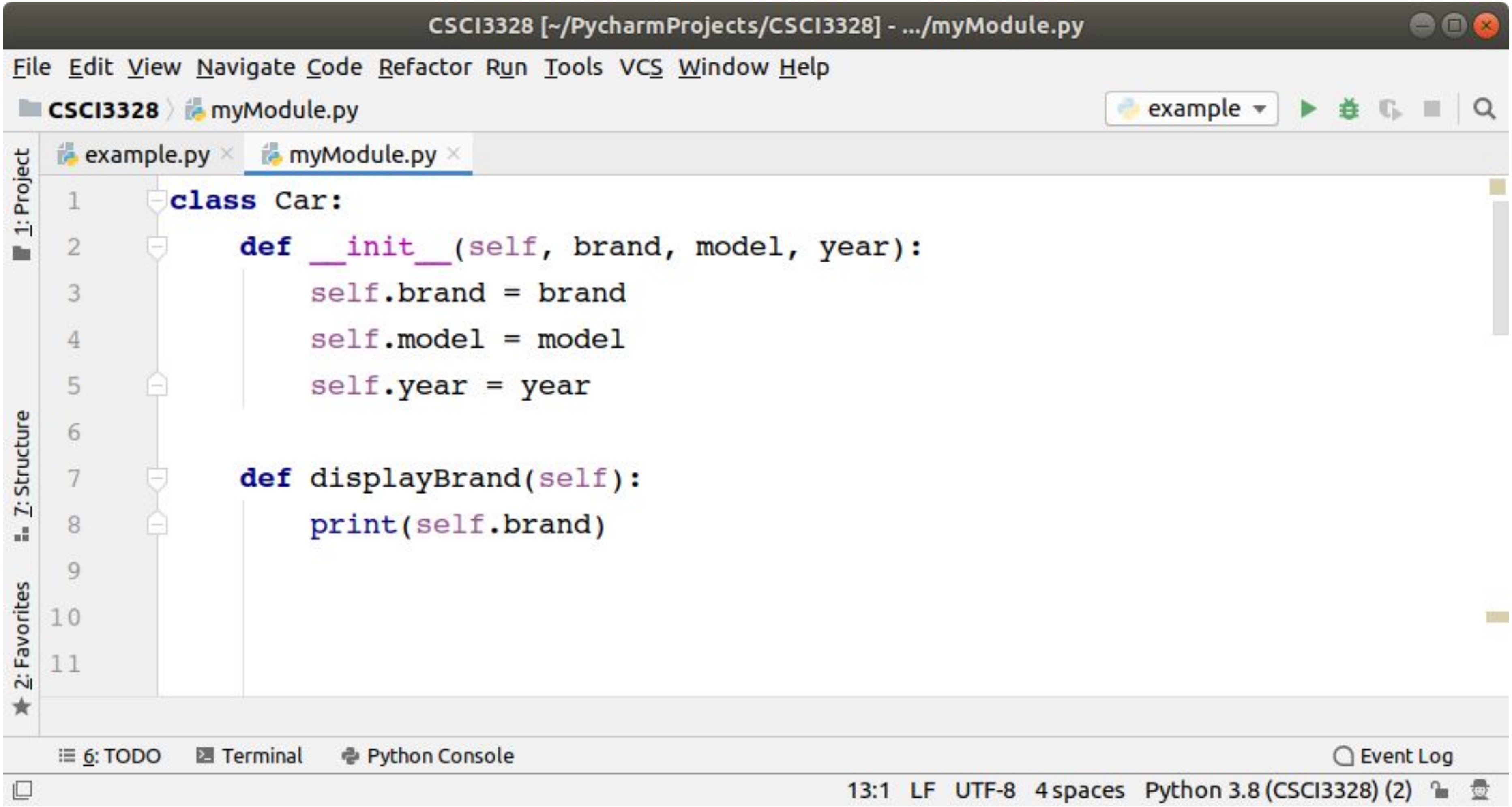
```
def greeting():
    print("Hello, Dr. Kim")
```

```
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py
Hello, Dr. Kim

Process finished with exit code 0
```

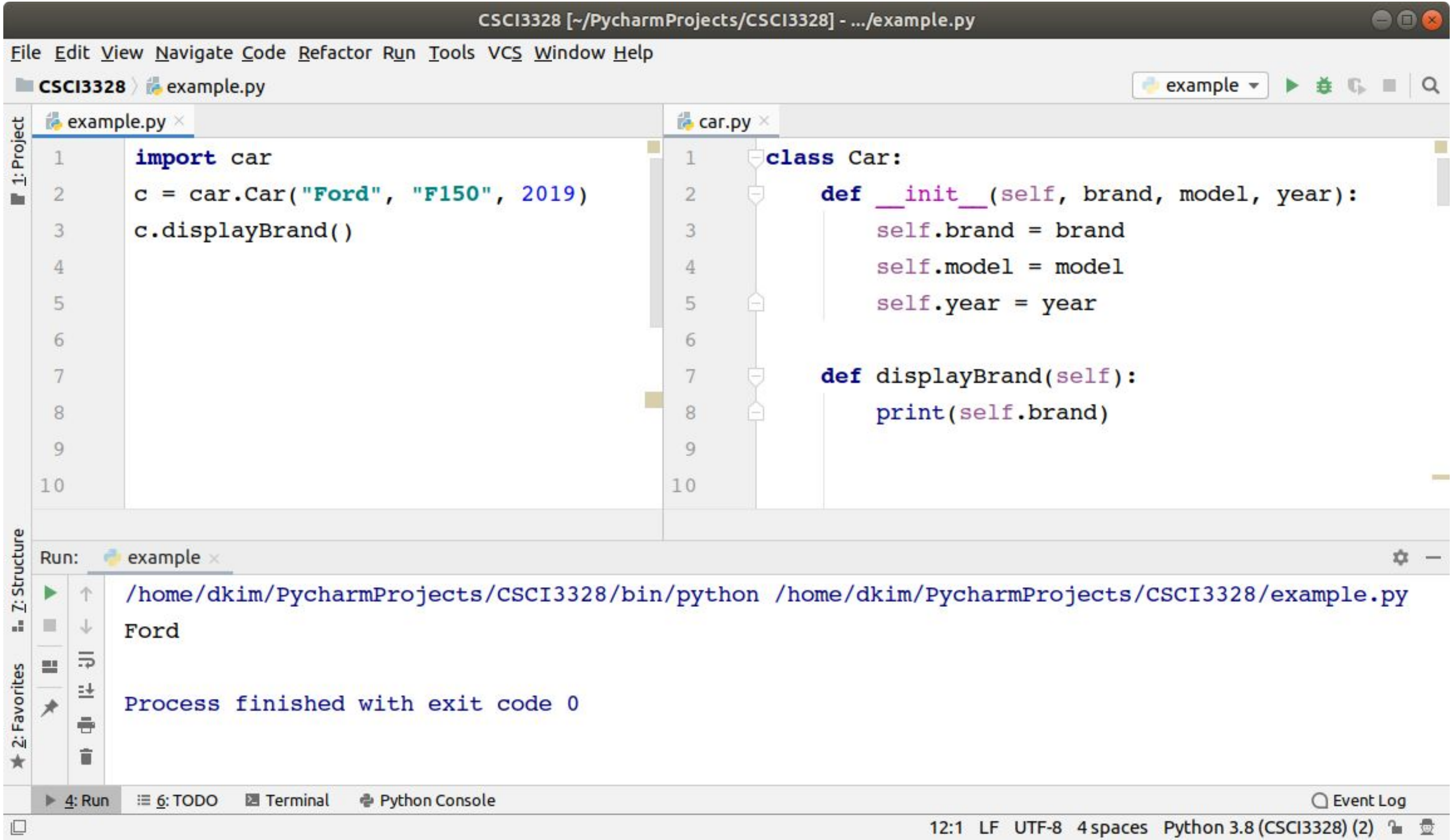
Example

Save this code in a file named `car.py`

```
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year

    def displayBrand(self):
        print(self.brand)
```


CSCI3328 [~/PycharmProjects/CSCI3328] - .../myModule.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328
myModule.py
example
example.py
myModule.py
1: Project
7: Structure
2: Favorites
6: TODO
Terminal
Python Console
Event Log
13:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

```python
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year

    def displayBrand(self):
        print(self.brand)


```


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 〉 example.py
example
1: Project
7: Structure
2: Favorites
example.py
car.py
Run: example
4: Run
6: TODO
Terminal
Python Console
Event Log
12:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

example.py

```python
import car
c = car.Car("Ford", "F150", 2019)
c.displayBrand()
```

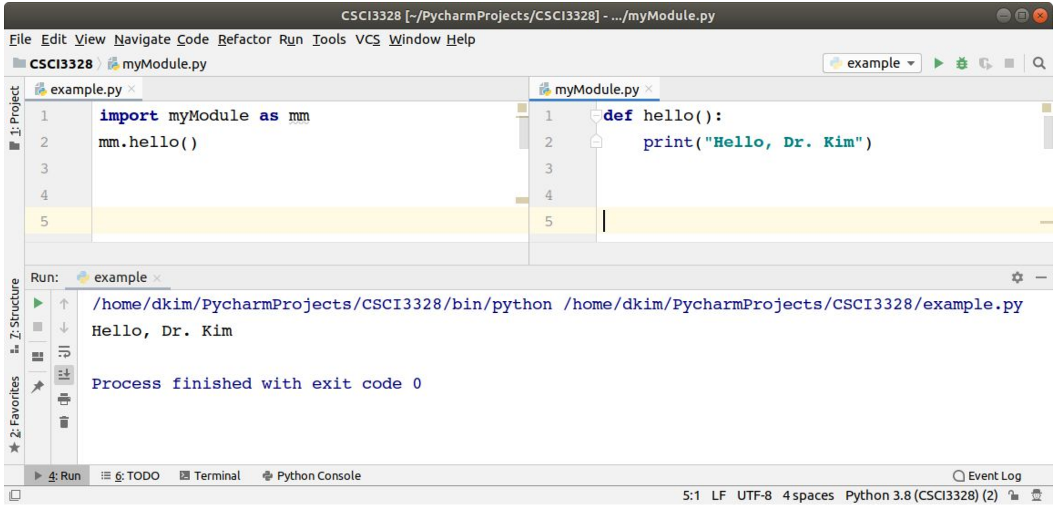
car.py

```python
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year

    def displayBrand(self):
        print(self.brand)
```

```
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py
Ford

Process finished with exit code 0
```

Renaming a Module

You can create an alias when you import a module, by using the `as` keyword:



CSCI3328 [~/PycharmProjects/CSCI3328] - .../myModule.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 myModule.py
example
example.py
myModule.py
1: Project
7: Structure
2: Favorites
Run: example
4: Run 6: TODO Terminal Python Console Event Log
5:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

example.py:

```python
import myModule as mm
mm.hello()
```

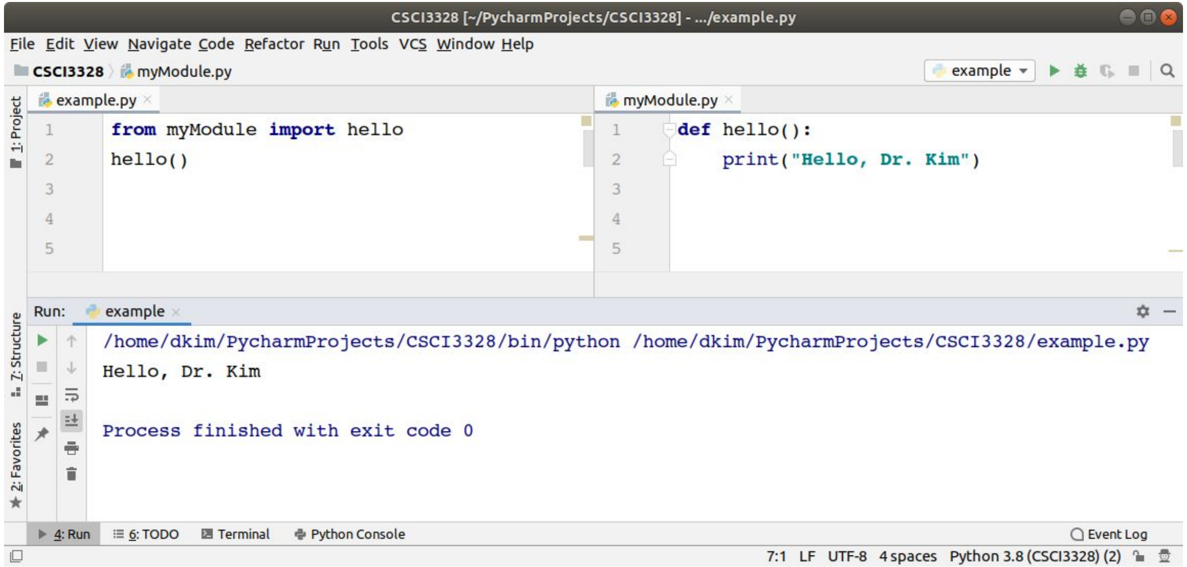
myModule.py:

```python
def hello():
    print("Hello, Dr. Kim")
```

```
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py
Hello, Dr. Kim

Process finished with exit code 0
```

Import From Module

You can choose to import only parts from a module, by using the `from` keyword.


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328
myModule.py
example
1: Project
7: Structure
2: Favorites
Run:
4: Run
6: TODO
Terminal
Python Console
Event Log
7:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

example.py

```python
from myModule import hello
hello()
```

myModule.py

```python
def hello():
    print("Hello, Dr. Kim")
```

Run: example

```
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py
Hello, Dr. Kim

Process finished with exit code 0
```

Lab 13-1

Make your own module and define a function in the module, then import the module and call the function in a python program.

The module name is `myModule` and the function name is `hello` that simply displays "`hello`".

Python Packages

A Python package is a collection of Python modules organized in a directory hierarchy.

Later in the course, we will explore Python Packages in more detail by learning to create and manage our own packages.

numpy

numpy

"NumPy," which stands for "Numeric Python" or "Numerical Python," is an extension module for Python primarily written in C.

This composition ensures that the precompiled mathematical and numerical functions provided by NumPy deliver exceptional execution speed.

NumPy is renowned for its support of large, multi-dimensional arrays and matrices, and it offers a vast array of high-level mathematical functions to work with these arrays.

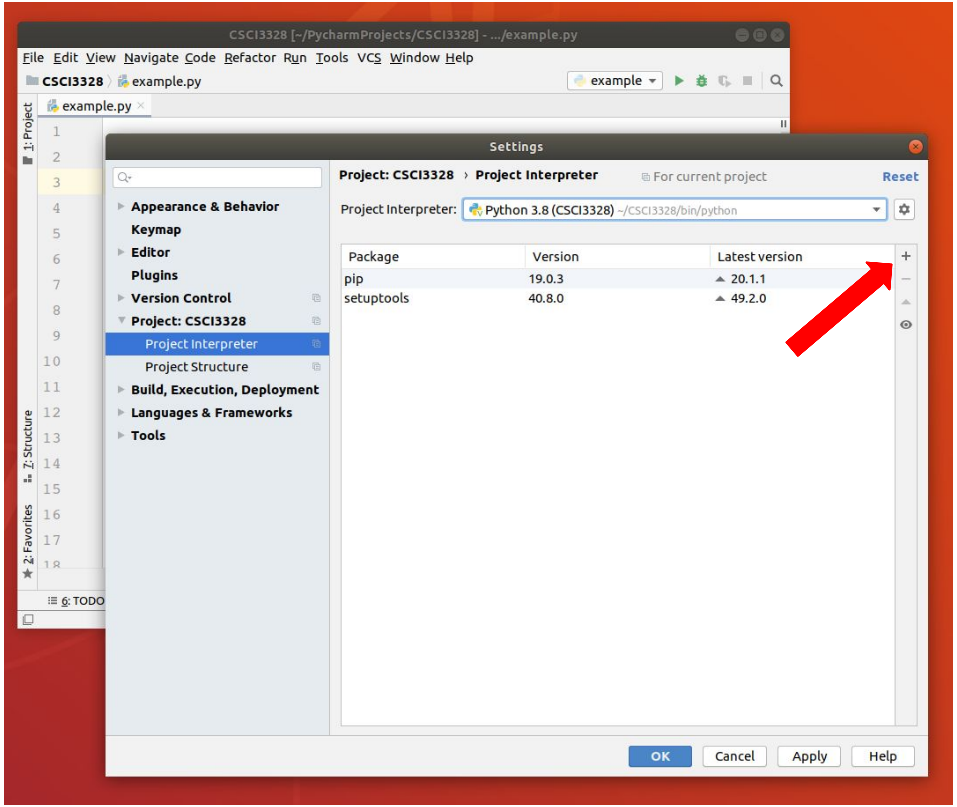
How to add a module in Pycharm

Go to **File** tab on the menu.

Select **Project: your_project_name** and click **Project Interpreter**.

It shows already installed packages.

Now, click **+** button on right side.



CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py

Settings

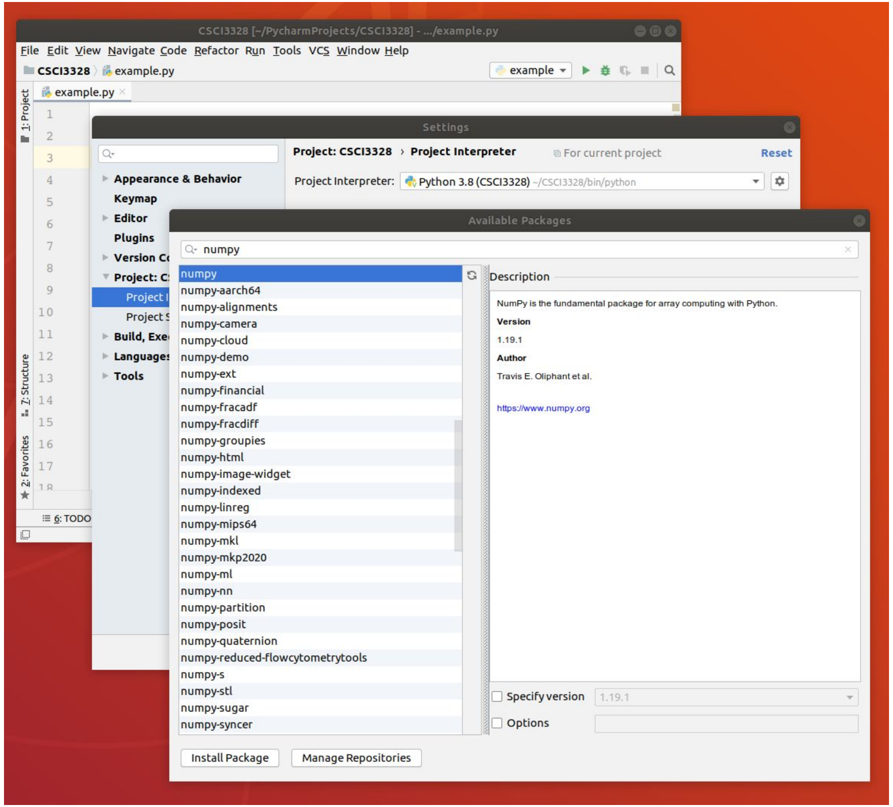
Project: CSCI3328 › Project Interpreter   For current project   Reset

Project Interpreter: Python 3.8 (CSCI3328) ~/CSCI3328/bin/python

Package	Version	Latest version
pip	19.0.3	▲ 20.1.1
setuptools	40.8.0	▲ 49.2.0

How to add a module in Pycharm

Type the package name you want to install on the top and click **Install Package** button.



Project Interpreter: Python 3.8 (CSCI3328) ~/CSCI3328/bin/python

numpy

numpy
numpy-aarch64
numpy-alignments
numpy-camera
numpy-cloud
numpy-demo
numpy-ext
numpy-financial
numpy-fracadf
numpy-fracdiff
numpy-groupies
numpy-html
numpy-image-widget
numpy-indexed
numpy-linreg
numpy-mips64
numpy-mkl
numpy-mkp2020
numpy-ml
numpy-nn
numpy-partition
numpy-posit
numpy-quaternion
numpy-reduced-flowcytometrytools
numpy-s
numpy-stl
numpy-sugar
numpy-syncer

NumPy is the fundamental package for array computing with Python.

Version

1.19.1

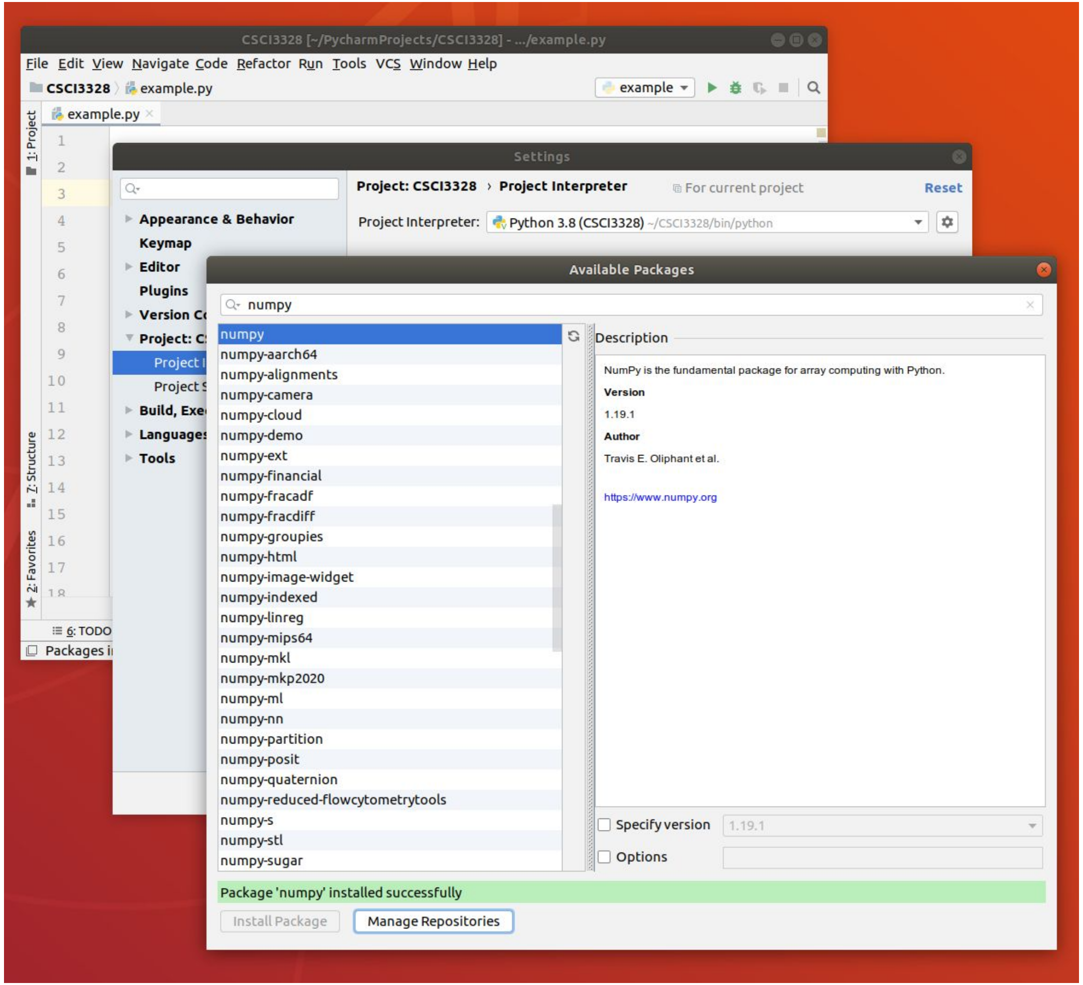
Author

Travis E. Oliphant et al.

https://www.numpy.org

□ Specify version 1.19.1

□ Options



Project: CSCI3328 › Project Interpreter

Project Interpreter: Python 3.8 (CSCI3328) ~/CSCI3328/bin/python

numpy

- numpy
- numpy-aarch64
- numpy-alignments
- numpy-camera
- numpy-cloud
- numpy-demo
- numpy-ext
- numpy-financial
- numpy-fracadf
- numpy-fracdiff
- numpy-groupies
- numpy-html
- numpy-image-widget
- numpy-indexed
- numpy-linreg
- numpy-mips64
- numpy-mkl
- numpy-mkp2020
- numpy-ml
- numpy-nn
- numpy-partition
- numpy-posit
- numpy-quaternion
- numpy-reduced-flowcytometrytools
- numpy-s
- numpy-stl
- numpy-sugar

Description

NumPy is the fundamental package for array computing with Python.

Version

1.19.1

Author

Travis E. Oliphant et al.

https://www.numpy.org

□ Specify version 1.19.1

□ Options

Package 'numpy' installed successfully

`numpy`

Before we can use `numpy` we will have to `import` it. It has to be imported like any other module:

```
import numpy
```

But you will hardly ever see this. `numpy` is usually renamed to `np`:

```
import numpy as np
```

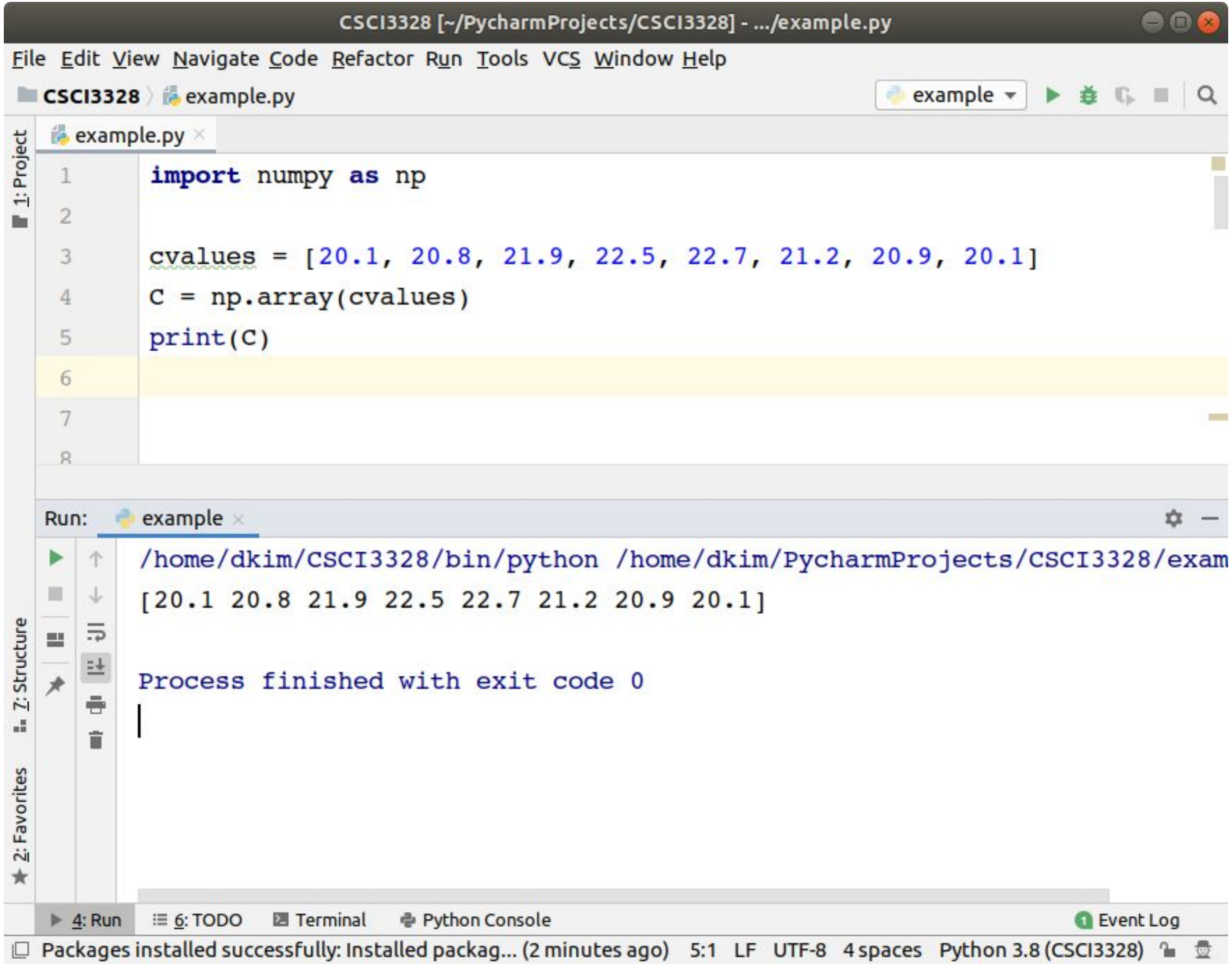
numpy

Our first simple Numpy example deals with temperatures. Given is a list with values, e.g. temperatures in Celsius:

```
cvalues = [20.1, 20.8, 21.9, 22.5, 22.7, 22.3, 21.8, 21.2, 20.9, 20.1]
```

We will turn our list "cvalues" into a one-dimensional numpy array:

```
C = np.array(cvalues)
```



```
import numpy as np

cvalues = [20.1, 20.8, 21.9, 22.5, 22.7, 21.2, 20.9, 20.1]
C = np.array(cvalues)
print(C)
```

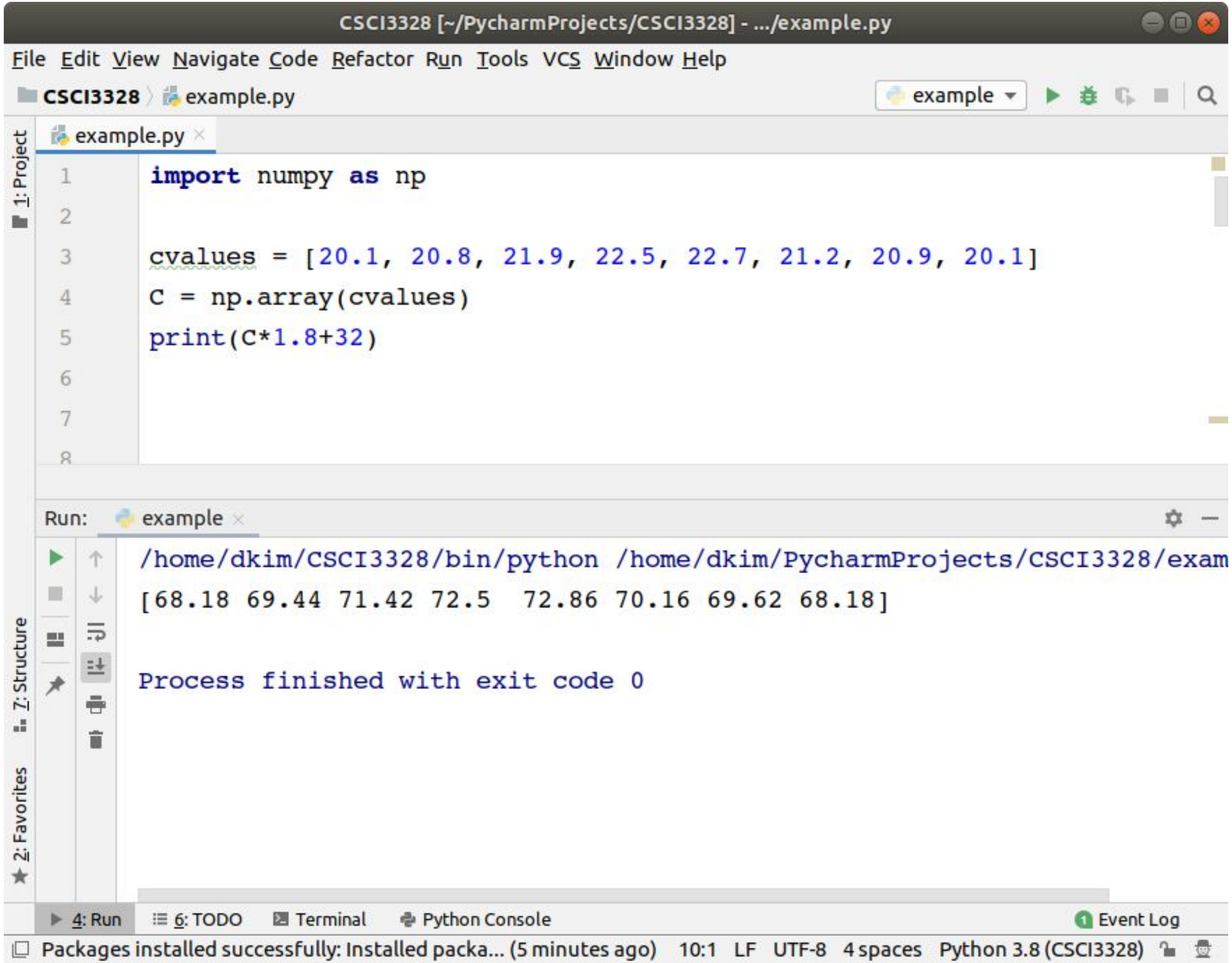
```
/home/dkim/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/exam
[20.1 20.8 21.9 22.5 22.7 21.2 20.9 20.1]

Process finished with exit code 0
```

Packages installed successfully: Installed packag... (2 minutes ago) 5:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328)

numpy

Let's assume, we want to turn the values into degrees Fahrenheit. This is very easy to accomplish with a numpy array. The solution to our problem can be achieved by simple scalar multiplication:

```
print(C * 9 / 5 + 32)
```



CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 › example.py
example
example.py ×
1: Project
Run: example ×
7: Structure
2: Favorites
4: Run
6: TODO
Terminal
Python Console
Event Log
Packages installed successfully: Installed packa... (5 minutes ago)
10:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328)

```python
import numpy as np

cvalues = [20.1, 20.8, 21.9, 22.5, 22.7, 21.2, 20.9, 20.1]
C = np.array(cvalues)
print(C*1.8+32)
```

```
/home/dkim/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/exam
[68.18 69.44 71.42 72.5  72.86 70.16 69.62 68.18]

Process finished with exit code 0
```

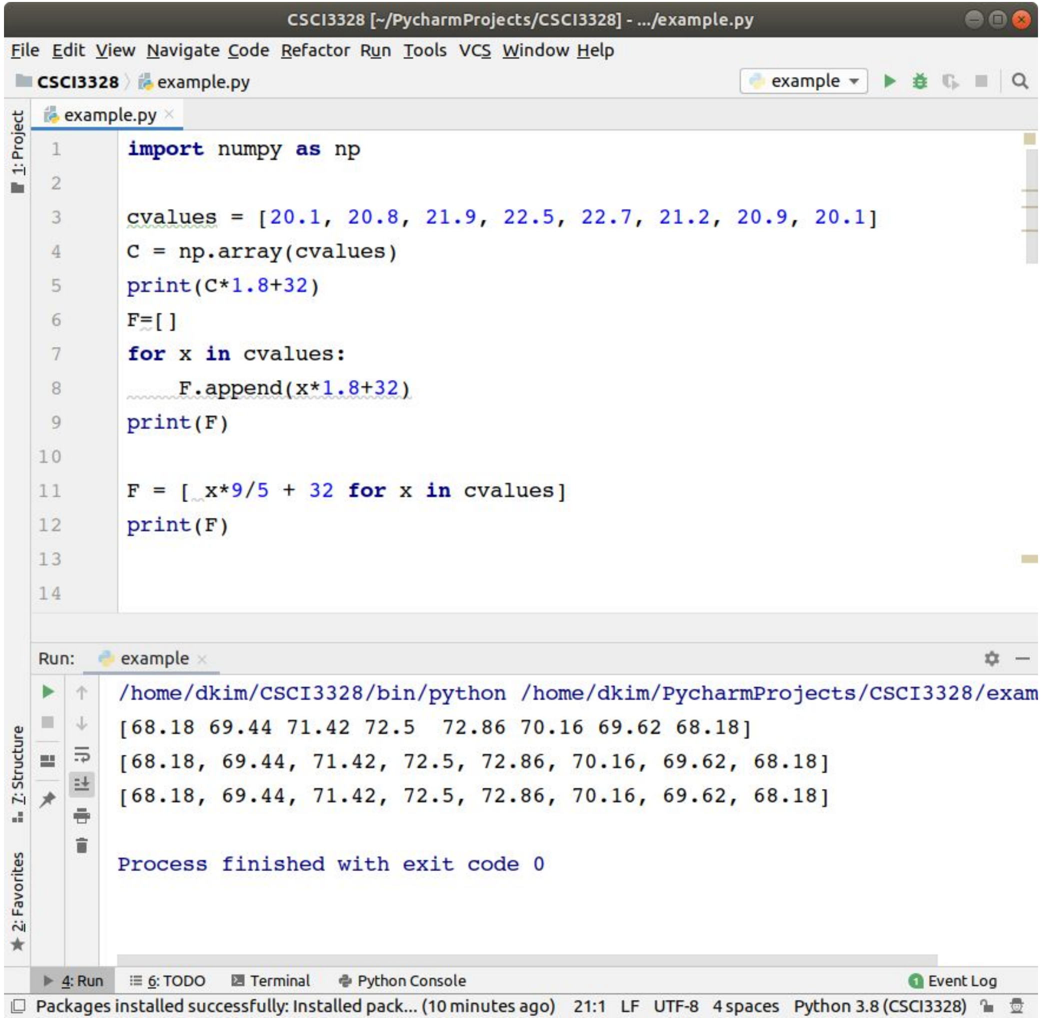
numpy

Compared to this, the solution for our Python list looks awkward:

```
F = [ x*9/5 + 32 for x in cvalues]
```

Or

```
F=[]

For x in cvalues:

    F.append(x*1.8+32)
```


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 example.py
example
example.py
1: Project
7: Structure
2: Favorites
Run: example
4: Run
6: TODO
Terminal
Python Console
Event Log
Packages installed successfully: Installed pack... (10 minutes ago)
21:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328)

```python
import numpy as np

cvalues = [20.1, 20.8, 21.9, 22.5, 22.7, 21.2, 20.9, 20.1]
C = np.array(cvalues)
print(C*1.8+32)
F=[]
for x in cvalues:
    F.append(x*1.8+32)
print(F)

F = [ x*9/5 + 32 for x in cvalues]
print(F)
```

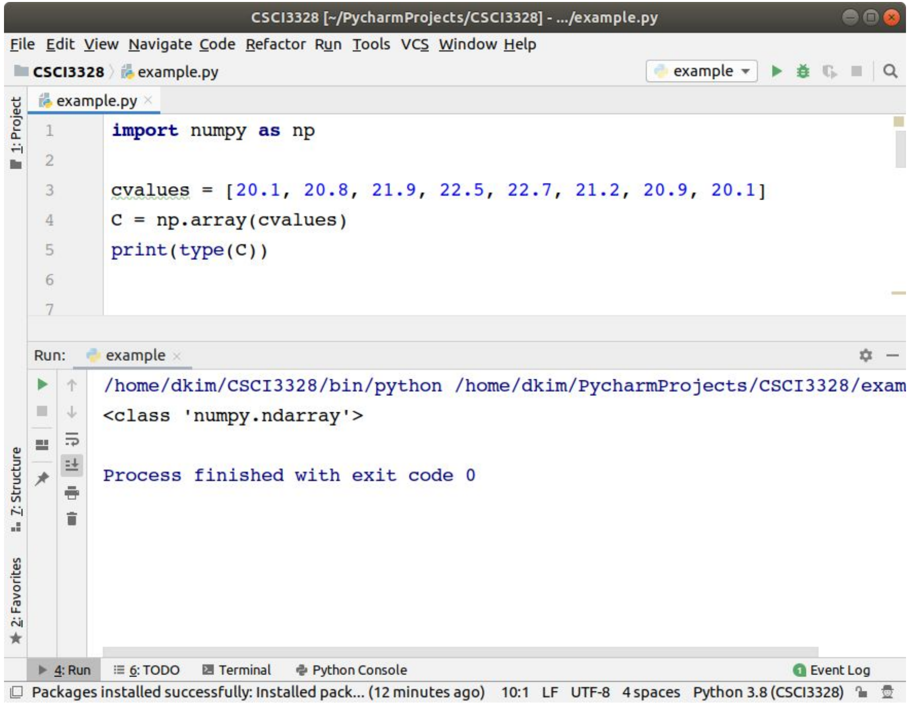
```
/home/dkim/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/exam
[68.18 69.44 71.42 72.5  72.86 70.16 69.62 68.18]
[68.18, 69.44, 71.42, 72.5, 72.86, 70.16, 69.62, 68.18]
[68.18, 69.44, 71.42, 72.5, 72.86, 70.16, 69.62, 68.18]

Process finished with exit code 0
```

numpy array

So far, we referred to C as an array. The internal type is "ndarray" or to be even more precise "C is an instance of the class numpy.ndarray":

The terms "array" and "ndarray" are used in most cases synonymously.

Normally we call it "**numpy array**".


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328
example.py
example
1: Project
7: Structure
2: Favorites
Run:
example
4: Run
6: TODO
Terminal
Python Console
Event Log
Packages installed successfully: Installed pack... (12 minutes ago)
10:1
LF
UTF-8
4 spaces
Python 3.8 (CSCI3328)

```python
import numpy as np

cvalues = [20.1, 20.8, 21.9, 22.5, 22.7, 21.2, 20.9, 20.1]
C = np.array(cvalues)
print(type(C))
```

```
/home/dkim/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/exam
<class 'numpy.ndarray'>

Process finished with exit code 0
```

Lab 13-2

Make a python program that convert Fahrenheit to Celsius given a list of float values in Fahrenheit using **numpy package** (Package is a set of modules).

Once you run the program, the program displays a list of values (at least 10) in Fahrenheit and then displays converted numpy array in Celsius.

Useful Numpy functions

np.array(): Creates an array.

np.arange(): Returns evenly spaced values within a given interval.

np.zeros(), np.ones(): Generate arrays of zeros or ones.

np.linspace(): Creates evenly spaced numbers over a specified interval.

np.reshape(): Gives a new shape to an array without changing its data.

np.sum(), np.mean(): Calculate the sum or mean of array elements.

np.std(): Computes the standard deviation.

np.dot(): Dot product of two arrays.

np.concatenate(): Joins a sequence of arrays along an existing axis.

np.sort(): Returns a sorted copy of an array.