

Algoritmos y Programación 1

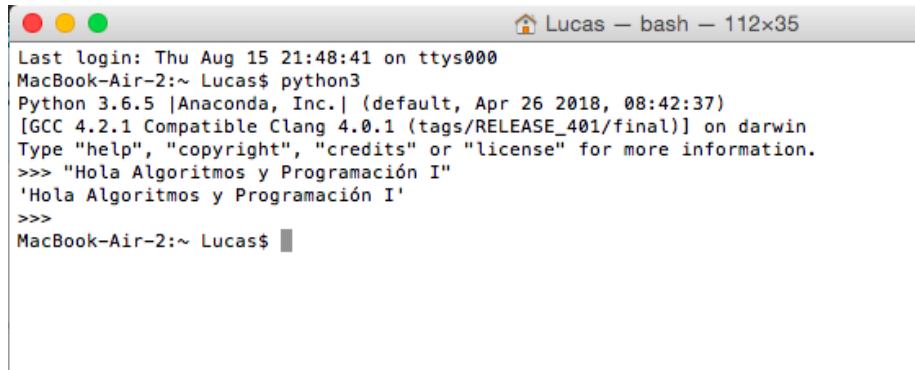
Cátedra Essaya - Práctica Grace

Lucas Alejo Pavlov - Legajo 105412

EJ1 - 2 de septiembre de 2019

Parte 1

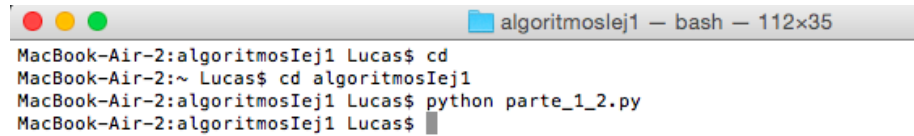
Parte 1.1

A screenshot of a macOS terminal window. The title bar at the top shows three colored window control buttons (red, yellow, green) on the left and a home icon followed by the text "Lucas — bash — 112x35" on the right. The terminal content shows the following text:

```
Last login: Thu Aug 15 21:48:41 on ttys000
MacBook-Air-2:~ Lucas$ python3
Python 3.6.5 |Anaconda, Inc.| (default, Apr 26 2018, 08:42:37)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> "Hola Algoritmos y Programación I"
'Hola Algoritmos y Programación I'
>>>
MacBook-Air-2:~ Lucas$ █
```

Figura 1: Captura de pantalla correspondiente a la parte 1.1.

Parte 1.2



```
MacBook-Air-2:algoritmosIej1 Lucas$ cd
MacBook-Air-2:~ Lucas$ cd algoritmosIej1
MacBook-Air-2:algoritmosIej1 Lucas$ python parte_1_2.py
MacBook-Air-2:algoritmosIej1 Lucas$
```

Figura 2: Captura de pantalla correspondiente a la parte 1.2.

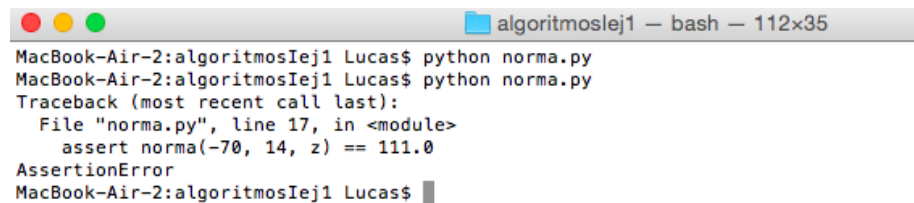
5) Para obtener el mismo resultado que en la parte 1.1 debería usar la función `print`, reemplazando la línea

```
'Hola Algoritmos y Programación I'
```

por

```
print('Hola Algoritmos y Programación I')
```

Parte 2



```
MacBook-Air-2:algoritmosIej1 Lucas$ python norma.py
MacBook-Air-2:algoritmosIej1 Lucas$ python norma.py
Traceback (most recent call last):
  File "norma.py", line 17, in <module>
    assert norma(-70, 14, z) == 111.0
AssertionError
MacBook-Air-2:algoritmosIej1 Lucas$
```

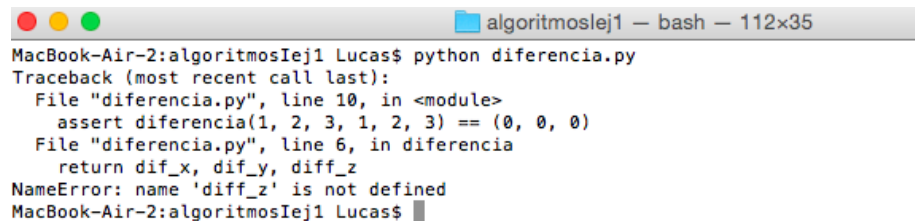
Figura 3: Captura de pantalla correspondiente a la parte 2.

4.1) El programa no tiene ninguna salida salvo que alguno de los asserts sea **False**. En ese caso la salida es un **AssertionError** que ocurre en la primera línea en la cual la expresión que se le pasa a **assert** es **False**.

4.2) El error se generó en la línea 17. Significa que la expresión luego de **assert** en esa línea es **False** (en el sentido booleano), ya que **norma(-70, 14, -80)** no es igual a 111.0. Para solucionarlo habría que cambiar esa expresión para que de un resultado verdadero. Se podría hacer cambiando una o más de las entradas de norma y/o cambiando el resultado de norma, aunque todo hace indicar que la forma que se pide sería modificando la línea 16, que haría que la entrada de norma que se modifique sea la tercera. Habría que elegir un valor de **z** que haga que **norma(-70, 14, z)** sea igual a 111.0. Los valores que hacen eso son $z = 85$ y $z = -85$.

4.3) En el punto 2 no imprimió nada porque todas las expresiones luego de **assert** que fueron ejecutadas (es decir, desde la línea 5 hasta la 14, ya que la de la línea 17 estaba comentada) eran **True**. La instrucción **assert** no hace nada si lo que está luego de ella es **True**, en caso contrario (**False**) termina la ejecución del programa y sale del mismo devolviendo un error.

Parte 3



```
MacBook-Air-2:algoritmosIej1 Lucas$ python diferencia.py
Traceback (most recent call last):
  File "diferencia.py", line 10, in <module>
    assert diferencia(1, 2, 3, 1, 2, 3) == (0, 0, 0)
  File "diferencia.py", line 6, in diferencia
    return dif_x, dif_y, diff_z
NameError: name 'diff_z' is not defined
MacBook-Air-2:algoritmosIej1 Lucas$
```

Figura 4: Captura de pantalla correspondiente a la parte 3.

4) Se detectó un error en la última línea de la definición de la función **diferencia**, ya que se estaba pidiendo devolver una variable que no había sido definida anteriormente (**diff_z**). Para corregir el error cambié la línea

```
return dif_x , dif_y , diff_z
```

por

```
return dif_x , dif_y , dif_z
```

ya que `diff_z` es la variable que fue definida en el cuerpo de la función. Luego de hacer ese cambio, al correr el programa no sale ningún mensaje, con lo cual todas las condiciones planteadas luego de los `assert` son `True`.

Parte 4

4) Muestra un `AssertionError` en la línea 10. Luego, si se comenta esa línea, aparece un error en la línea 11, y así sucesivamente, todas las líneas debajo de la 10 dan `AssertionError`.

5) Incluyendo llamadas a `print` en la función (antes de `return`) se ve que el problema está en la segunda componente del producto vectorial (`var2`). Viendo el cuerpo de la función en esa línea se ve que hay una potenciación en lugar de una multiplicación. Corrigiendo ese bug, el programa se ejecuta sin dar ningún `AssertionError`. Más específicamente, hay que cambiar la línea

```
var2 = z1**x2 - x1*z2
```

por

```
var2 = z1*x2 - x1*z2
```

4.7) Si. La línea sería:

```
return y1*z2 - z1*y2, z1*x2 - x1*z2, x1*y2 - y1*x2
```

Parte 5

5) La reutilización de funciones es importante ya que permite tener un código mucho más ordenado, y hacer un programa por partes puede ahorrarnos tiempo en futuros programas que requieran partes (funciones) similares, es decir, programando las distintas funciones tengo la posibilidad de reutilizarlas más adelante en otros programas ahorrando tiempo.

Además, si alguna función tiene un bug y se usa muchas veces en un programa, alcanza con corregir la función para corregir el bug, mientras que si no se escribe el código del programa via funciones sino de manera explícita, a la hora de corregir el bug habría que hacerlo en distintos lugares del código, lo cual aumenta el tiempo de trabajo y también la probabilidad de saltarse alguna corrección y que el bug persista en alguna instancia del programa final.