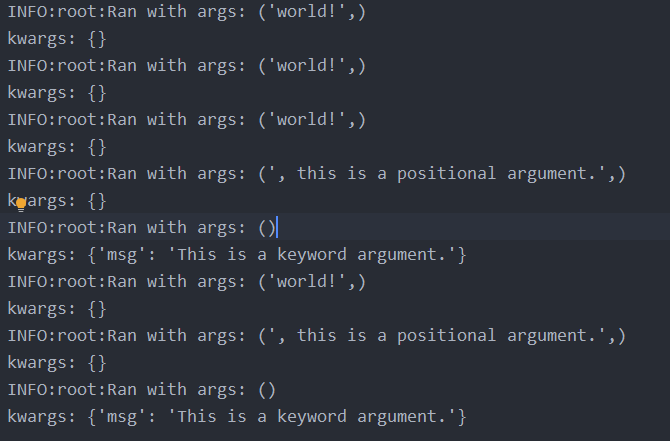
#Information about decorators#

Decorators in python are a bit trippy but if you pay attention they will make sense. Here below we are describing some basic information about them and showing you some real world examples where they could be implemented.The information from this document was taken from Corey Schafer’s Youtube tutorial on decorators (<https://www.youtube.com/watch?v=FsAPt_9Bf3U&list=PL-osiE80TeTt2d9bfVyTiXJA-UTHn6WwU&index=37>)

def decorator\_function(original\_function): # We pass in original\_function as a arg  
 def wrapper\_function(): # We define the inner function also called Wrapper function  
 print('This was printed before') # Everything here gets executed before our original\_function  
 return original\_function() # After we execute what we wanted we then execute our original\_function  
 return wrapper\_function # We return wrapper\_function in an "unsolicited way"  
  
  
@decorator\_function # This is basically saying new\_function = decorator\_function(the\_func\_we\_want\_to\_be\_executed)  
def hello\_world():  
 print("Hello World")  
  
  
hello\_world() # In the end we simply call our hello\_world function which is basically new\_func = decorator\_function(hello\_world)  
  
### Some real-world examples ###  
  
def logger\_function(function\_to\_be\_logged):  
 import logging  
 logging.basicConfig(filename=f'{function\_to\_be\_logged.\_\_name\_\_}.log', level=logging.INFO) # .\_\_name\_\_ gives us the name of the function  
  
 def wrapper(\*args, \*\*kwargs): # args = positional arguments , kwargs = keyword arguments (example below)  
 logging.info(  
 f'Ran with args: {args}\nkwargs: {kwargs}'  
 )  
 return function\_to\_be\_logged(\*args, \*\*kwargs)  
  
 return wrapper  
  
@logger\_function  
def hello\_world(msg):  
 print(f'Hello {msg}')  
  
hello\_world('world!')  
  
# Example of \*arg - positional argument  
hello\_world(', this is a positional argument.')  
# Example of \*\*kwarg - keyword argument  
hello\_world(msg='This is a keyword argument.')

Example output of our .log file :



def my\_timer(orig\_func):  
 import time # Import time module  
 def wrapper(\*args, \*\*kwargs):  
 """We make a variable called to\_be\_timed otherwise we have to run the function  
 and that would print hello + {msg} again."""  
 t1 = time.time() # Get's T1  
 to\_be\_timed = orig\_func(\*args, \*\*kwargs)   
 t2 = time.time() - t1 # t2 - t1 gives us the actual runtime of our function  
 print(f'{orig\_func.\_\_name\_\_} ran in : {t2} sec')  
 return to\_be\_timed # We return our newly made variable  
 return wrapper  
  
@my\_timer  
def hello\_world(msg):  
 print(f'Hello {msg}!')  
  
hello\_world('world')