



EXPLORER

▲ DIGITALCRAFTS

▶ .vscode

▶ Week1

▶ Week2

▶ Day 1

▶ Day 2



Show All Commands ⌘⌘P

Go to File ⌘T

Find in Files ⌘⌘F

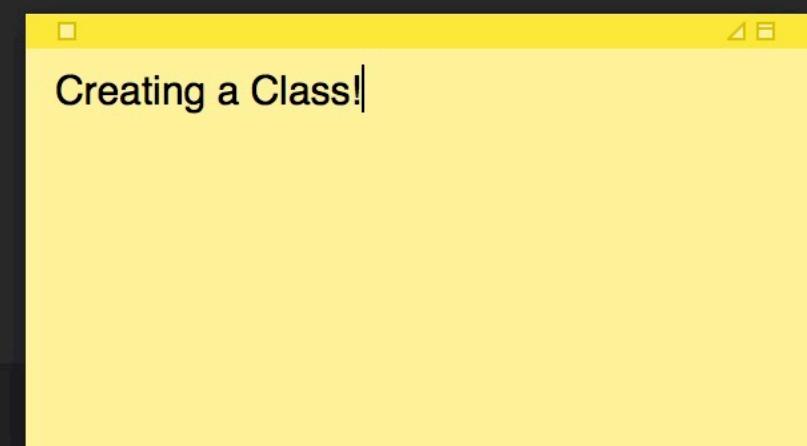
Start Debugging F5

Toggle Terminal ⌘`



PROJECTS

✖ 0 ▲ 0 Combo: 3 ⚙





EXPLORER



DIGITALCRAFTS



.vscode

Week1

Week2

Day 1

Day 2



person_class.py



person_class.py X

1



1

PROJECTS

0 ▲ 0 Combo: 3

First we want to create a new file to store our class in|

Ln 1, Col 1 Space



EXPLORER



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py



PROJECTS

person_class.py

```
1 class Person(object):  
2     ...
```



Like a function, we want to make a new definition but instead of 'def' to define a function, we use 'class' to define a class.



EXPLORER



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py



PROJECTS

person_class.py

```
1 class Person(object):
2     def __init__(self):
3         ...
```

Now, inside of that class, we can create our methods (functions) and properties (data).|

EXPLORER

DIGITALCRAFTS

- .vscode
- Week1
- Week2
 - Day 1
 - Day 2
 - person_class.py

PROJECTS

1

0 0 Combo: 1 Space

Ln 3, Col 9

person_class.py

```
1 class Person(object):
2     def __init__(self):
3         ...
```

Now, inside of that class, we can create our methods (functions) and properties (data). First, lets look at the '__init__' function.



EXPLORER



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py



PROJECTS

x 0 ▲ 0 Combo: 1 ⌂

person_class.py

```
1 class Person(object):
2     def __init__(self):
3         ...
```



The '`__init__`' class will be called as soon as a new instance of that class is created (sometimes called construction, or initialization).

We can use this function to set the default state of our class.

Ln 3, Col 9 Space



EXPLORER



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py



PROJECTS

x 0 ▲ 0

Combo: 1

person_class.py

```
1 class Person(object):
2     def __init__(self):
3         self.name = 'Janice'
4
5     ....
```

Ln 5, Col 5 Space

In this case, we know we want our Person class to have a name, so we set it in the constructor.



EXPLORER



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py



PROJECTS

0 0 0

Combo: 3

person_class.py

```
1 class Person(object):
2     def __init__(self):
3         self.name = 'Janice'
4
5     def greet(self):
6         print 'Hello, %s! Nice to meet you!' % (self.name)
7
```

File Browser

Now we can create a method that can be called off that object.

Let's make 'greet'

Ln 7, Col 1 (1 selected) Space



EXPLORER

person_class.py



DIGITAL CRAFTS

class Person(object):

1. python (Python)



python (Python) #1

~/Projects/DigitalCrafts/Week2/Day 2 \$ python [23:41:10]

Python 2.7.10 (default, Jul 15 2017, 17:16:57)

[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin

Type "help", "copyright", "credits" or "license" for more information.

>>> []



PROJECTS

x 0 ▲ 0

Combo: 3



Ln 1, Col 22 Space

class Person(object):

1. python (Python)

Now let's test that in the python console



EXPLORER

person_class.py



DIGITAL CRAFTS

python (Python) #1

~/Projects/DigitalCrafts/Week2/Day 2 \$ python [23:42:09]

```
Python 2.7.10 (default, Jul 15 2017, 17:16:57)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from person_class import Person
>>>
```



PROJECTS ▶

0 0 Combo: 3

Ln 1, Col 22 Space

First we import our class

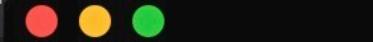


EXPLORER

person_class.py



DIGITAL CRAFTS



python (Python) #1

```
~/Projects/DigitalCrafts/Week2/Day 2 $ python [23:42:09]
Python 2.7.10 (default, Jul 15 2017, 17:16:57)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from person_class import Person
>>> janice = Person()
>>>
```



1

PROJECTS

x 0 ▲ 0

Combo: 3



Ln 1, Col 22 Space

Then we make a new instance of that class



EXPLORER

person_class.py



DIGITAL CRAFTS



python (Python) #1

```
~/Projects/DigitalCrafts/Week2/Day 2 $ python [23:42:09]
Python 2.7.10 (default, Jul 15 2017, 17:16:57)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from person_class import Person
>>> janice = Person()
>>> print janice.name
Janice
>>>
```



1

PROJECTS

x 0 ▲ 0

Combo: 3



Ln 7, Col 1 Space

Now we can print the name directly



EXPLORER

person_class.py



DIGITAL CRAFTS

python (Python) #1

```
~/Projects/DigitalCrafts/Week2/Day 2 $ python [23:42:09]
Python 2.7.10 (default, Jul 15 2017, 17:16:57)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from person_class import Person
>>> janice = Person()
>>> print janice.name
Janice
>>> janice.greet()
Hello, Janice! Nice to meet you!
>>>
```



PROJECTS

x 0 ▲ 0 Combo: 3 -o-

Ln 7, Col 1 Space

Or we can call the 'greet' method



EXPLORER

person_class.py



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py

person_class.pyc



PROJECTS

```
1 class Person(object):
2     def __init__(self, name):
3         self.name = name
4
5     def greet(self):
6         print 'Hello, %s! Nice to meet you!' % (self.name)
7
```

2 selections (4 characters selected) Space

Now, not every person is called Janice,
so let's make that accept a new variable
on initialization|

x 0 ▲ 0

Combo: 4



EXPLORER

person_class.py



DIGITAL CRAFTS

python (Python) #1

```
~/Projects/DigitalCrafts/Week2/Day 2 $ python [23:47:52]
Python 2.7.10 (default, Jul 15 2017, 17:16:57)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from person_class import Person
>>> bob = Person('Bob')
>>> karen = Person('Karen')
>>> janice = Person('Janice')
>>> 
```

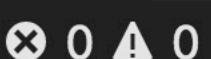


Now when we define a person, we can give them a unique name.

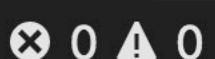
As you can see, I've created three new objects of the Person class with different names



PROJECTS



Combo: 1



Ln 4, Col 1 Space



EXPLORER

person_class.py



DIGITAL CRAFTS



python (Python) #1

class Person(object):

1. python (Python)

```
~/Projects/DigitalCrafts/Week2/Day 2 $ python [23:47:52]
Python 2.7.10 (default, Jul 15 2017, 17:16:57)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from person_class import Person
>>> bob = Person('Bob')
>>> karen = Person('Karen')
>>> janice = Person('Janice')
>>> bob.greet()
Hello, Bob! Nice to meet you!
>>> karen.greet()
Hello, Karen! Nice to meet you!
>>> janice.greet()
Hello, Janice! Nice to meet you!
>>>
```



1

PROJECTS

x 0 ▲ 0

Combo: 1



Ln 4, Col 1 Space

So when I call the greet method, they perform the same action, with slightly different results



EXPLORER



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py

person_class.pyc



person_class.py x

```
1 class Person(object):
2     def __init__(self, name):
3         self.name = name
4
5     def greet(self):
6         print 'Hello, %s! Nice to meet you!' % (self.name)
7
```



PROJECTS

x 0 ▲ 0 Combo: 1 ⌂

Ln 4, Col 1 Space

Let's add another method to introduce different people to each other, and then set their friends



EXPLORER

person_class.py



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py

person_class.pyc



```
1 class Person(object):
2     def __init__(self, name):
3         self.name = name
4
5     def greet(self):
6         print 'Hello, %s! Nice to meet you!' % (self.name)
7
8     def introduce(self, new_friend):
```

9

10



PROJECTS



It's going to take the 'self' argument as all class methods do, plus a second 'new_friend' argument

x 0 ▲ 0

Combo: 1



Ln 8, Col 37 Space



EXPLORER



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py

person_class.pyc



PROJECTS

x 0 ▲ 0

Combo: 32

person_class.py

```
1 class Person(object):
2     def __init__(self, name):
3         self.name = name
4         self.friends = []
5
6     def greet(self):
7         print 'Hello, %s! Nice to meet you!' % (self.name)
8
9     def introduce(self, new_friend):
```

Before we add new friends, we need to set the default list of friends to be an empty list

Ln 4, Col 21 Space



EXPLORER



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py

person_class.pyc



person_class.py

```
1 class Person(object):
2     def __init__(self, name):
3         self.name = name
4         self.friends = []
5
6     def greet(self):
7         print 'Hello, %s! Nice to meet you!' % (self.name)
8
9     def introduce(self, new_friend):
10        self.friends.append(new_friend.name)
```



PROJECTS

Now we can append the introduced friend to the Person's friends list!



EXPLORER



DIGITALCRAFTS

.vscode

Week1

Week2

Day 1

Day 2

person_class.py

person_class.pyc



person_class.py x

```
1 class Person(object):
2     def __init__(self, name):
3         self.name = name
4         self.friends = []
5
6     def greet(self):
7         print 'Hello, %s! Nice to meet you!' % (self.name)
8
9     def introduce(self, new_friend):
10        self.friends.append(new_friend.name)
11        print '%s made friends with %s' % (self.name, new_friend.name)
```



PROJECTS

x 0 ▲ 0 Combo: 1 ⌂

Ln 11, Col 71 Space

Let's add some output to show that it's working

The screenshot shows a dark-themed code editor interface. At the top, there's a navigation bar with icons for file operations, a search bar, and a tab labeled "EXPLORER". Below the navigation bar is a sidebar titled "DIGITAL CRAFTS" containing project-related icons and a "python (Python) #1" entry. The main area is a terminal window titled "person_class.py" with the file path "class Person(object)". The terminal output is as follows:

```
~/Projects/DigitalCrafts/Week2/Day 2 $ python [0:02:12]
Python 2.7.10 (default, Jul 15 2017, 17:16:57)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from person_class import Person
>>> bob = Person('Bob')
>>> karen = Person('Karen')
>>> 
```

The cursor is currently at the end of the last line of input. In the bottom right corner of the terminal window, there is a small yellow callout box with the text: "And then let's open the console again, import our class, define some Persons, and introduce them".

At the bottom of the screen, there are standard OS X-style system status icons for battery, signal, and volume, along with a "PROJECTS" section showing one item.

And then let's open the console again,
import our class, define some Persons,
and introduce them

The screenshot shows a Python code editor interface with a dark theme. On the left, there's a sidebar with icons for Explorer, Search, and Projects. The Projects section shows one item. At the bottom, there are status indicators for file changes (0), combo count (1), and space usage (Space).

The main area has tabs for 'EXPLORER' and 'person_class.py'. The 'person_class.py' tab is active, showing the following code:

```
class Person(object):
    def __init__(self, name):
        self.name = name
    def introduce(self, friend):
        print("%s made friends with %s" % (self.name, friend.name))
```

Below the code is a terminal window titled 'python (Python) #1' showing the execution of the code:

```
~/Projects/DigitalCrafts/Week2/Day 2 $ python
Python 2.7.10 (default, Jul 15 2017, 17:16:57)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from person_class import Person
>>> bob = Person('Bob')
>>> karen = Person('Karen')
>>> bob.introduce(karen)
Bob made friends with Karen
>>>
```

A yellow callout box in the bottom right corner contains the text: "And then let's open the console again, import our class, define some Persons, and introduce them".

And then let's open the console again, import our class, define some Persons, and introduce them