Chat to us!

Linters for Python Code Quality. Remember to save your work to your GitHub repository.

Copy the following Python code into a file named code_with_lint.py:

```
import io
from math import *


from time import time

some_global_var = 'GLOBAL VAR NAMES SHOULD BE IN ALL_CAPS_WITH_UNDERSCOES'

def multiply(x, y):
    """
    This returns the result of a multiplation of the inputs
    """
    some_global_var = 'this is actually a local variable...'
    result = x* y
    return result
    if result == 777:
        print("jackpot!")

def is_sum_lucky(x, y):
    """This returns a string describing whether or not the sum of input is lucky
    This function first makes sure the inputs are valid and then calculates the
    sum. Then, it will determine a message to return based on whether or not
    that sum should be considered "lucky"
    """
    if x != None:
        if y is not None:
            result = x+y;
            if result == 7:
                return 'a lucky number!'
            else:
                return( 'an unlucky number!')

            return ('just a normal number')

class SomeClass:

    def __init__(self, some_arg,  some_other_arg, verbose = False):
        self.some_other_arg  =  some_other_arg
        self.some_arg        =  some_arg
        list_comprehension = [((100/value)*pi) for value in some_arg if value != 0]
        time = time()
        from datetime import datetime
        date_and_time = datetime.now()
        return
```

Code source: https://realpython.com/python-code-quality/

Now run the code against a variety of linters to test the code quality:

- pylint code_with_lint.py
- pyflakes code_with_lint.py
- pycodestyle code_with_lint.py
- pydocstyle code_with_lint.py

Compare the effectiveness of each tool in defining and identifying code quality. What can you conclude about the effectiveness of each approach?

Chat to us!

© 2025 University of Essex Online. All rights reserved.

Chat to us!