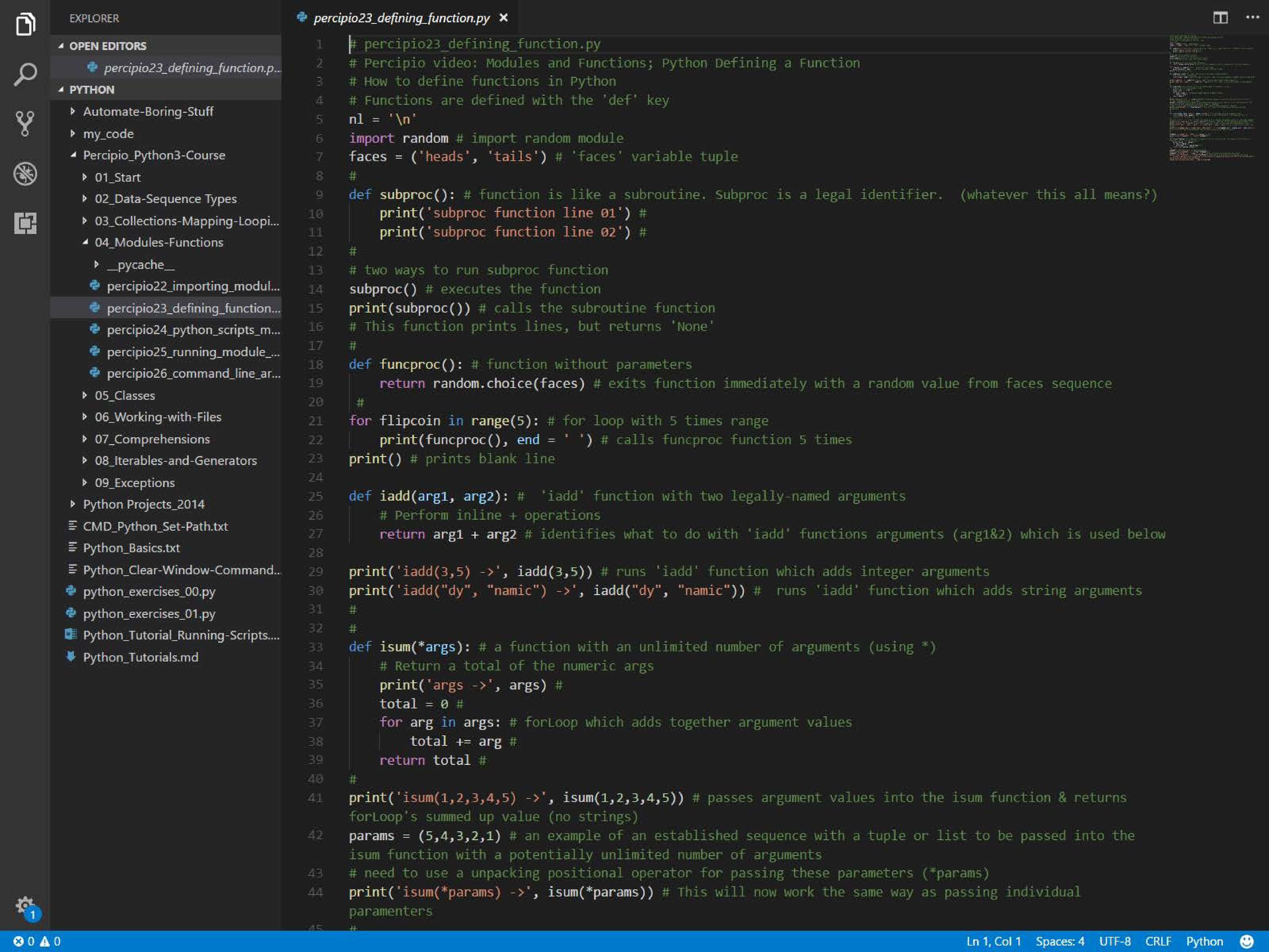
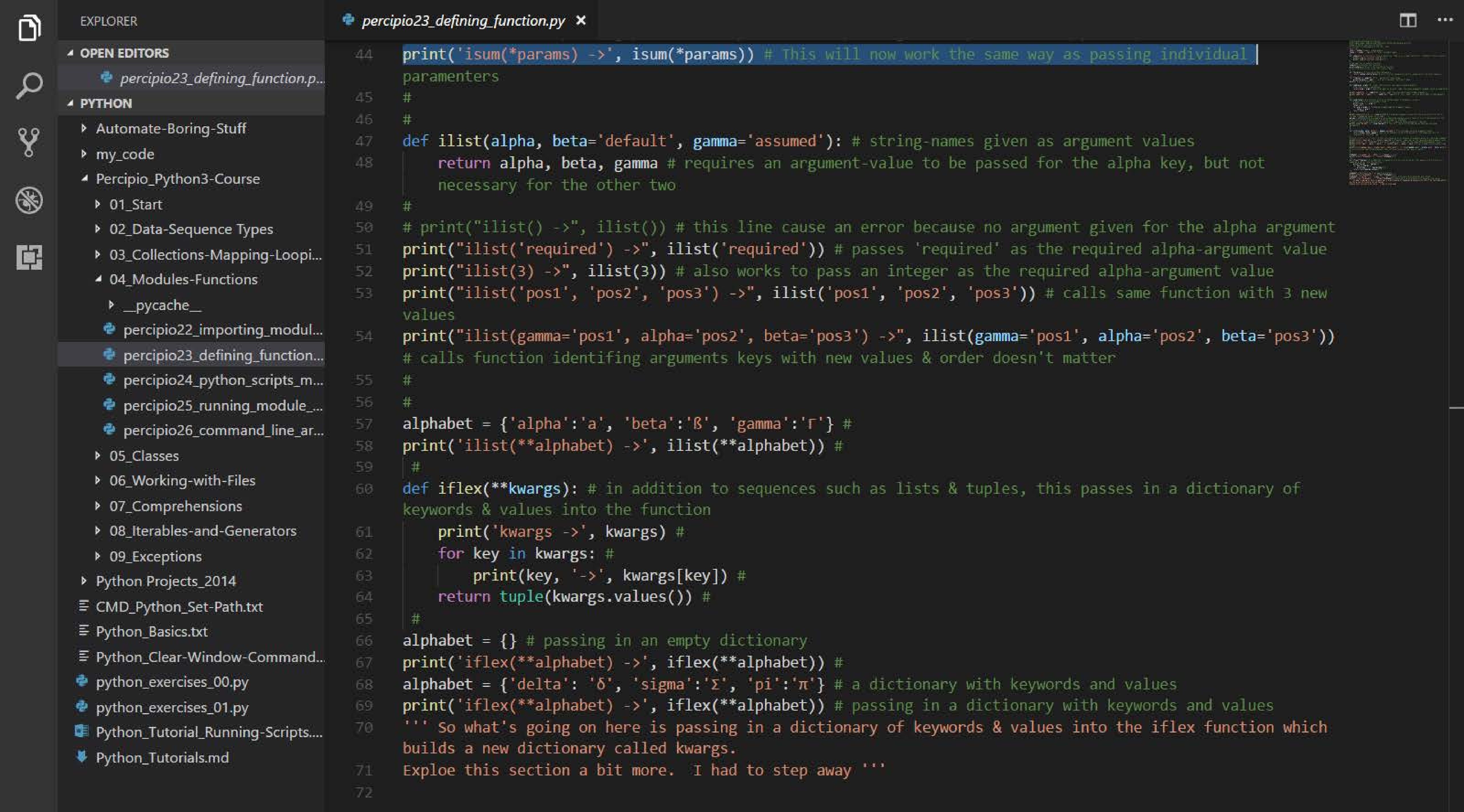


```python
# percipio23_defining_function.py
# Percipio video: Modules and Functions; Python Defining a Function
# How to define functions in Python
# Functions are defined with the 'def' key
nl = '\n'
import random # import random module
faces = ('heads', 'tails') # 'faces' variable tuple
#
def subproc(): # function is like a subroutine. Subproc is a legal identifier.  (whatever this all means?)
    print('subproc function line 01') #
    print('subproc function line 02') #
#
# two ways to run subproc function
subproc() # executes the function
print(subproc()) # calls the subroutine function
# This function prints lines, but returns 'None'
#
def funcproc(): # function without parameters
    return random.choice(faces) # exits function immediately with a random value from faces sequence
 #
for flipcoin in range(5): # for loop with 5 times range
    print(funcproc(), end = ' ') # calls funcproc function 5 times
print() # prints blank line

def iadd(arg1, arg2): #  'iadd' function with two legally-named arguments
    # Perform inline + operations
    return arg1 + arg2 # identifies what to do with 'iadd' functions arguments (arg1&2) which is used below

print('iadd(3,5) ->', iadd(3,5)) # runs 'iadd' function which adds integer arguments
print('iadd("dy", "namic") ->', iadd("dy", "namic")) #  runs 'iadd' function which adds string arguments
#
#
def isum(*args): # a function with an unlimited number of arguments (using *)
    # Return a total of the numeric args
    print('args ->', args) #
    total = 0 #
    for arg in args: # forLoop which adds together argument values
        total += arg #
    return total #
#
print('isum(1,2,3,4,5) ->', isum(1,2,3,4,5)) # passes argument values into the isum function & returns
forLoop's summed up value (no strings)
params = (5,4,3,2,1) # an example of an established sequence with a tuple or list to be passed into the
isum function with a potentially unlimited number of arguments
# need to use a unpacking positional operator for passing these parameters (*params)
print('isum(*params) ->', isum(*params)) # This will now work the same way as passing individual
paramenters
#
```


EXPLORER
OPEN EDITORS
percipio23_defining_function.p...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
03_Collections-Mapping-Loopi...
04_Modules-Functions
__pycache__
percipio22_importing_modul...
percipio23_defining_function...
percipio24_python_scripts_m...
percipio25_running_module_...
percipio26_command_line_ar...
05_Classes
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...
python_exercises_00.py
python_exercises_01.py
Python_Tutorial_Running-Scripts....
Python_Tutorials.md
percipio23_defining_function.py

```python
print('isum(*params) ->', isum(*params)) # This will now work the same way as passing individual paramenters
#
#
def ilist(alpha, beta='default', gamma='assumed'): # string-names given as argument values
    return alpha, beta, gamma # requires an argument-value to be passed for the alpha key, but not necessary for the other two
#
# print("ilist() ->", ilist()) # this line cause an error because no argument given for the alpha argument
print("ilist('required') ->", ilist('required')) # passes 'required' as the required alpha-argument value
print("ilist(3) ->", ilist(3)) # also works to pass an integer as the required alpha-argument value
print("ilist('pos1', 'pos2', 'pos3') ->", ilist('pos1', 'pos2', 'pos3')) # calls same function with 3 new values
print("ilist(gamma='pos1', alpha='pos2', beta='pos3') ->", ilist(gamma='pos1', alpha='pos2', beta='pos3')) # calls function identifing arguments keys with new values & order doesn't matter
#
#
alphabet = {'alpha':'a', 'beta':'ß', 'gamma':'Γ'} #
print('ilist(**alphabet) ->', ilist(**alphabet)) #
 #
def iflex(**kwargs): # in addition to sequences such as lists & tuples, this passes in a dictionary of keywords & values into the function
    print('kwargs ->', kwargs) #
    for key in kwargs: #
        print(key, '->', kwargs[key]) #
    return tuple(kwargs.values()) #
 #
alphabet = {} # passing in an empty dictionary
print('iflex(**alphabet) ->', iflex(**alphabet)) #
alphabet = {'delta': 'δ', 'sigma':'Σ', 'pi':'π'} # a dictionary with keywords and values
print('iflex(**alphabet) ->', iflex(**alphabet)) # passing in a dictionary with keywords and values
''' So what's going on here is passing in a dictionary of keywords & values into the iflex function which builds a new dictionary called kwargs.
Exploe this section a bit more.  I had to step away '''

```