```
"""
functions_lab.py
@author: ITP 150 Student
This script covers functions with positional arguments, keyword arguments,
default parameters, arbitrary arguments *args, argument tuple packing and
unpacking, **kwargs.
"""

import random

print("\nTask 1. Import a function")
integer = random.randint(0, 100)
print("A random int between 0 and 100 is ", integer)

print("\nTask 2. See the help.")
print("If you want to see a function's documentation, enter ")
print("help followed by the function name.")
# help(random)  # Please be sure to comment this out after running it the first
time.

print("\nTask 3. Basic function without parameters and arguments.")
def my_function():
    location = "We are inside of my function."
    print(location)

print("Before calling my_function.")
my_function()
print("After calling my function")

print("\nTask 4. Basic function that has a docstring and the keyword pass")
def my_empty_function():
    """ This function is empty but has this line of documentation."""
    pass  # You can put in the keyword pass for an incomplete function

my_empty_function()
help(my_empty_function)

print("\nTask 5 Positional Arguments require same qty of arguments and \
parameters in the same order representing the same data types.")
def water(qty, h2o, price):
    print(f'{qty} {h2o} cost ${price:.2f}')
print('Call 1:')
water(1, "Aquafina", 2.69)
print('Call 2 throws an error bc we passed only 2 arguments.')
# water(1, "Dasani")  # Please comment out after running the first time.
'print(Call 3 does not throw error but is logically incorrect)'
water("Nestle Pure Water", 2.69, 1)  # Logic error qty of 1 instead of price


print("\nTask 6. Keyword Arguments allow flexibility")
def water_keywords(qty, h2o, price):
    print(f'{qty} {h2o} cost ${price:.2f}')

print("\nCall 1. You must supply parameters")
# water_keywords()  # Please comment after running the first time.
print("\nCall 2. Keyword Arguments in the same order as the function def.")
water_keywords(qty=2, h2o="Aquafina", price=2.69)
print("\nCall 3. Keyword arguments NOT in the order of the function def.")
```

```
water_keywords(h2o="Aquafina", price=2.69, qty=2)
print("Takeaway. Using keyword arguments allows us to pass in different order")
print("\nCall 4. Can we leave out an argument using keyword arguments? NO")
# water_keywords(h2o = "Aquafina", price = 2.69)  # Comment after running 1st time.
print("\nCall 5. Try a mix of positional and keyword arguments")
water_keywords(2, h2o="Aquafina", price=2.69)
print('positional arguments must come before keyword args')
print("\nCall 6. keyword args can not follow positional args")
# water_keywords(qty=2, "Aquafina", price=2.69)  # comment after running 1st time
print("\nCall 7. Can you change the keyword argument name? NO")
# water_keywords(qty = 2, brand = "Aquafina", price = 2.69)  # comment after running 1st time
# the brand parameter did not match the keyword argument h2o so we got an error

print("\nTask 7. Default Parameters allow us to set default values for")
print("a parameter in the function definition.")
def water_default(qty=1, h2o="Aquafina", price=2.69):
    print(f'{qty} {h2o} cost ${price:.2f}')

print("\nCall 1. Sending different arguments than the default")
water_default(2, "Dasani", 3.50)
print("Takeaway is that if you send different arguments up when")
print("calling a function with default parameters, your arguments")
print("will override the defaults which is a good thing.")
print("\nCall 2. Sending up 2 arguments to see the default come into play")
water_default(2, "Dasani")
print("Takeaway is that you no longer have to supply all 3 arguments")
print("if default parameters have been specified in the function def.")
print("\nCall 3. Experiment using position and keyword args.")
print("on a function with default parameters")
# water_default(2, h2o = "Dasani", 3.50) # comment after running first time.
print("Takeaway is that you can use positional and keyword arguments")
print("With a function that has default parameters but you still must")
print("follow the rule that positional arguments must come before ")
print("keyword arguments.")

print("\nTask 8. *args (Argument Tuple Packing")
print("You can use argument tuple packing when you don't know how")
print("many arguments you need.")
def args_function(*args):
    print(args)
    print(type(args), len(args))
    for x in args:
        print(x)

args_function(2, 4, 6, 8, 10)

print("\nTask 9. Practical Example of *args")
def avg_args(*args):
    sum_args = 0
    for i in args:
        sum_args = sum_args + i
    return sum_args / len(args)

print(avg_args(2, 4, 6, 8, 10))
print(avg_args(1, 3, 5))

```

```
print("\nTask 10. Argument Tuple Unpacking")
print("This works with tuples, lists, and sets. Example is tuple.")
def unpack_args(a, b, c):
    print("a is ", a)
    print("b is ", b)
    print("c is ", c)


stuff = (1, "Perrier", 5.00)
print(type(stuff))
unpack_args(*stuff)

print("\nTask 11. Argument Tuple Packing and Unpacking.")
print("This works with tuples, lists, and sets. Example is a tuple.")
def packing(*args):
    print(type(args), args)

stuff = (1, "Perrier", 5.00)
packing(*stuff)

print('\nTask 12. Arbitrary Keyword Arguments, **kwargs"')
print('If you do not know how many keyword arguments that will be passed into \
your function, add two asterisk: ** before the parameter name in the \
function definition.')
def pets_function(**pets):
  print("My dog's name is " + pets["dog"])
  print("My cat's name is " + pets["cat"])

pets_function(cat = "Miss Kitty", dog = "Buddy")

```