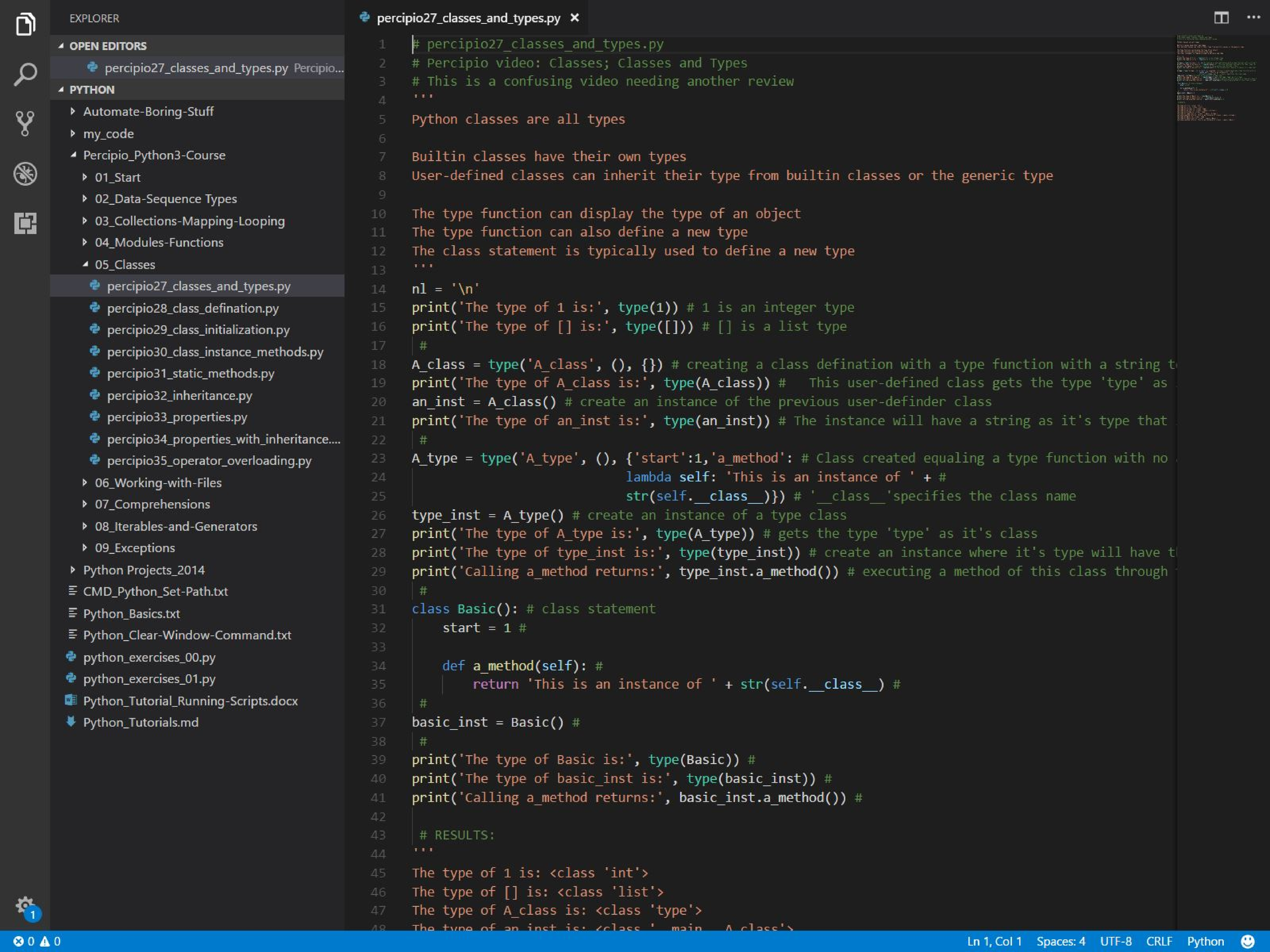


EXPLORER

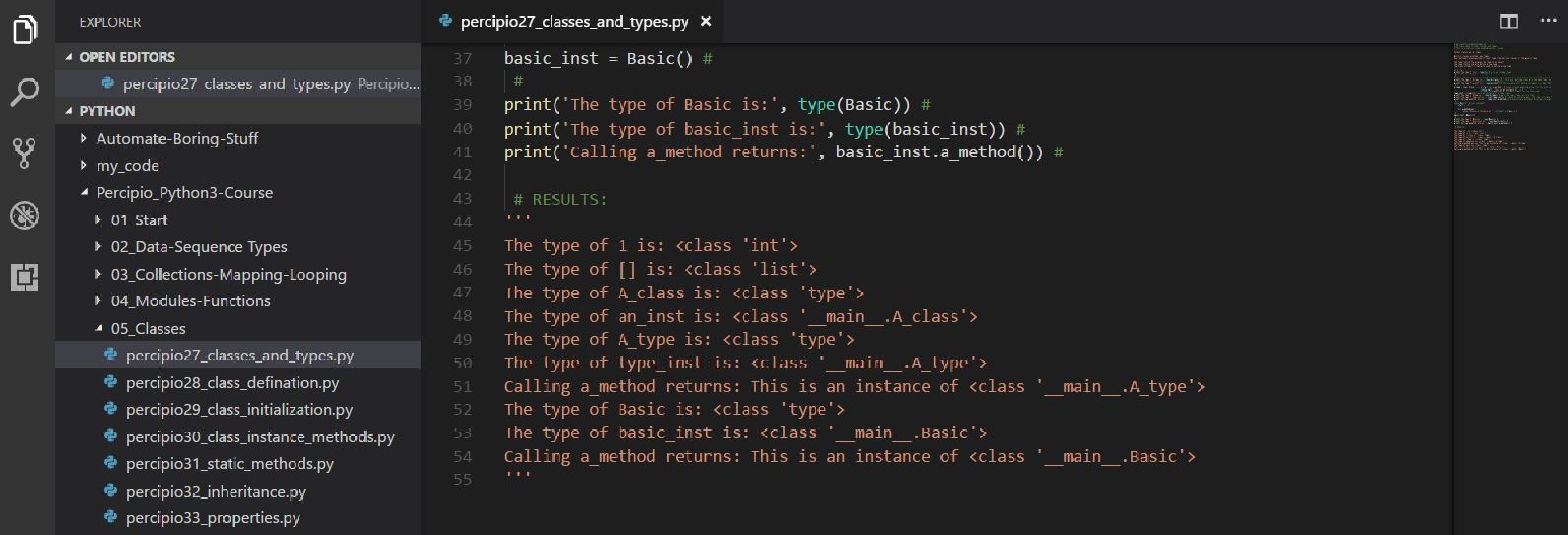
OPEN EDITORS
- percipio27_classes_and_types.py Percipio...

PYTHON
- Automate-Boring-Stuff
- my_code
- Percipio_Python3-Course
  - 01_Start
  - 02_Data-Sequence Types
  - 03_Collections-Mapping-Looping
  - 04_Modules-Functions
  - 05_Classes
    - percipio27_classes_and_types.py
    - percipio28_class_defination.py
    - percipio29_class_initialization.py
    - percipio30_class_instance_methods.py
    - percipio31_static_methods.py
    - percipio32_inheritance.py
    - percipio33_properties.py
    - percipio34_properties_with_inheritance....
    - percipio35_operator_overloading.py
  - 06_Working-with-Files
  - 07_Comprehensions
  - 08_Iterables-and-Generators
  - 09_Exceptions
- Python Projects_2014
- CMD_Python_Set-Path.txt
- Python_Basics.txt
- Python_Clear-Window-Command.txt
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Running-Scripts.docx
- Python_Tutorials.md

percipio27_classes_and_types.py

```python
# percipio27_classes_and_types.py
# Percipio video: Classes; Classes and Types
# This is a confusing video needing another review
'''
Python classes are all types

Builtin classes have their own types
User-defined classes can inherit their type from builtin classes or the generic type

The type function can display the type of an object
The type function can also define a new type
The class statement is typically used to define a new type
'''
nl = '\n'
print('The type of 1 is:', type(1)) # 1 is an integer type
print('The type of [] is:', type([])) # [] is a list type
#
A_class = type('A_class', (), {}) # creating a class defination with a type function with a string t
print('The type of A_class is:', type(A_class)) #   This user-defined class gets the type 'type' as
an_inst = A_class() # create an instance of the previous user-definder class
print('The type of an_inst is:', type(an_inst)) # The instance will have a string as it's type that
#
A_type = type('A_type', (), {'start':1,'a_method': # Class created equaling a type function with no
                            lambda self: 'This is an instance of ' + #
                            str(self.__class__)}) # '__class__'specifies the class name
type_inst = A_type() # create an instance of a type class
print('The type of A_type is:', type(A_type)) # gets the type 'type' as it's class
print('The type of type_inst is:', type(type_inst)) # create an instance where it's type will have t
print('Calling a_method returns:', type_inst.a_method()) # executing a method of this class through
#
class Basic(): # class statement
    start = 1 #

    def a_method(self): #
        return 'This is an instance of ' + str(self.__class__) #
#
basic_inst = Basic() #

#
print('The type of Basic is:', type(Basic)) #
print('The type of basic_inst is:', type(basic_inst)) #
print('Calling a_method returns:', basic_inst.a_method()) #

 # RESULTS:
'''
The type of 1 is: <class 'int'>
The type of [] is: <class 'list'>
The type of A_class is: <class 'type'>
The type of an_inst is: <class '__main__.A_class'>
```

⊗ 0 ⚠ 0 Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python



EXPLORER
OPEN EDITORS
percipio27_classes_and_types.py Percipio...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
percipio27_classes_and_types.py
percipio28_class_defination.py
percipio29_class_initialization.py
percipio30_class_instance_methods.py
percipio31_static_methods.py
percipio32_inheritance.py
percipio33_properties.py
percipio27_classes_and_types.py

```python
basic_inst = Basic() #
 #
print('The type of Basic is:', type(Basic)) #
print('The type of basic_inst is:', type(basic_inst)) #
print('Calling a_method returns:', basic_inst.a_method()) #

 # RESULTS:
'''
The type of 1 is: <class 'int'>
The type of [] is: <class 'list'>
The type of A_class is: <class 'type'>
The type of an_inst is: <class '__main__.A_class'>
The type of A_type is: <class 'type'>
The type of type_inst is: <class '__main__.A_type'>
Calling a_method returns: This is an instance of <class '__main__.A_type'>
The type of Basic is: <class 'type'>
The type of basic_inst is: <class '__main__.Basic'>
Calling a_method returns: This is an instance of <class '__main__.Basic'>
'''
```