
```
'''
percipio55_Implement an Iterable Using __getItem_.py
Percipio video: Iterables-and-Generators; Implement an Iterable Using __getItem_

* Demonstrate implementing iteration using the special method __getItem__ within a class.
* __getItem__ allows retrieving by index or slice if regular iteration over an object is not desired
'''
nl = '\n'
import logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s') # logs to the terminal
logging.debug('Start of program')

from random import randint # generates random integers

class LuckyBall(): # class will generate 6 different random numbers from the random module
    ''' Provide an iterable with 6 random numbers '''

    def __init__(self): # new instance is created that will generate a list of 6 unique integers
        ''' Generate a list of 6 unique random integers '''
        self.nums = [] # empty list created with the attribute self.nums
        while len(self.nums) < 6: # while the length of 'self.nums' is less than 6,...
            rand_num = randint(1,69) # ...it will generate a random integer from 1 to 69 including both endpoints and assign this to rand_num
            if not rand_num in self.nums: # if rand_num is not in the list of self.nums, then it's unique,...
                self.nums.append(rand_num) # ...then self.nums will have that rand number appended to it.
    ''' Once 6 random, unique numbers are generated, the while loop terminates and a new instance of the class LuckyBall is created '''

```





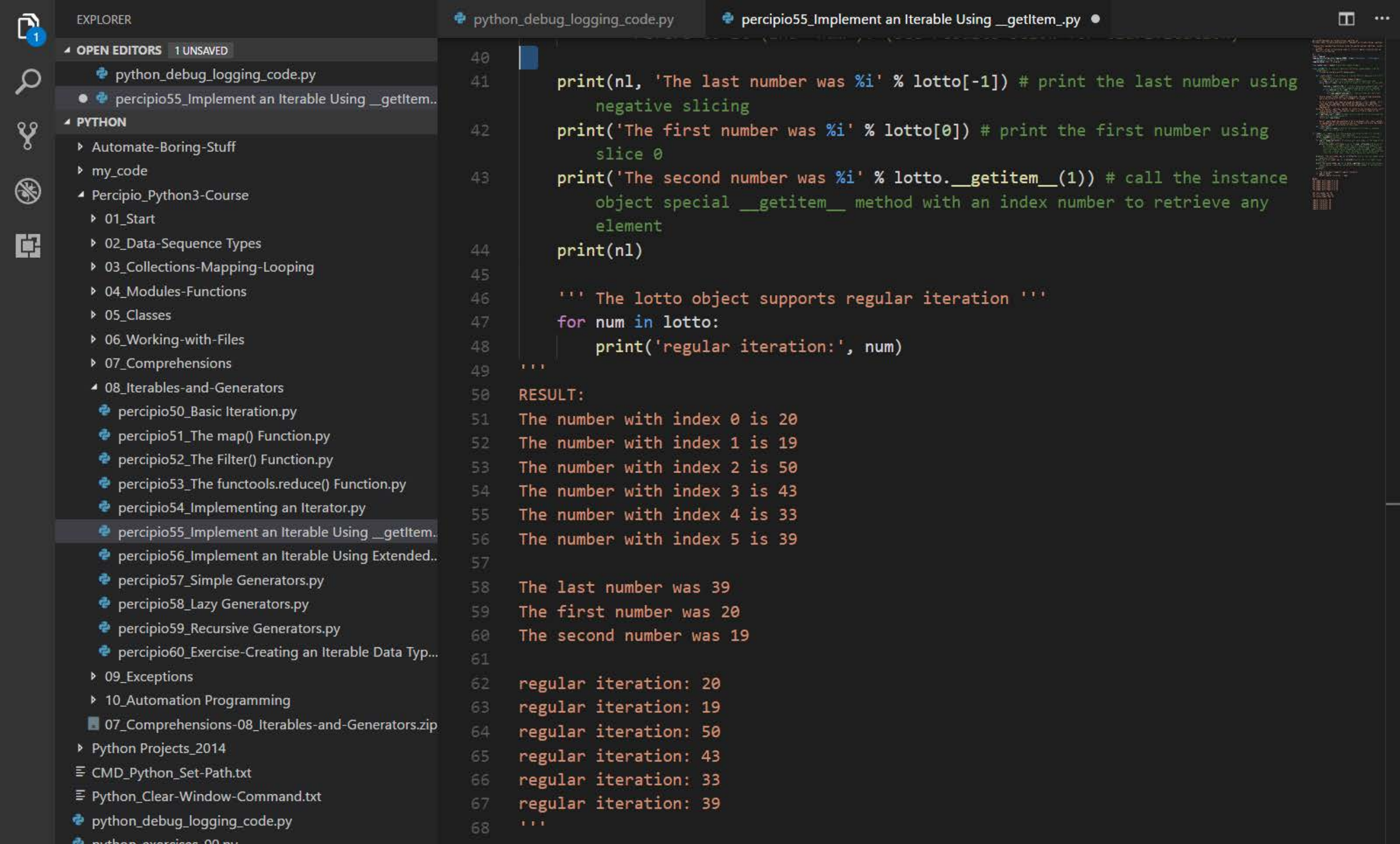
EXPLORER

OPEN EDITORS 1 UNSAVED
- python_debug_logging_code.py
- percipio55_Implement an Iterable Using __getItem...

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.py
    - percipio53_The functools.reduce() Function.py
    - percipio54_Implementing an Iterator.py
    - percipio55_Implement an Iterable Using __getItem...
    - percipio56_Implement an Iterable Using Extended...
    - percipio57_Simple Generators.py
    - percipio58_Lazy Generators.py
    - percipio59_Recursive Generators.py
    - percipio60_Exercise-Creating an Iterable Data Typ...
  - 09_Exceptions
  - 10_Automation Programming
  - 07_Comprehensions-08_Iterables-and-Generators.zip
- Python Projects_2014
- CMD_Python_Set-Path.txt
- Python_Clear-Window-Command.txt

python_debug_logging_code.py | percipio55_Implement an Iterable Using __getItem_.py ●

```
26
27      ''' So far itteration has required using the special __iter__ method.  If a
            class does not implement that method but implements the __getitem__
            method, then Python falls back to using the getitem if the iter is not
            implemented.
28      Using the special __getitem__ method, it refers to the instance itself (self),
             and a special index.  The special index is a slice or index of the
            element within the object. '''
29      def __getitem__(self, index): # index returns from the list of self.nums the
            number with the index passed in
30          return self.nums[index]
31
32      ''' Use of length might be something to do to an object, the __len__ method
            is implemented for the instance so there's the ability to use the length()
             function applied to instances of this class '''
33      def __len__(self):
34          return len(self.nums) # returns the length of self.nums, or how many
                numbers are in the list
35
36  if __name__ == '__main__': # if this program runs directly then it's running in
        the main namespace and hasn't been imported
37      lotto = LuckyBall() # when run directly, creates an instance of the class
            'LuckyBall' called 'lotto'
38      for num in range(len(lotto)): # iterates over each number in the range of the
            length of lotto
39          print('The number with index %s is %i' % (num, lotto[num])) # lotto, as
                an instance of LuckyBall class, has self.nums with a length of 6.
                This iterates over 6 different numbers from 0 through 5.  This code
                prints out these different numbers by using the number index itself
                (1st 'num') and by referring to the object ('lotto') & the slice that
                refers to it (2nd 'num'). (See results below for clarification)
40
```


EXPLORER
python_debug_logging_code.py
percipio55_Implement an Iterable Using __getItem_.py
OPEN EDITORS 1 UNSAVED
python_debug_logging_code.py
percipio55_Implement an Iterable Using __getItem..
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.py
percipio53_The functools.reduce() Function.py
percipio54_Implementing an Iterator.py
percipio55_Implement an Iterable Using __getItem..
percipio56_Implement an Iterable Using Extended..
percipio57_Simple Generators.py
percipio58_Lazy Generators.py
percipio59_Recursive Generators.py
percipio60_Exercise-Creating an Iterable Data Typ...
09_Exceptions
10_Automation Programming
07_Comprehensions-08_Iterables-and-Generators.zip
Python Projects_2014
CMD_Python_Set-Path.txt
Python_Clear-Window-Command.txt
python_debug_logging_code.py
print(nl, 'The last number was %i' % lotto[-1]) # print the last number using negative slicing
print('The first number was %i' % lotto[0]) # print the first number using slice 0
print('The second number was %i' % lotto.__getitem__(1)) # call the instance object special __getitem__ method with an index number to retrieve any element
print(nl)
''' The lotto object supports regular iteration '''
for num in lotto:
print('regular iteration:', num)
'''
RESULT:
The number with index 0 is 20
The number with index 1 is 19
The number with index 2 is 50
The number with index 3 is 43
The number with index 4 is 33
The number with index 5 is 39
The last number was 39
The first number was 20
The second number was 19
regular iteration: 20
regular iteration: 19
regular iteration: 50
regular iteration: 43
regular iteration: 33
regular iteration: 39
'''