



EXPLORER

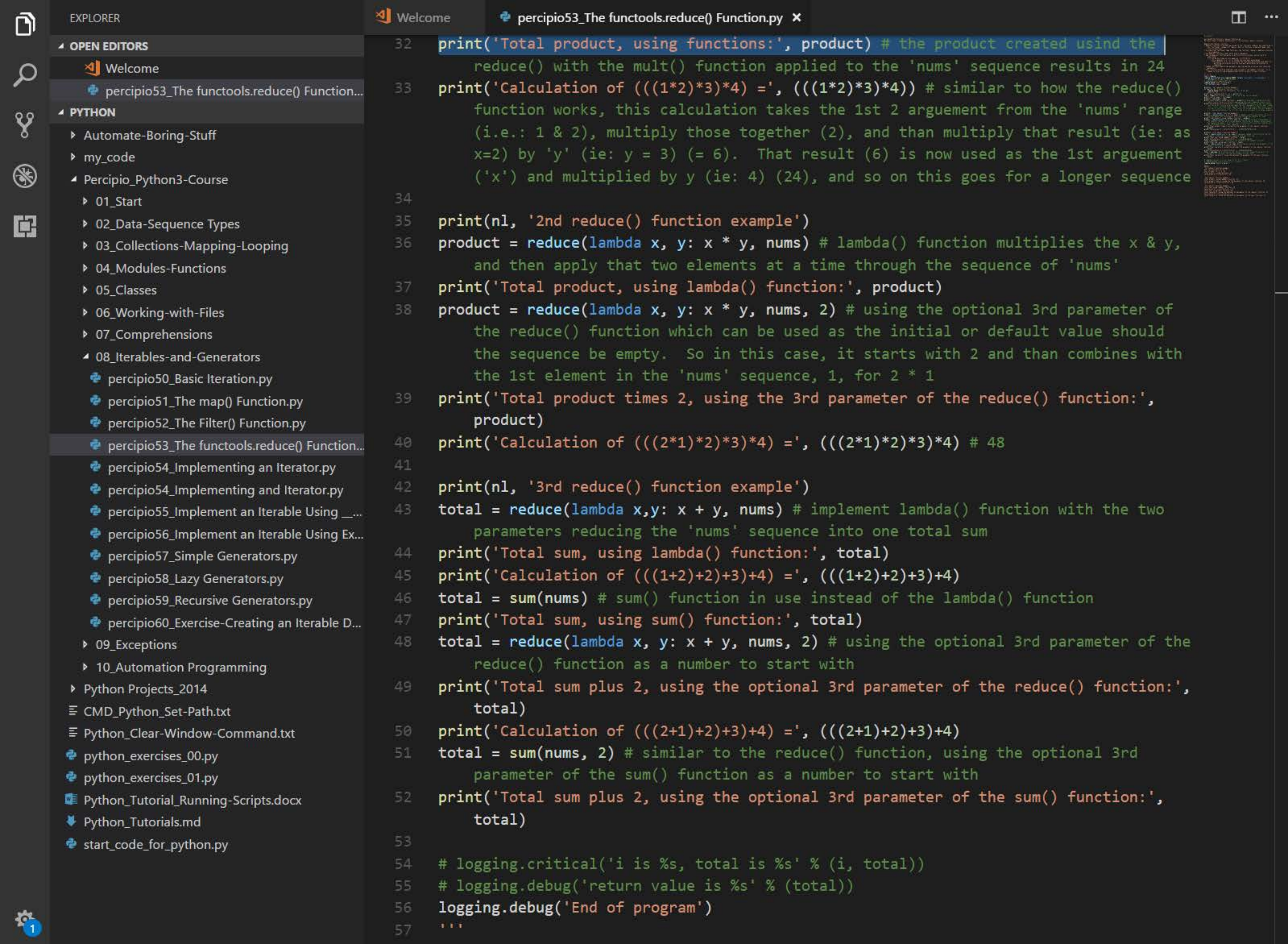
- OPEN EDITORS
  - Welcome
  - percipio53_The functools.reduce() Function...
- PYTHON
  - Automate-Boring-Stuff
  - my_code
  - Percipio_Python3-Course
    - 01_Start
    - 02_Data-Sequence Types
    - 03_Collections-Mapping-Looping
    - 04_Modules-Functions
    - 05_Classes
    - 06_Working-with-Files
    - 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...
      - percipio54_Implementing an Iterator.py
      - percipio54_Implementing and Iterator.py
      - percipio55_Implement an Iterable Using __...
      - percipio56_Implement an Iterable Using Ex...
      - percipio57_Simple Generators.py
      - percipio58_Lazy Generators.py
      - percipio59_Recursive Generators.py
      - percipio60_Exercise-Creating an Iterable D...
    - 09_Exceptions
    - 10_Automation Programming
  - Python Projects_2014
  - CMD_Python_Set-Path.txt
  - Python_Clear-Window-Command.txt
  - python_exercises_00.py
  - python_exercises_01.py
  - Python_Tutorial_Running-Scripts.docx
  - Python_Tutorials.md
  - start_code_for_python.py

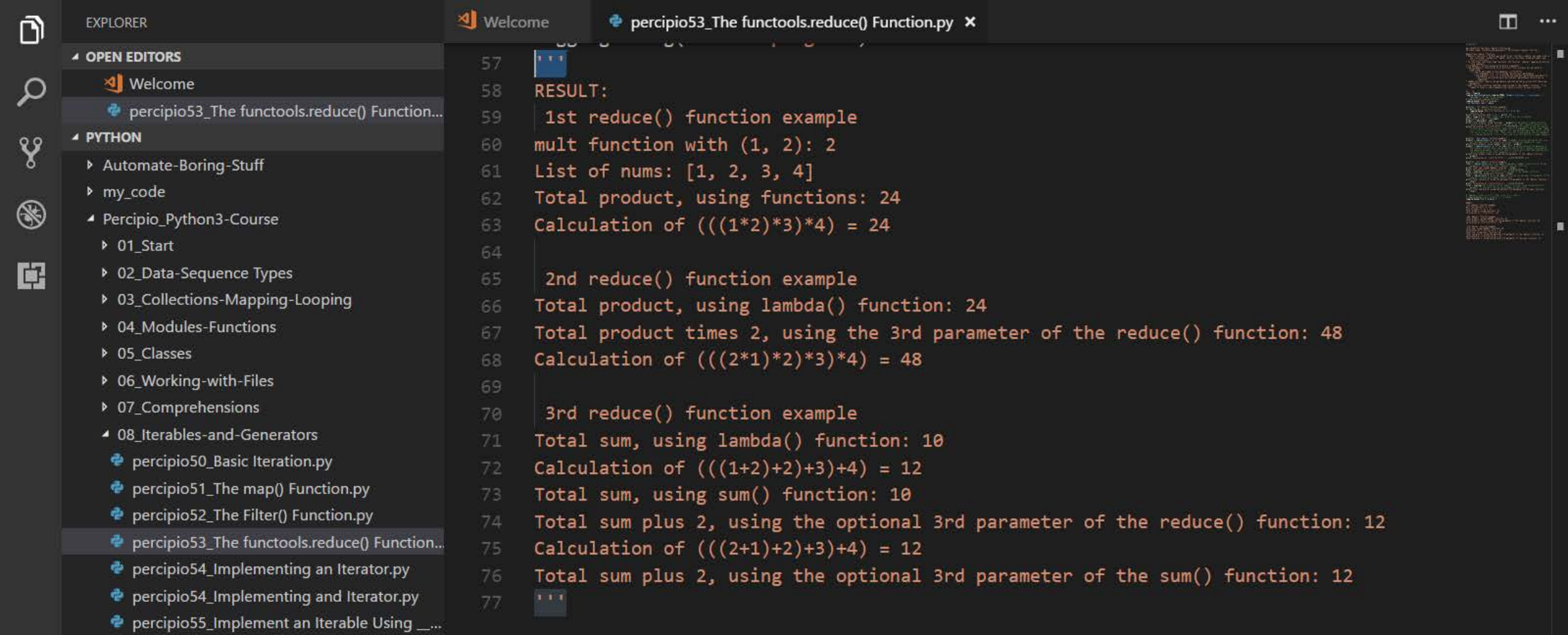
Welcome | percipio53_The functools.reduce() Function.py ×

```
#! python3
'''
percipio53_The functools.reduce() Function.py
Percipio video: Iterables-and-Generators; The functools.reduce() function

Demonstrate reduce() function
* Reduce() function in Python3 was moved to the 'functools' module, but used to be in
    the 'functional' category of regular, buit-in functions, along with map() and
    filter() functions.
* So like other functional-type functions, the function, reduce(), applies a function
     to a sequence.
* The Reduce() function always works with 2 arguements
* How the Reduce() function works is difficult to articulate, but not hard to
    understand.
    * It can be described in this equation, (((1*2)*3)*4)
        * 1st arguement (i.e.: x) is inside the innermost parentheses
        * 2nd arguement (i.e.: y) is outside the innermost parentheses and about to
            perform a calculation once the innermost parentheses calculation is
            completed
* Lambda functions require two parameters and then perfom an action with those two
    parameters.
* Sum() function should be sometimes used instead of thw lambda() function.  It is
    common to total or sum a sequence that there's a built-in sum() function
'''
nl = '\n'
import logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %
    (message)s') # logs to the terminal
# logging.disabl(logging.CRITICAL)
logging.debug('Start of program')
from functools import reduce

print(nl, '1st reduce() function example')
def mult(x, y): # define a function, 'mult'
    logging.debug('Start of function(%s, %s)' % (x, y))
    return x * y
print('mult function with (1, 2):', mult(1, 2))
nums = range(1,5) # number range from 1 & up to but not including 5
print('List of nums:', list(nums))
product = reduce(mult, nums)
print('Total product, using functions:', product) # the product created usind the
```


EXPLORER
OPEN EDITORS
Welcome
percipio53_The functools.reduce() Function...
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...
percipio54_Implementing an Iterator.py
percipio55_Implement an Iterable Using __...
percipio56_Implement an Iterable Using Ex...
percipio57_Simple Generators.py
percipio58_Lazy Generators.py
percipio59_Recursive Generators.py
percipio60_Exercise-Creating an Iterable D...
09_Exceptions
10_Automation Programming
Python Projects_2014
CMD_Python_Set-Path.txt
Python_Clear-Window-Command.txt
python_exercises_00.py
python_exercises_01.py
Python_Tutorial_Running-Scripts.docx
Python_Tutorials.md
start_code_for_python.py
Welcome
percipio53_The functools.reduce() Function.py
print('Total product, using functions:', product) # the product created usind the reduce() with the mult() function applied to the 'nums' sequence results in 24
print('Calculation of (((1*2)*3)*4) =', (((1*2)*3)*4)) # similar to how the reduce() function works, this calculation takes the 1st 2 arguement from the 'nums' range (i.e.: 1 & 2), multiply those together (2), and than multiply that result (ie: as x=2) by 'y' (ie: y = 3) (= 6). That result (6) is now used as the 1st arguement ('x') and multiplied by y (ie: 4) (24), and so on this goes for a longer sequence
print(nl, '2nd reduce() function example')
product = reduce(lambda x, y: x * y, nums) # lambda() function multiplies the x & y, and then apply that two elements at a time through the sequence of 'nums'
print('Total product, using lambda() function:', product)
product = reduce(lambda x, y: x * y, nums, 2) # using the optional 3rd parameter of the reduce() function which can be used as the initial or default value should the sequence be empty. So in this case, it starts with 2 and than combines with the 1st element in the 'nums' sequence, 1, for 2 * 1
print('Total product times 2, using the 3rd parameter of the reduce() function:', product)
print('Calculation of (((2*1)*2)*3)*4) =', (((2*1)*2)*3)*4) # 48
print(nl, '3rd reduce() function example')
total = reduce(lambda x,y: x + y, nums) # implement lambda() function with the two parameters reducing the 'nums' sequence into one total sum
print('Total sum, using lambda() function:', total)
print('Calculation of (((1+2)+2)+3)+4) =', (((1+2)+2)+3)+4)
total = sum(nums) # sum() function in use instead of the lambda() function
print('Total sum, using sum() function:', total)
total = reduce(lambda x, y: x + y, nums, 2) # using the optional 3rd parameter of the reduce() function as a number to start with
print('Total sum plus 2, using the optional 3rd parameter of the reduce() function:', total)
print('Calculation of (((2+1)+2)+3)+4) =', (((2+1)+2)+3)+4)
total = sum(nums, 2) # similar to the reduce() function, using the optional 3rd parameter of the sum() function as a number to start with
print('Total sum plus 2, using the optional 3rd parameter of the sum() function:', total)
# logging.critical('i is %s, total is %s' % (i, total))
# logging.debug('return value is %s' % (total))
logging.debug('End of program')
'''


EXPLORER
OPEN EDITORS
Welcome
percipio53_The functools.reduce() Function...
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...
percipio54_Implementing an Iterator.py
percipio54_Implementing and Iterator.py
percipio55_Implement an Iterable Using __...
Welcome
percipio53_The functools.reduce() Function.py
57 '''
58 RESULT:
59 1st reduce() function example
60 mult function with (1, 2): 2
61 List of nums: [1, 2, 3, 4]
62 Total product, using functions: 24
63 Calculation of (((1*2)*3)*4) = 24
64
65 2nd reduce() function example
66 Total product, using lambda() function: 24
67 Total product times 2, using the 3rd parameter of the reduce() function: 48
68 Calculation of (((2*1)*2)*3)*4) = 48
69
70 3rd reduce() function example
71 Total sum, using lambda() function: 10
72 Calculation of (((1+2)+2)+3)+4) = 12
73 Total sum, using sum() function: 10
74 Total sum plus 2, using the optional 3rd parameter of the reduce() function: 12
75 Calculation of (((2+1)+2)+3)+4) = 12
76 Total sum plus 2, using the optional 3rd parameter of the sum() function: 12
77 '''