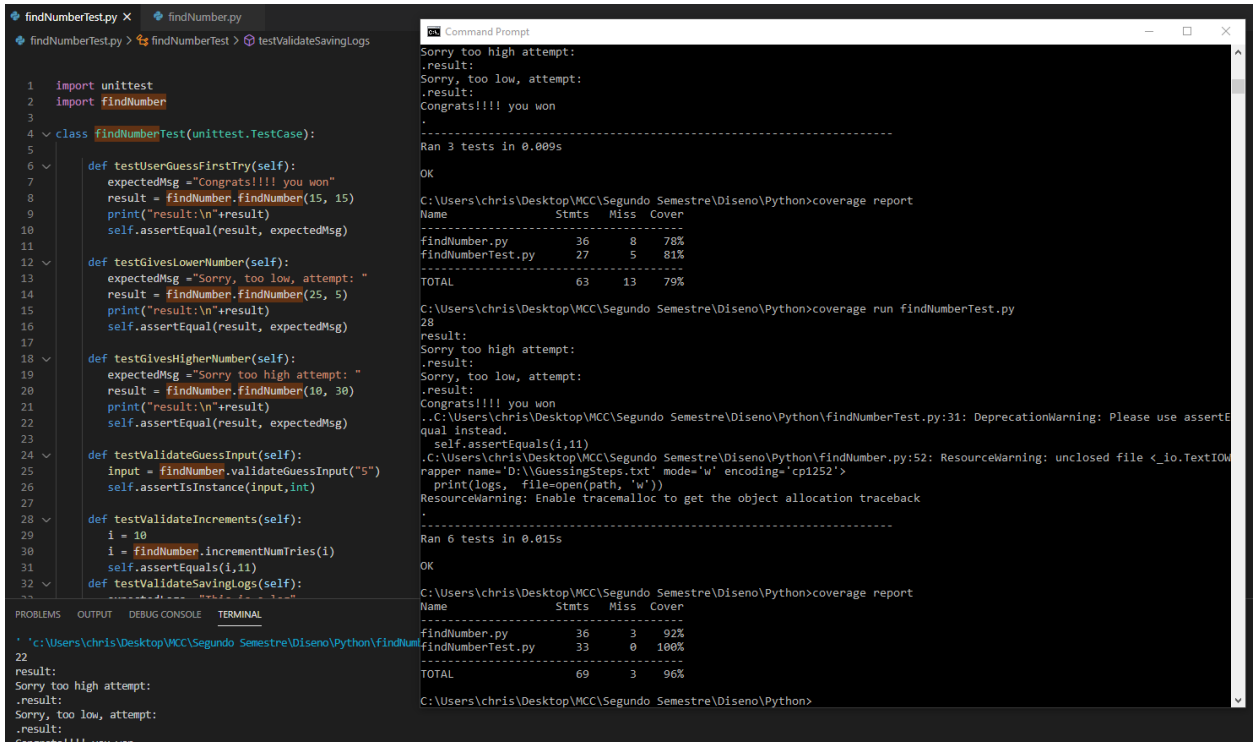
A00354891 – Christian Medina

1. FIND THE NUMBER

Evidence of code coverage and achieving code coverage. (96%)



findNumberTest.py

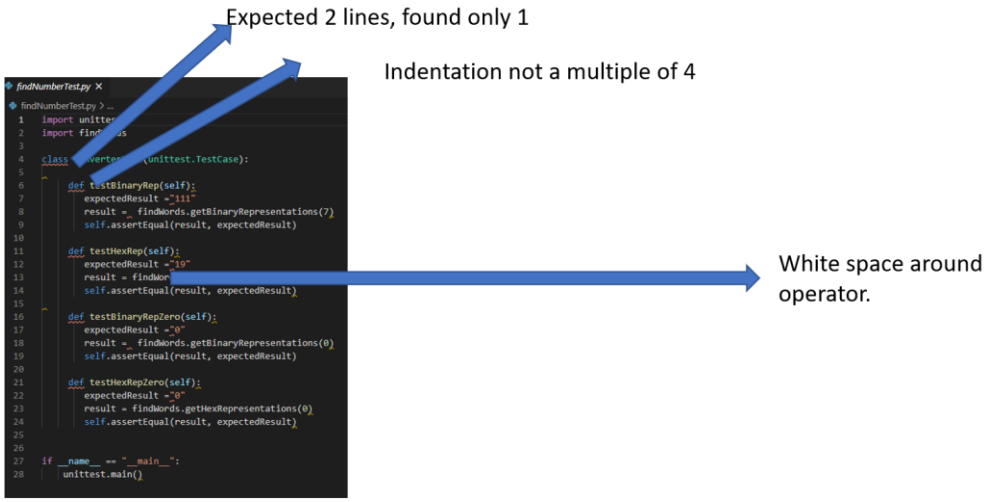
```
import unittest
import findNumber

class findNumberTest(unittest.TestCase):

    def testUserGuessFirstTry(self):
        expectedMsg ="Congrats!!!! you won"
        result = findNumber.findNumber(15, 15)
        print("result:\n"+result)
        self.assertEqual(result, expectedMsg)

    def testGivesLowerNumber(self):
        expectedMsg ="Sorry, too low, attempt: "
        result = findNumber.findNumber(25, 5)
        print("result:\n"+result)
        self.assertEqual(result, expectedMsg)

    def testGivesHigherNumber(self):
        expectedMsg ="Sorry too high attempt: "
        result = findNumber.findNumber(10, 30)
        print("result:\n"+result)
        self.assertEqual(result, expectedMsg)

    def testValidateGuessInput(self):
        input = findNumber.validateGuessInput("5")
        self.assertIsInstance(input,int)

    def testValidateIncrements(self):
        i = 10
        i = findNumber.incrementNumTries(i)
        self.assertEquals(i,11)
    def testValidateSavingLogs(self):
```

```
' 'c:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\findNum
22
result:
Sorry too high attempt:
.result:
Sorry, too low, attempt:
.result:
```

Command Prompt

```
Sorry too high attempt:
.result:
Sorry, too low, attempt:
.result:
Congrats!!!! you won
.
----------------------------------------------------------------------
Ran 3 tests in 0.009s

OK

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage report
Name                Stmts   Miss  Cover
---------------------------------------
findNumber.py          36      8    78%
findNumberTest.py      27      5    81%
---------------------------------------
TOTAL                  63     13    79%

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage run findNumberTest.py
28
result:
Sorry too high attempt:
.result:
Sorry, too low, attempt:
.result:
Congrats!!!! you won
..C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\findNumberTest.py:31: DeprecationWarning: Please use assertE
qual instead.
  self.assertEquals(i,11)
.C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\findNumber.py:52: ResourceWarning: unclosed file <_io.TextIOW
rapper name='D:\\GuessingSteps.txt' mode='w' encoding='cp1252'>
  print(logs,  file=open(path, 'w'))
ResourceWarning: Enable tracemalloc to get the object allocation traceback
.
----------------------------------------------------------------------
Ran 6 tests in 0.015s

OK

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage report
Name                Stmts   Miss  Cover
---------------------------------------
findNumber.py          36      3    92%
findNumberTest.py      33      0   100%
---------------------------------------
TOTAL                  69      3    96%

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>
```

# Pep8

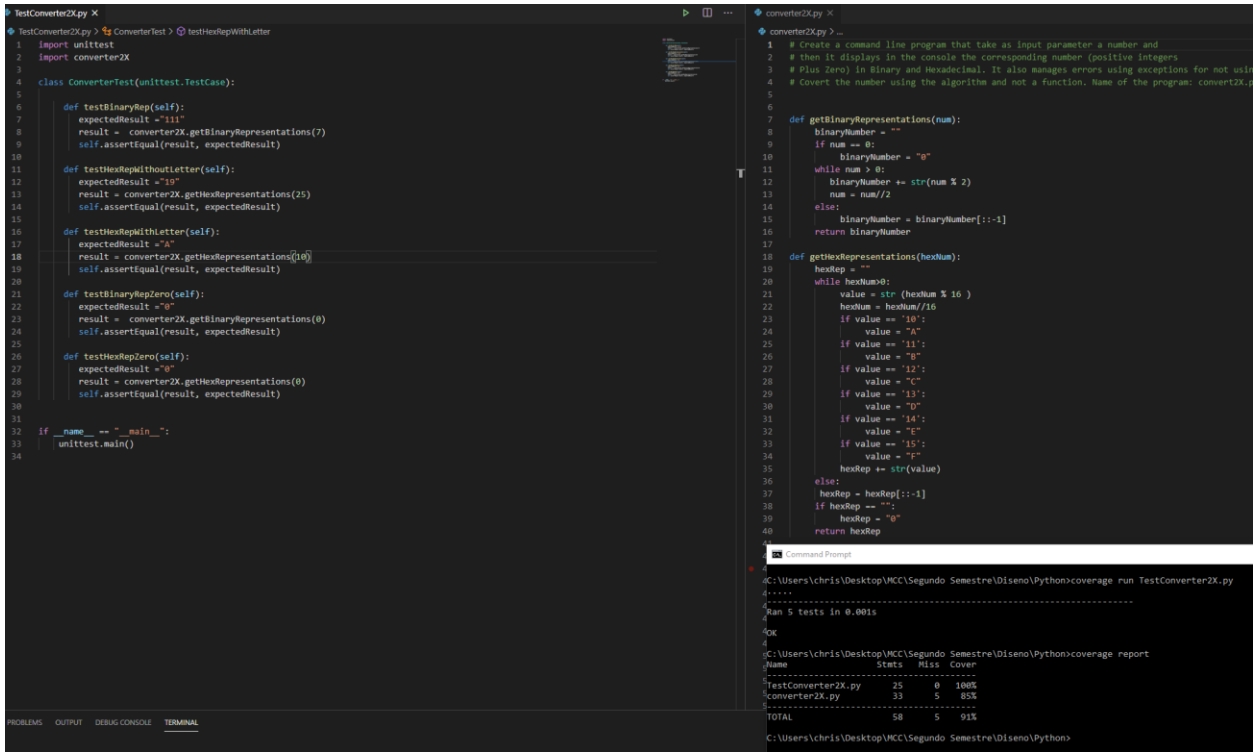



findNumberTest.py

```python
import unittest
import findWords


class ConverterTest(unittest.TestCase):
    def testBinaryRep(self):
        expectedResult = "111"
        result = findWords.getBinaryRepresentations(7)
        self.assertEqual(result, expectedResult)

    def testHexRep(self):
        expectedResult = "19"
        result = findWords.getHexRepresentations(25)
        self.assertEqual(result, expectedResult)

    def testBinaryRepZero(self):
        expectedResult = "0"
        result = findWords.getBinaryRepresentations(0)
        self.assertEqual(result, expectedResult)

    def testHexRepZero(self):
        expectedResult = "0"
        result = findWords.getHexRepresentations(0)
        self.assertEqual(result, expectedResult)


if __name__ == "__main__":
    unittest.main()
```

2. CONVERTER

Evidence of code coverage and achieving code coverage. (91%)

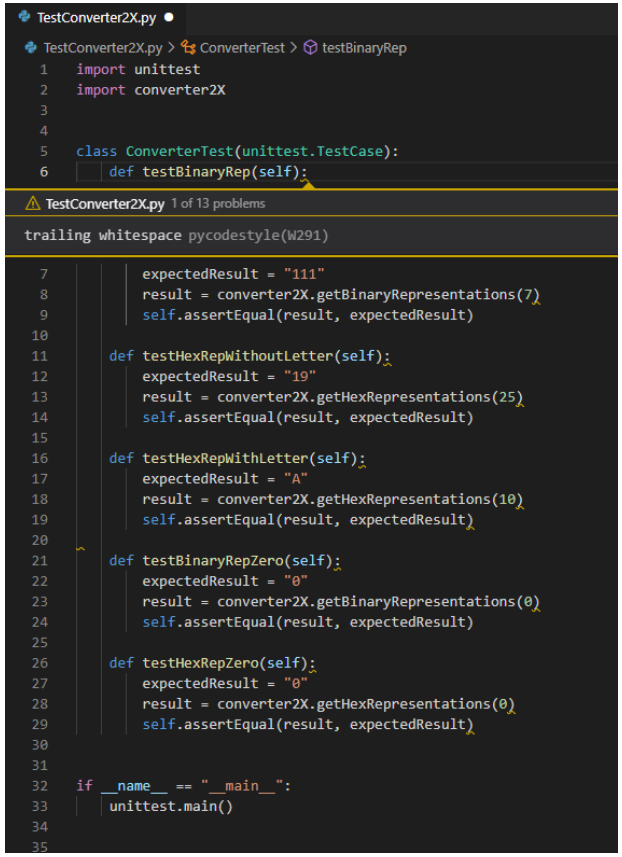


```
import unittest
import converter2X

class ConverterTest(unittest.TestCase):

    def testBinaryRep(self):
        expectedResult ="111"
        result =  converter2X.getBinaryRepresentations(7)
        self.assertEqual(result, expectedResult)

    def testHexRepWithoutLetter(self):
        expectedResult ="19"
        result = converter2X.getHexRepresentations(25)
        self.assertEqual(result, expectedResult)

    def testHexRepWithLetter(self):
        expectedResult ="A"
        result = converter2X.getHexRepresentations(10)
        self.assertEqual(result, expectedResult)

    def testBinaryRepZero(self):
        expectedResult ="0"
        result =  converter2X.getBinaryRepresentations(0)
        self.assertEqual(result, expectedResult)

    def testHexRepZero(self):
        expectedResult ="0"
        result = converter2X.getHexRepresentations(0)
        self.assertEqual(result, expectedResult)


if __name__ == "__main__":
    unittest.main()
```

```
# Create a command line program that take as input parameter a number and
# then it displays in the console the corresponding number (positive integers
# Plus Zero) in Binary and Hexadecimal. It also manages errors using exceptions for not usin
# Covert the number using the algorithm and not a function. Name of the program: convert2X.p


def getBinaryRepresentations(num):
    binaryNumber = ""
    if num == 0:
        binaryNumber = "0"
    while num > 0:
        binaryNumber += str(num % 2)
        num = num//2
    else:
        binaryNumber = binaryNumber[::-1]
    return binaryNumber

def getHexRepresentations(hexNum):
    hexRep = ""
    while hexNum>0:
        value = str (hexNum % 16 )
        hexNum = hexNum//16
        if value == '10':
            value = "A"
        if value == '11':
            value = "B"
        if value == '12':
            value = "C"
        if value == '13':
            value = "D"
        if value == '14':
            value = "E"
        if value == '15':
            value = "F"
        hexRep += str(value)
    else:
        hexRep = hexRep[::-1]
    if hexRep == "":
        hexRep = "0"
    return hexRep
```

```
C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage run TestConverter2X.py
....
----------------------------------------------------------------------
Ran 5 tests in 0.001s

OK

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage report
Name                 Stmts   Miss  Cover
----------------------------------------
TestConverter2X.py      25      0   100%
converter2X.py          33      5    85%
----------------------------------------
TOTAL                   58      5    91%

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>
```

Pep 8




Warning message because of trailing whitespaces. pycodestyle(W291).

There should be no whitespace after the final character in a line.

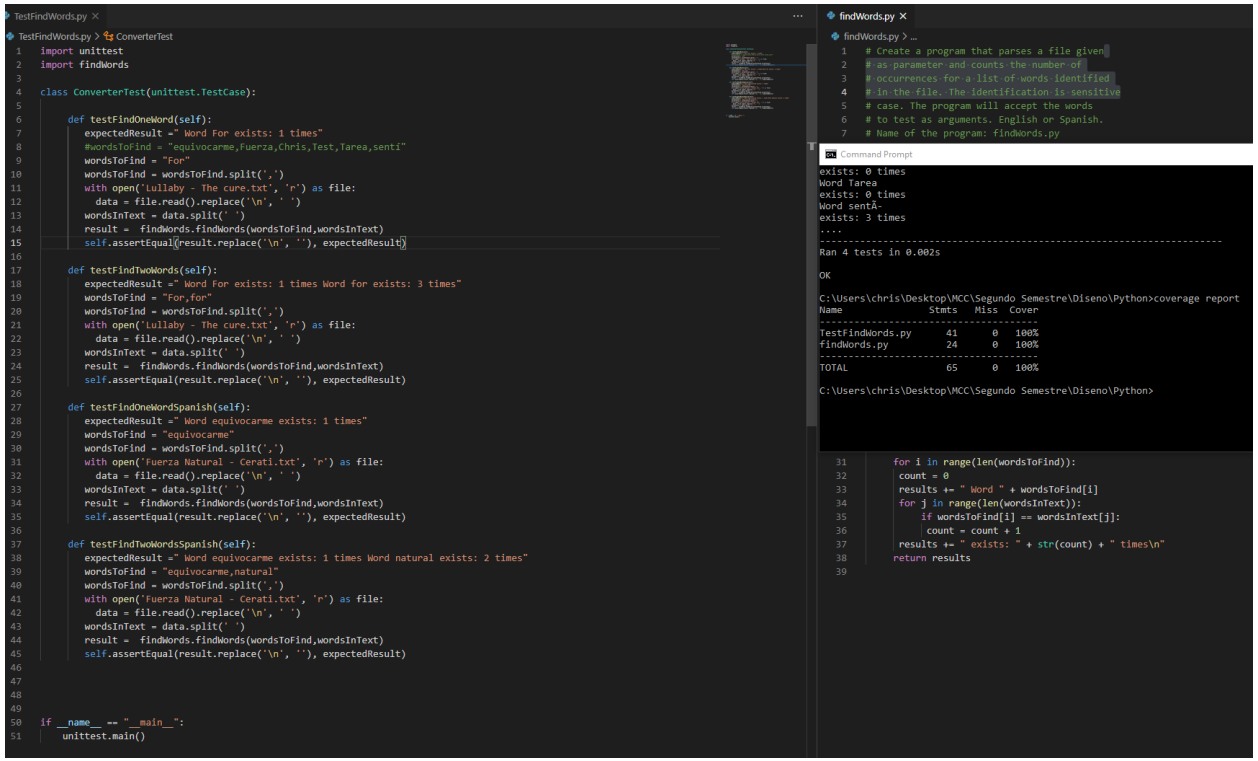
## Anti-pattern

**Note:** The • character represents a space.

```
def first_func():
    # The line below has two spaces after its final character
    pass••
```

3. Find Words.

Evidence of code coverage and achieving code coverage. (100%)



TestFindWords.py

```
import unittest
import findWords

class ConverterTest(unittest.TestCase):

    def testFindOneWord(self):
        expectedResult =" Word For exists: 1 times"
        #wordsToFind = "equivocarme,Fuerza,Chris,Test,Tarea,sentí"
        wordsToFind = "For"
        wordsToFind = wordsToFind.split(',')
        with open('Lullaby - The cure.txt', 'r') as file:
          data = file.read().replace('\n', ' ')
        wordsInText = data.split(' ')
        result =  findWords.findWords(wordsToFind,wordsInText)
        self.assertEqual(result.replace('\n', ''), expectedResult)

    def testFindTwoWords(self):
        expectedResult =" Word For exists: 1 times Word for exists: 3 times"
        wordsToFind = "For,for"
        wordsToFind = wordsToFind.split(',')
        with open('Lullaby - The cure.txt', 'r') as file:
          data = file.read().replace('\n', ' ')
        wordsInText = data.split(' ')
        result =  findWords.findWords(wordsToFind,wordsInText)
        self.assertEqual(result.replace('\n', ''), expectedResult)

    def testFindOneWordSpanish(self):
        expectedResult =" Word equivocarme exists: 1 times"
        wordsToFind = "equivocarme"
        wordsToFind = wordsToFind.split(',')
        with open('Fuerza Natural - Cerati.txt', 'r') as file:
          data = file.read().replace('\n', ' ')
        wordsInText = data.split(' ')
        result =  findWords.findWords(wordsToFind,wordsInText)
        self.assertEqual(result.replace('\n', ''), expectedResult)

    def testFindTwoWordsSpanish(self):
        expectedResult =" Word equivocarme exists: 1 times Word natural exists: 2 times"
        wordsToFind = "equivocarme,natural"
        wordsToFind = wordsToFind.split(',')
        with open('Fuerza Natural - Cerati.txt', 'r') as file:
          data = file.read().replace('\n', ' ')
        wordsInText = data.split(' ')
        result =  findWords.findWords(wordsToFind,wordsInText)
        self.assertEqual(result.replace('\n', ''), expectedResult)



if __name__ == "__main__":
    unittest.main()
```

findWords.py

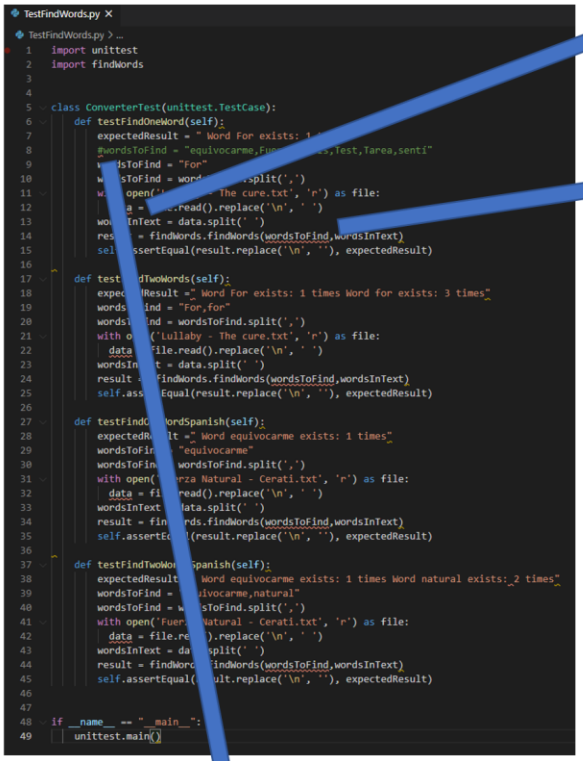
```
# Create a program that parses a file given
# as parameter and counts the number of
# occurrences for a list of words identified
# in the file. The identification is sensitive
# case. The program will accept the words
# to test as arguments. English or Spanish.
# Name of the program: findWords.py
```

Command Prompt

```
exists: 0 times
Word Tarea
exists: 0 times
Word sentÃexists: 3 times
....
----------------------------------------------------------------------
Ran 4 tests in 0.002s

OK

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage report
Name                 Stmts   Miss  Cover
----------------------------------------
TestFindWords.py        41      0   100%
findWords.py            24      0   100%
----------------------------------------
TOTAL                   65      0   100%

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>
```

```
    for i in range(len(wordsToFind)):
        count = 0
        results += " Word " + wordsToFind[i]
        for j in range(len(wordsInText)):
            if wordsToFind[i] == wordsInText[j]:
                count = count + 1
        results += " exists: " + str(count) + " times\n"
    return results

```

Pep8



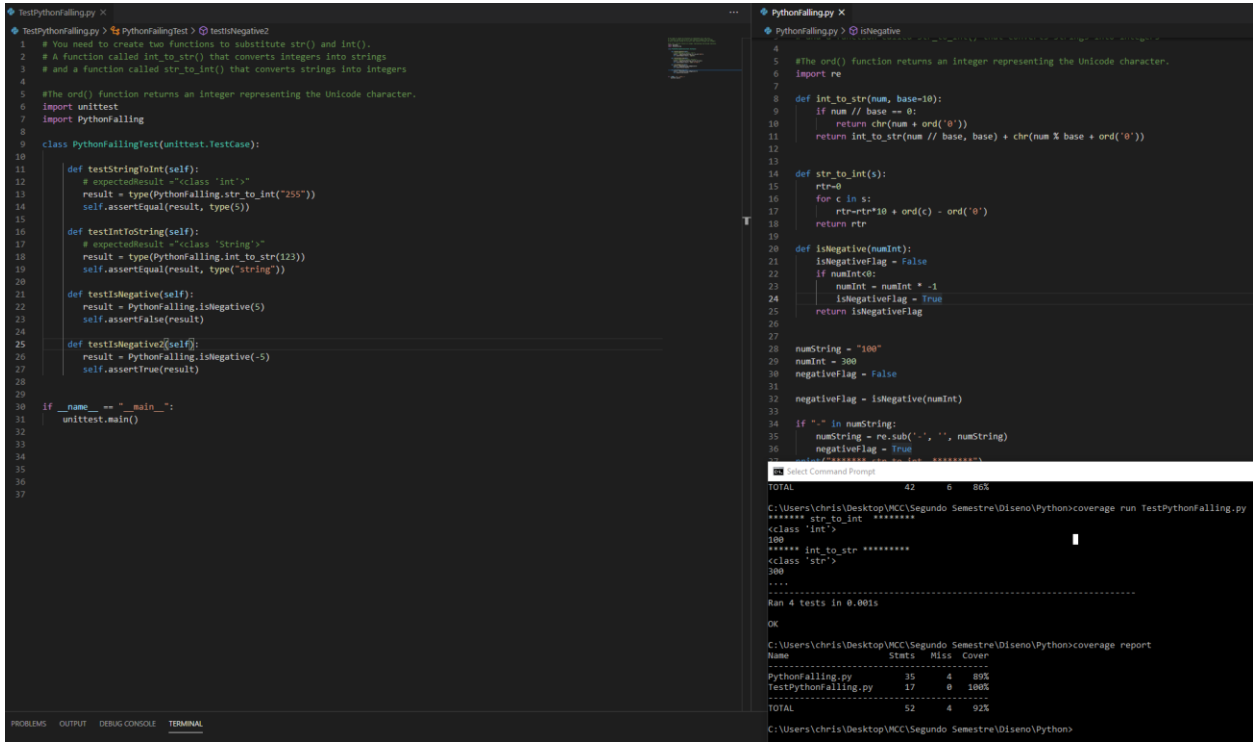

indentation is not a multiple of fourpycodestyle(E111)

missing whitespace after ','pycodestyle(E231)

block comment should start with '# 'pycodestyle(E265)

4. Python Failing

Evidence of code coverage and achieving code coverage. (92%)



```
# You need to create two functions to substitute str() and int().
# A function called int_to_str() that converts integers into strings
# and a function called str_to_int() that converts strings into integers

#The ord() function returns an integer representing the Unicode character.
import unittest
import PythonFalling

class PythonFallingTest(unittest.TestCase):

    def testStringToInt(self):
        # expectedResult ="<class 'int'>"
        result = type(PythonFalling.str_to_int("255"))
        self.assertEqual(result, type(5))

    def testIntToString(self):
        # expectedResult ="<class 'String'>"
        result = type(PythonFalling.int_to_str(123))
        self.assertEqual(result, type("string"))

    def testIsNegative(self):
        result = PythonFalling.isNegative(5)
        self.assertFalse(result)

    def testIsNegative2(self):
        result = PythonFalling.isNegative(-5)
        self.assertTrue(result)


if __name__ == "__main__":
    unittest.main()
```

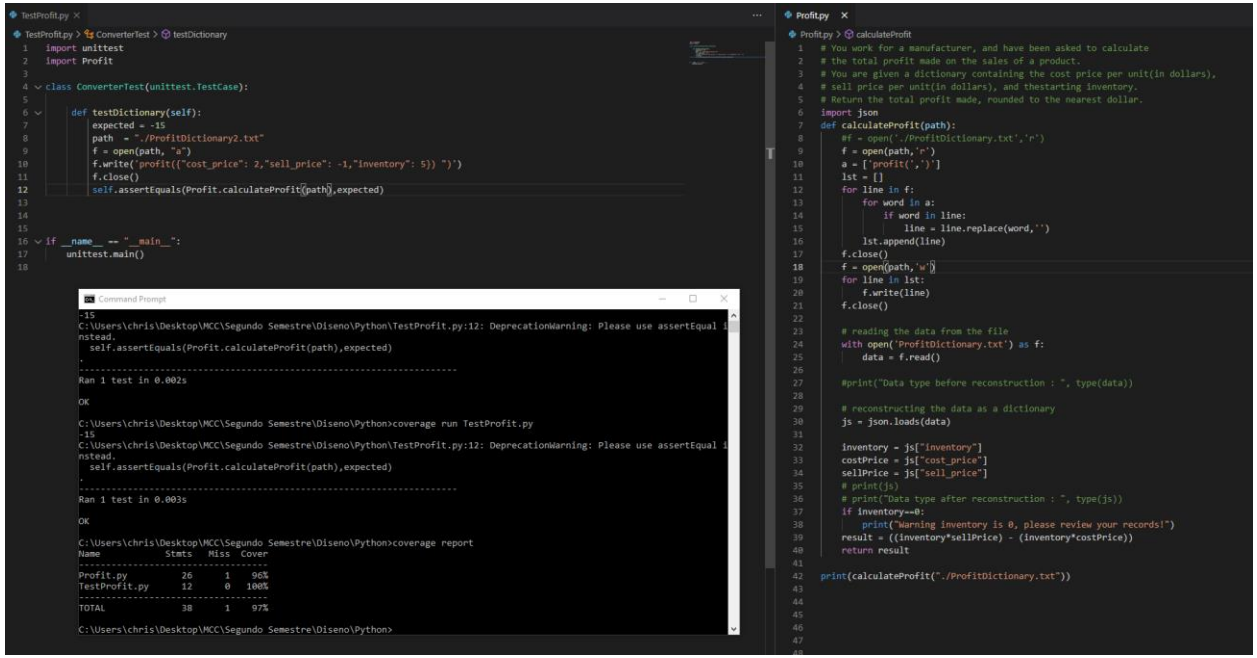
```
#The ord() function returns an integer representing the Unicode character.
import re

def int_to_str(num, base=10):
    if num // base == 0:
        return chr(num + ord('0'))
    return int_to_str(num // base, base) + chr(num % base + ord('0'))


def str_to_int(s):
    rtr=0
    for c in s:
        rtr=rtr*10 + ord(c) - ord('0')
    return rtr

def isNegative(numInt):
    isNegativeFlag = False
    if numInt<0:
        numInt = numInt * -1
        isNegativeFlag = True
    return isNegativeFlag


numString = "100"
numInt = 300
negativeFlag = False

negativeFlag = isNegative(numInt)

if "-" in numString:
    numString = re.sub('-', '', numString)
    negativeFlag = True
```

```
TOTAL                    42      6    86%

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage run TestPythonFalling.py
******* str_to_int  ********
<class 'int'>
100
****** int_to_str *********
<class 'str'>
300
....
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage report
Name                     Stmts   Miss  Cover
--------------------------------------------
PythonFalling.py            35      4    89%
TestPythonFalling.py        17      0   100%
--------------------------------------------
TOTAL                       52      4    92%

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>
```

Pep8 were like previous one’s findings. Here is the corrected version.

5. Profit

Evidence of code coverage and achieving code coverage. (97%)



```
TestProfit.py
import unittest
import Profit

class ConverterTest(unittest.TestCase):

    def testDictionary(self):
        expected = -15
        path  = "./ProfitDictionary2.txt"
        f = open(path, "a")
        f.write('profit({"cost_price": 2,"sell_price": -1,"inventory": 5}) ')
        f.close()
        self.assertEquals(Profit.calculateProfit(path),expected)


if __name__ == "__main__":
    unittest.main()
```

```
Command Prompt
-15
C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\TestProfit.py:12: DeprecationWarning: Please use assertEqual i
nstead.
  self.assertEquals(Profit.calculateProfit(path),expected)
.
----------------------------------------------------------------------
Ran 1 test in 0.002s

OK

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage run TestProfit.py
-15
C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\TestProfit.py:12: DeprecationWarning: Please use assertEqual i
nstead.
  self.assertEquals(Profit.calculateProfit(path),expected)
.
----------------------------------------------------------------------
Ran 1 test in 0.003s

OK

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage report
Name             Stmts   Miss  Cover
------------------------------------
Profit.py           26      1    96%
TestProfit.py       12      0   100%
------------------------------------
TOTAL               38      1    97%

C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>
```

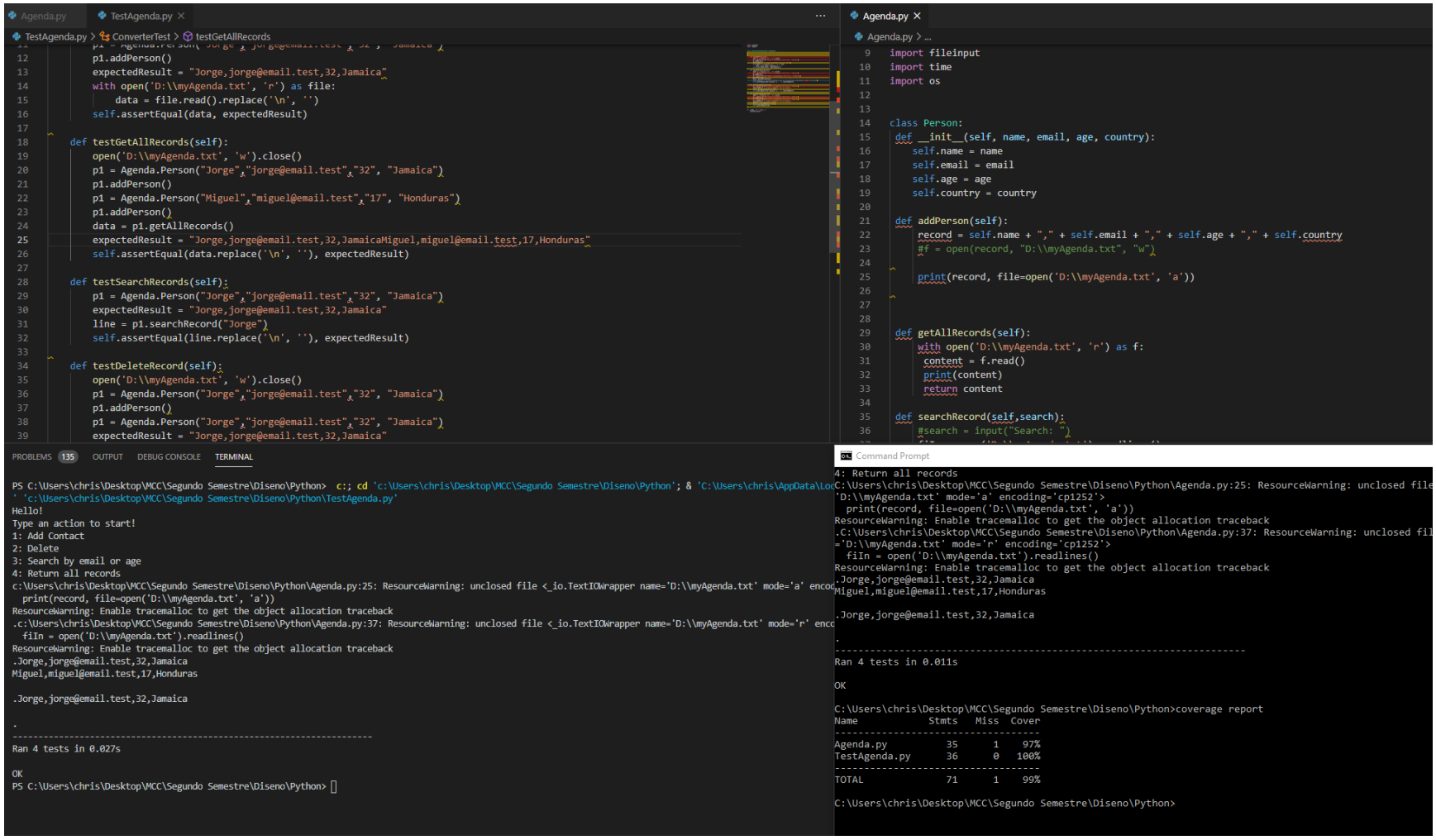
```
Profit.py
# You work for a manufacturer, and have been asked to calculate
# the total profit made on the sales of a product.
# You are given a dictionary containing the cost price per unit(in dollars),
# sell price per unit(in dollars), and thestarting inventory.
# Return the total profit made, rounded to the nearest dollar.
import json
def calculateProfit(path):
    #f = open('./ProfitDictionary.txt','r')
    f = open(path,'r')
    a = ['profit(',')']
    lst = []
    for line in f:
        for word in a:
            if word in line:
                line = line.replace(word,'')
        lst.append(line)
    f.close()
    f = open(path,'w')
    for line in lst:
        f.write(line)
    f.close()

    # reading the data from the file
    with open('ProfitDictionary.txt') as f:
        data = f.read()

    #print("Data type before reconstruction : ", type(data))

    # reconstructing the data as a dictionary
    js = json.loads(data)

    inventory = js["inventory"]
    costPrice = js["cost_price"]
    sellPrice = js["sell_price"]
    # print(js)
    # print("Data type after reconstruction : ", type(js))
    if inventory==0:
        print("Warning inventory is 0, please review your records!")
    result = ((inventory*sellPrice) - (inventory*costPrice))
    return result

print(calculateProfit("./ProfitDictionary.txt"))
```

Pep8 were like previous one’s findings. Here is the corrected version.

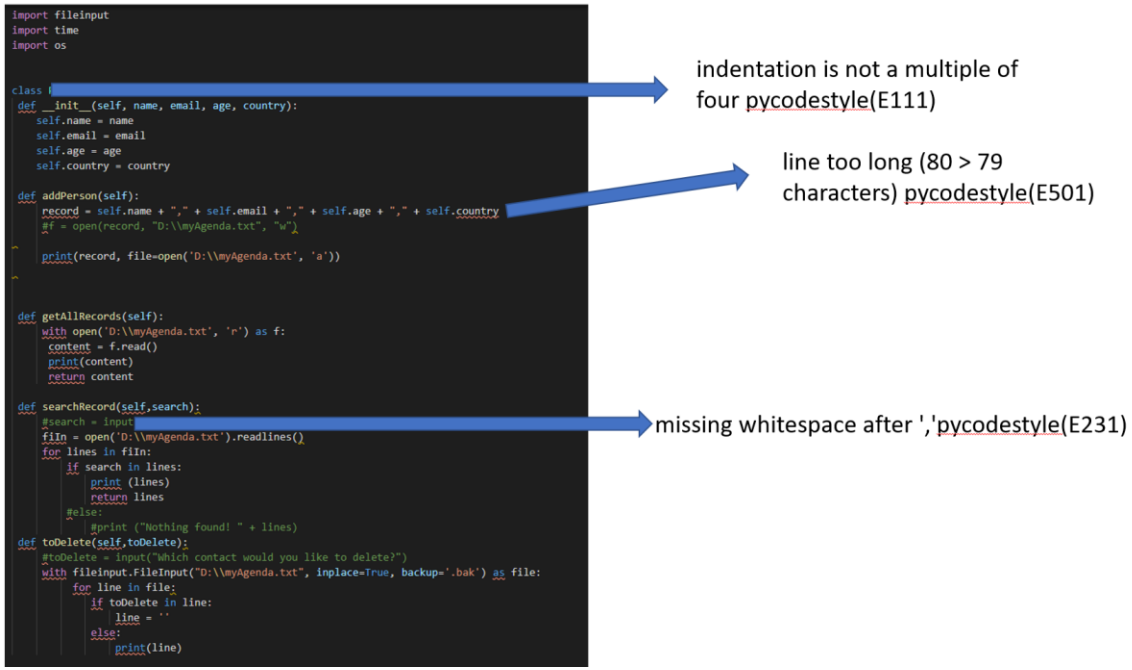
# Pep8


```python
import unittest
import Profit


class ConverterTest(unittest.TestCase):
    def testDictionary(self):
        expected = -15
        path = "./ProfitDictionary2.txt"
        f = open(path, "a")
        f.write('profit({"cost_price": 2,"sell_price": -1,"inventory": 5}) ")')
        f.close()
        self.assertEquals(Profit.calculateProfit(path),expected)


if __name__ == "__main__":
    unittest.main()
```

6. Agenda (99%)


TestAgenda.py
TestAgenda.py > ConverterTest > testGetAllRecords
p1.addPerson()
expectedResult = "Jorge,jorge@email.test,32,Jamaica"
with open('D:\\myAgenda.txt', 'r') as file:
data = file.read().replace('\n', '')
self.assertEqual(data, expectedResult)
def testGetAllRecords(self):
open('D:\\myAgenda.txt', 'w').close()
p1 = Agenda.Person("Jorge","jorge@email.test","32", "Jamaica")
p1.addPerson()
p1 = Agenda.Person("Miguel","miguel@email.test","17", "Honduras")
p1.addPerson()
data = p1.getAllRecords()
expectedResult = "Jorge,jorge@email.test,32,JamaicaMiguel,miguel@email.test,17,Honduras"
self.assertEqual(data.replace('\n', ''), expectedResult)
def testSearchRecords(self):
p1 = Agenda.Person("Jorge","jorge@email.test","32", "Jamaica")
expectedResult = "Jorge,jorge@email.test,32,Jamaica"
line = p1.searchRecord("Jorge")
self.assertEqual(line.replace('\n', ''), expectedResult)
def testDeleteRecord(self):
open('D:\\myAgenda.txt', 'w').close()
p1 = Agenda.Person("Jorge","jorge@email.test","32", "Jamaica")
p1.addPerson()
p1 = Agenda.Person("Jorge","jorge@email.test","32", "Jamaica")
expectedResult = "Jorge,jorge@email.test,32,Jamaica"
PROBLEMS 135
OUTPUT
DEBUG CONSOLE
TERMINAL
PS C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python> c:; cd 'c:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python'; & 'C:\Users\chris\AppData\Loc
' 'c:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\TestAgenda.py'
Hello!
Type an action to start!
1: Add Contact
2: Delete
3: Search by email or age
4: Return all records
c:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\Agenda.py:25: ResourceWarning: unclosed file <_io.TextIOWrapper name='D:\\myAgenda.txt' mode='a' enco
print(record, file=open('D:\\myAgenda.txt', 'a'))
ResourceWarning: Enable tracemalloc to get the object allocation traceback
.c:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\Agenda.py:37: ResourceWarning: unclosed file <_io.TextIOWrapper name='D:\\myAgenda.txt' mode='r' enc
fiIn = open('D:\\myAgenda.txt').readlines()
ResourceWarning: Enable tracemalloc to get the object allocation traceback
.Jorge,jorge@email.test,32,Jamaica
Miguel,miguel@email.test,17,Honduras
.Jorge,jorge@email.test,32,Jamaica
.
Ran 4 tests in 0.027s
OK
PS C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>
Agenda.py
Agenda.py > ...
import fileinput
import time
import os
class Person:
def __init__(self, name, email, age, country):
self.name = name
self.email = email
self.age = age
self.country = country
def addPerson(self):
record = self.name + "," + self.email + "," + self.age + "," + self.country
#f = open(record, "D:\\myAgenda.txt", "w")
print(record, file=open('D:\\myAgenda.txt', 'a'))
def getAllRecords(self):
with open('D:\\myAgenda.txt', 'r') as f:
content = f.read()
print(content)
return content
def searchRecord(self,search):
#search = input("Search: ")
Command Prompt
4: Return all records
C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\Agenda.py:25: ResourceWarning: unclosed file
'D:\\myAgenda.txt' mode='a' encoding='cp1252'>
print(record, file=open('D:\\myAgenda.txt', 'a'))
ResourceWarning: Enable tracemalloc to get the object allocation traceback
.C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python\Agenda.py:37: ResourceWarning: unclosed fil
='D:\\myAgenda.txt' mode='r' encoding='cp1252'>
fiIn = open('D:\\myAgenda.txt').readlines()
ResourceWarning: Enable tracemalloc to get the object allocation traceback
.Jorge,jorge@email.test,32,Jamaica
Miguel,miguel@email.test,17,Honduras
.Jorge,jorge@email.test,32,Jamaica
.
Ran 4 tests in 0.011s
OK
C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>coverage report
Name Stmts Miss Cover
Agenda.py 35 1 97%
TestAgenda.py 36 0 100%
TOTAL 71 1 99%
C:\Users\chris\Desktop\MCC\Segundo Semestre\Diseno\Python>

Pep 8




Line lengths are recommended to be no greater than 79 characters. The reasoning for this comes from PEP8 itself:

> *Limiting the required editor window width makes it possible to have several files open side-by-side, and works well when using code review tools that present the two versions in adjacent columns.*

It is fairly common for developers, especially those in closed-source projects, to change the maximum line length to 100 or 120 characters.

## Additional links

- https://www.python.org/dev/peps/pep-0008/#maximum-line-length