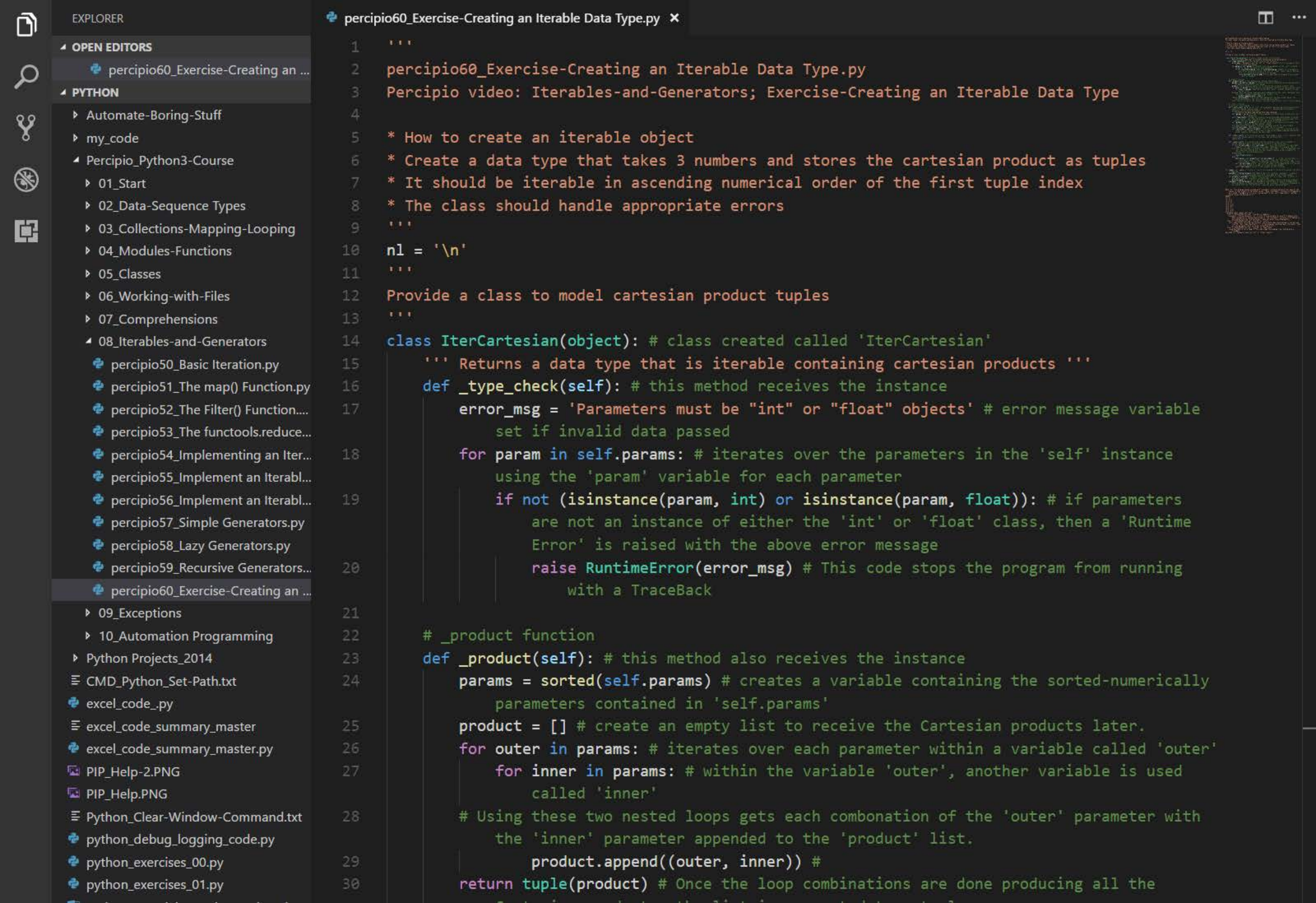

EXPLORER
percipio60_Exercise-Creating an Iterable Data Type.py
OPEN EDITORS
percipio60_Exercise-Creating an ...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
08_Iterables-and-Generators
percipio50_Basic Iteration.py
percipio51_The map() Function.py
percipio52_The Filter() Function....
percipio53_The functools.reduce...
percipio54_Implementing an Iter...
percipio55_Implement an Iterabl...
percipio56_Implement an Iterabl...
percipio57_Simple Generators.py
percipio58_Lazy Generators.py
percipio59_Recursive Generators...
percipio60_Exercise-Creating an ...
09_Exceptions
10_Automation Programming
Python Projects_2014
CMD_Python_Set-Path.txt
excel_code_.py
excel_code_summary_master
excel_code_summary_master.py
PIP_Help-2.PNG
PIP_Help.PNG
Python_Clear-Window-Command.txt
python_debug_logging_code.py
python_exercises_00.py
python_exercises_01.py
'''
percipio60_Exercise-Creating an Iterable Data Type.py
Percipio video: Iterables-and-Generators; Exercise-Creating an Iterable Data Type
* How to create an iterable object
* Create a data type that takes 3 numbers and stores the cartesian product as tuples
* It should be iterable in ascending numerical order of the first tuple index
* The class should handle appropriate errors
'''
nl = '\n'
'''
Provide a class to model cartesian product tuples
'''
class IterCartesian(object): # class created called 'IterCartesian'
''' Returns a data type that is iterable containing cartesian products '''
def _type_check(self): # this method receives the instance
error_msg = 'Parameters must be "int" or "float" objects' # error message variable
set if invalid data passed
for param in self.params: # iterates over the parameters in the 'self' instance
using the 'param' variable for each parameter
if not (isinstance(param, int) or isinstance(param, float)): # if parameters
are not an instance of either the 'int' or 'float' class, then a 'Runtime
Error' is raised with the above error message
raise RuntimeError(error_msg) # This code stops the program from running
with a TraceBack
_product function
def _product(self): # this method also receives the instance
params = sorted(self.params) # creates a variable containing the sorted-numerically
parameters contained in 'self.params'
product = [] # create an empty list to receive the Cartesian products later.
for outer in params: # iterates over each parameter within a variable called 'outer'
for inner in params: # within the variable 'outer', another variable is used
called 'inner'
Using these two nested loops gets each combonation of the 'outer' parameter with
the 'inner' parameter appended to the 'product' list.
product.append((outer, inner)) #
return tuple(product) # Once the loop combinations are done producing all the





EXPLORER

percipio60_Exercise-Creating an Iterable Data Type.py

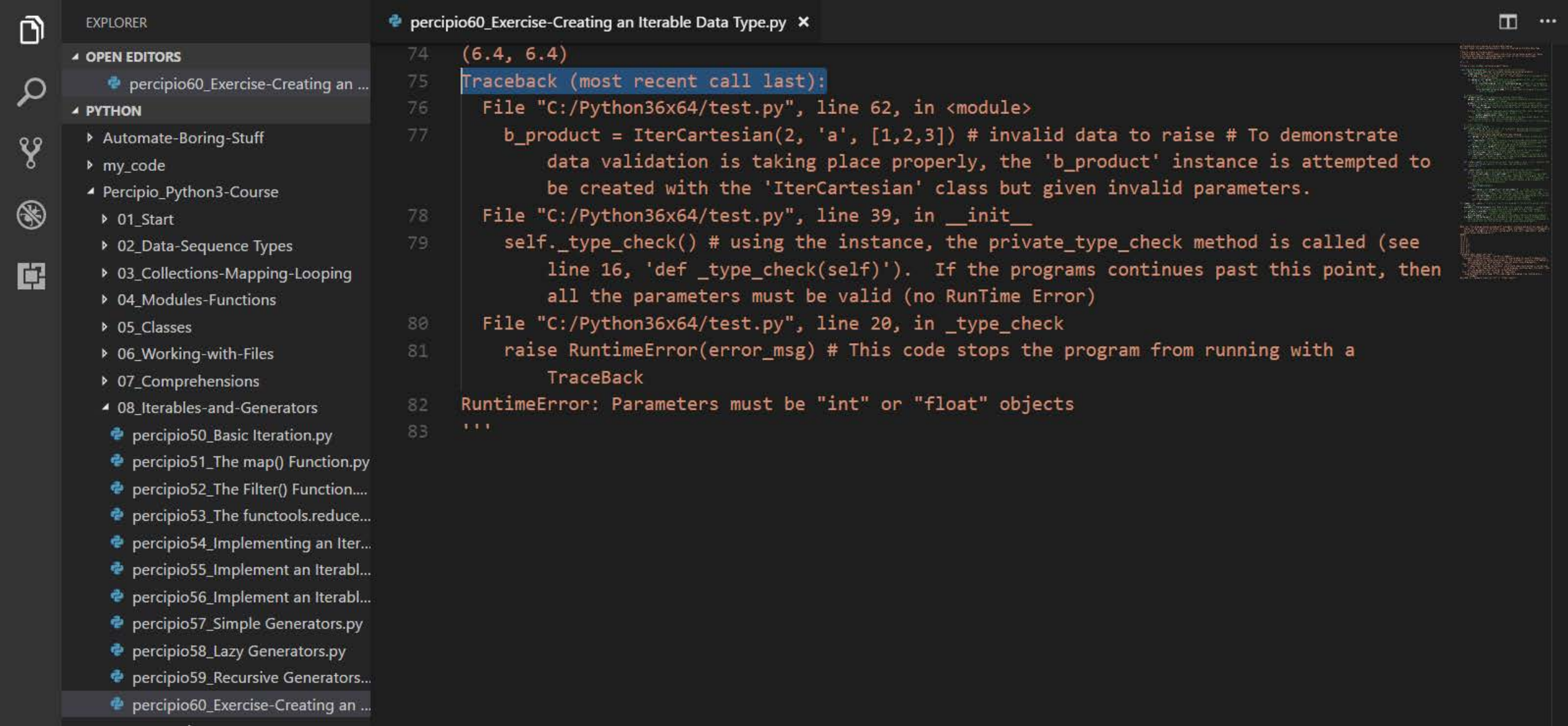
OPEN EDITORS
- percipio60_Exercise-Creating an ...

PYTHON
- Automate-Boring-Stuff
- my_code
- Percipio_Python3-Course
  - 01_Start
  - 02_Data-Sequence Types
  - 03_Collections-Mapping-Looping
  - 04_Modules-Functions
  - 05_Classes
  - 06_Working-with-Files
  - 07_Comprehensions
  - 08_Iterables-and-Generators
    - percipio50_Basic Iteration.py
    - percipio51_The map() Function.py
    - percipio52_The Filter() Function....
    - percipio53_The functools.reduce...
    - percipio54_Implementing an Iter...
    - percipio55_Implement an Iterabl...
    - percipio56_Implement an Iterabl...
    - percipio57_Simple Generators.py
    - percipio58_Lazy Generators.py
    - percipio59_Recursive Generators...
    - percipio60_Exercise-Creating an ...
  - 09_Exceptions
  - 10_Automation Programming
- Python Projects_2014
- CMD_Python_Set-Path.txt
- excel_code_.py
- excel_code_summary_master
- excel_code_summary_master.py
- PIP_Help-2.PNG
- PIP_Help.PNG
- Python_Clear-Window-Command.txt
- python_debug_logging_code.py
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Running-Scripts.docx
- Python_Tutorials.md

```
30          return tuple(product) # Once the loop combinations are done producing all the
                Cartesian products, the list is converted to a tuple.
31          # all this is stored in the 'product' attribute used in the __init__ function below.
32
33      # __init__ function
34      # starts off this program
35      def __init__(self, a, b, c): #  'self' is a variable representing the instance itself
            implicitly passed, and variables 'a', 'b', & 'c' representing three explicit
            parameters which should be numbers.
36          ''' Initialize the data type and perform type checking '''
37          self.params = (a, b, c) # the 'instance.params' attribute holds a tuple with the 3
                'a', 'b', & 'c' parameters
38          self._type_check() # using the instance, the private_type_check method is called
                (see line 16, 'def _type_check(self)').  If the programs continues past this
                point, then all the parameters must be valid (no RunTime Error)
39          self.product = self._product() # the 'self.product' attribute is calculated from
                the return value of 'product method' (above)
40          # To make iteration possible, different variables are set for the instance.
41          self.current_index = 0 # begins at 0 to index the first element
42          self.last_index = len(self.product) - 1 # 'self.last_index' is calculated as the
                length of 'self.product' minus one (-1 due to zero-based indexing leaving the
                actual length 1 too long as the last index)
43
44      def __iter__(self): # for iteration to work, there needs to be a __iter__ method() that
            returns an iterable object, in this case, the instance itself.
45          return self #
46
47      def __next__(self): # to actually perform the iteration, the __next__ method() is
            called repeatedly which references the instance of the object and,...
48          if self.current_index > self.last_index: # ...with that instances 'current_index'
                compare it to that instances 'last_index' checking if it's greater then it.
49          # If the 'current_index' is greater than the 'last_index', then it's gone past the
                last index and iterated over the last element leaving no more elements to
                iterate over raising StopIteration.
50          # Otherwise,...
51              raise StopIteration #
52          else: #
53              next_value = self.product[self.current_index] # ... if the 'current_index' is
                    less than or equal to the 'last_index', than the 'next_value' from
                    'self_product' tuple is retrieved by referencing 'self.current_index slice.
```

OPEN EDITORS
percipio60_Exercise-Creating an ...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
08_Iterables-and-Generators
percipio50_Basic Iteration.py
percipio51_The map() Function.py
percipio52_The Filter() Function....
percipio53_The functools.reduce...
percipio54_Implementing an Iter...
percipio55_Implement an Iterabl...
percipio56_Implement an Iterabl...
percipio57_Simple Generators.py
percipio58_Lazy Generators.py
percipio59_Recursive Generators...
percipio60_Exercise-Creating an ...
09_Exceptions
10_Automation Programming
Python Projects_2014
CMD_Python_Set-Path.txt
excel_code_.py
excel_code_summary_master
excel_code_summary_master.py

```
                    'self_product' tuple is retrieved by referencing 'self.current_index slice.
54                  self.current_index += 1 # ' self.current_index' is incremented by 1 so the next
                        time this method is called, it is onto the next element.
55                  return next_value # return th next element in this iteration
56
57  if __name__ == '__main__': # only is True if this program is run directly and will be False
        if imported
58      a_product = IterCartesian(2, 4.2, 6.4) # the first instance, 'a_product', creates a
            instance of the IterCartesian class with valid data of 3 different numbers
59      for a_tuple in a_product: # demonstrate iteration with a forLoop, that 'a_tuple' is
            available from that 'a_product' instance and...
60          print(a_tuple) # ,,,  'print' out each of the tuples that are Cartesian products.
61      b_product = IterCartesian(2, 'a', [1,2,3]) # invalid data to raise # To demonstrate
            data validation is taking place properly, the 'b_product' instance is attempted to
            be created with the 'IterCartesian' class but given invalid parameters.
62
63  '''
64  When run, this program should succeed with 'a_product' instance and print out each of the
        tuples in that product of Cartesian values.  When 'b_prooduct' instance is attempted to
        be created, TraceBack errors occur tracing back to the 'self._type_check()' method
        which raises the Runtime error.
65  RESULT:
66  (2, 2)
67  (2, 4.2)
68  (2, 6.4)
69  (4.2, 2)
70  (4.2, 4.2)
71  (4.2, 6.4)
72  (6.4, 2)
73  (6.4, 4.2)
74  (6.4, 6.4)
75  Traceback (most recent call last):
```


EXPLORER
OPEN EDITORS
percipio60_Exercise-Creating an ...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
08_Iterables-and-Generators
percipio50_Basic Iteration.py
percipio51_The map() Function.py
percipio52_The Filter() Function....
percipio53_The functools.reduce...
percipio54_Implementing an Iter...
percipio55_Implement an Iterabl...
percipio56_Implement an Iterabl...
percipio57_Simple Generators.py
percipio58_Lazy Generators.py
percipio59_Recursive Generators...
percipio60_Exercise-Creating an ...
percipio60_Exercise-Creating an Iterable Data Type.py
74 (6.4, 6.4)
75 Traceback (most recent call last):
76 File "C:/Python36x64/test.py", line 62, in <module>
77 b_product = IterCartesian(2, 'a', [1,2,3]) # invalid data to raise # To demonstrate data validation is taking place properly, the 'b_product' instance is attempted to be created with the 'IterCartesian' class but given invalid parameters.
78 File "C:/Python36x64/test.py", line 39, in __init__
79 self._type_check() # using the instance, the private_type_check method is called (see line 16, 'def _type_check(self)'). If the programs continues past this point, then all the parameters must be valid (no RunTime Error)
80 File "C:/Python36x64/test.py", line 20, in _type_check
81 raise RuntimeError(error_msg) # This code stops the program from running with a TraceBack
82 RuntimeError: Parameters must be "int" or "float" objects
83 '''