

Informe del Ejercicio 1

“Algoritmos y programación I (95.14)”

Cátedra Essaya Diego

Práctica: Alan

Alumno: Higa, Axel Tomás

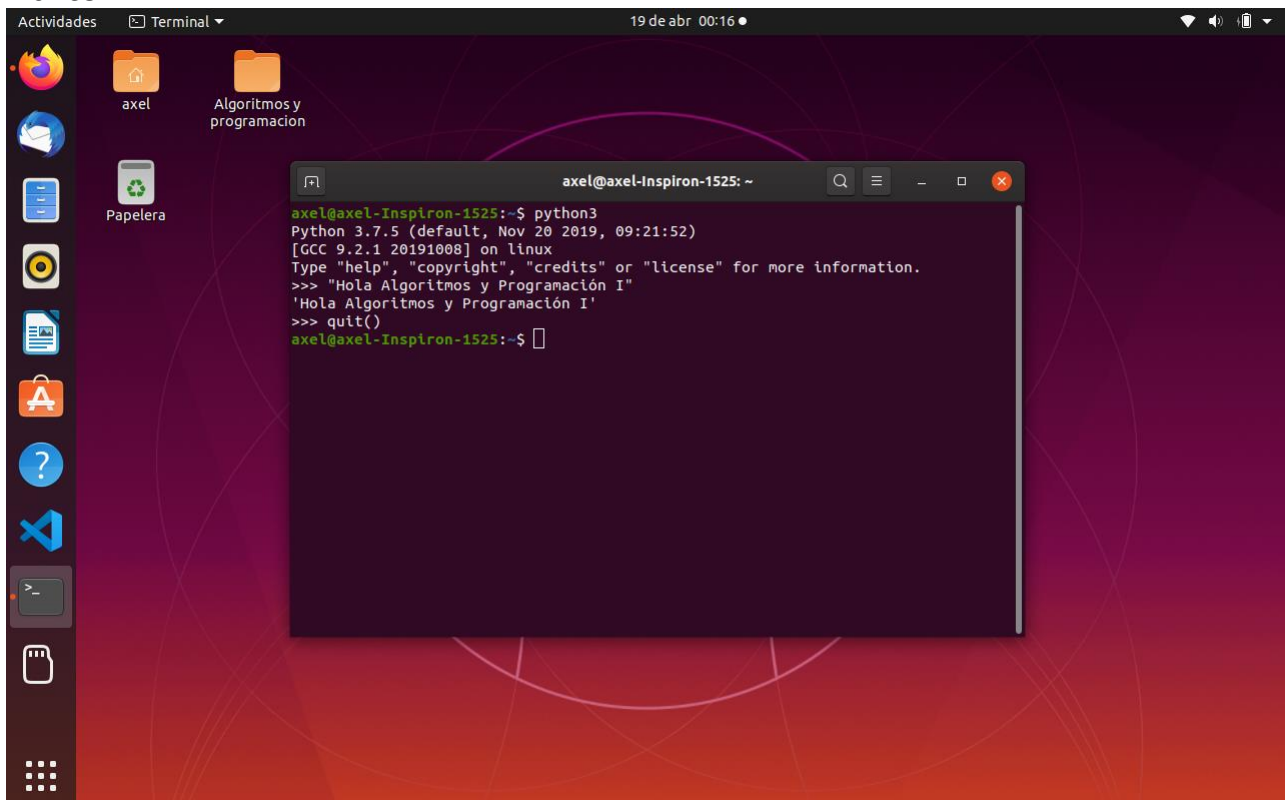
Padrón: 105848

Corrector: Sportelli, Luciano

Resolución

Parte 1

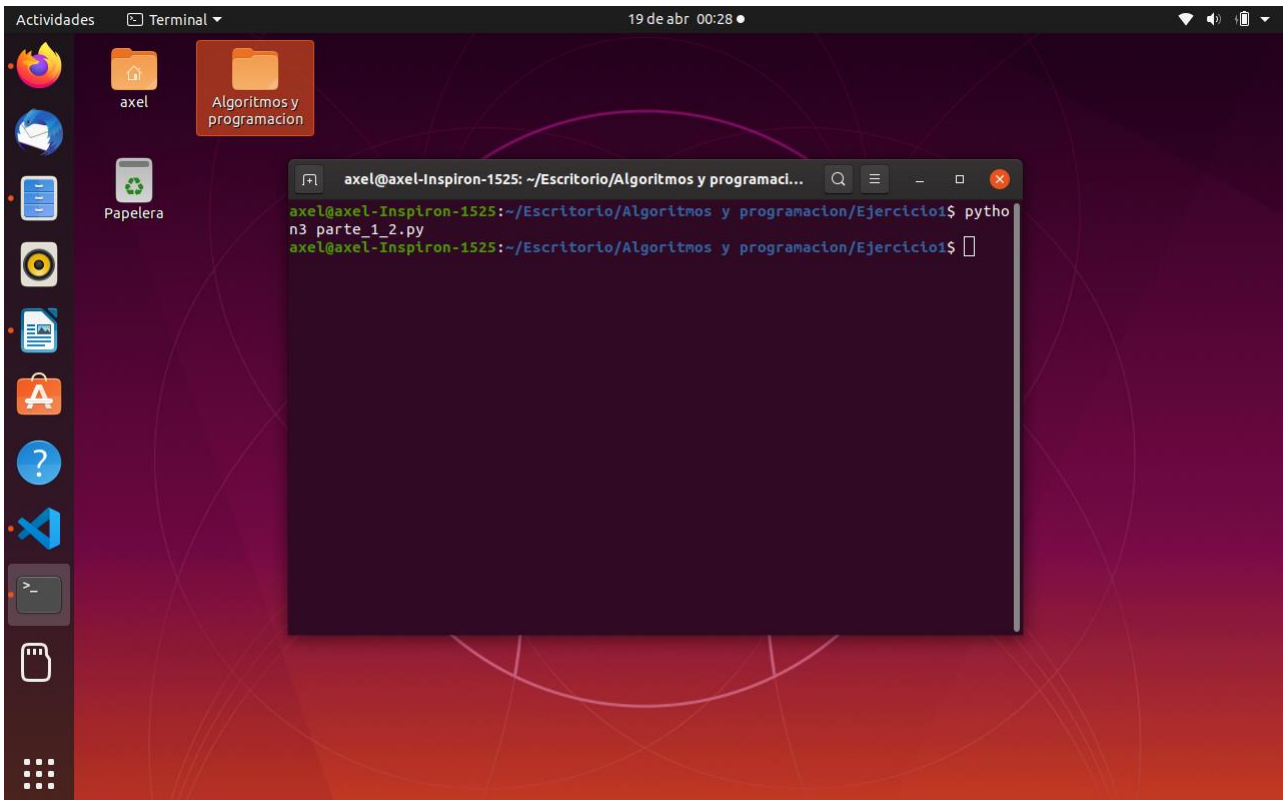
Parte1.1



The screenshot shows a Linux desktop environment with a dark theme. The desktop background is a dark purple with a faint geometric pattern. On the left side, there is a vertical dock with several application icons: a web browser, a file manager, a terminal, a music player, a video player, a document viewer, a shopping bag, a question mark, a code editor, a terminal, a file manager, and a grid icon. The top of the screen shows a panel with the date and time '19 de abr 00:16' and system status icons. A terminal window is open in the center, displaying the following text:

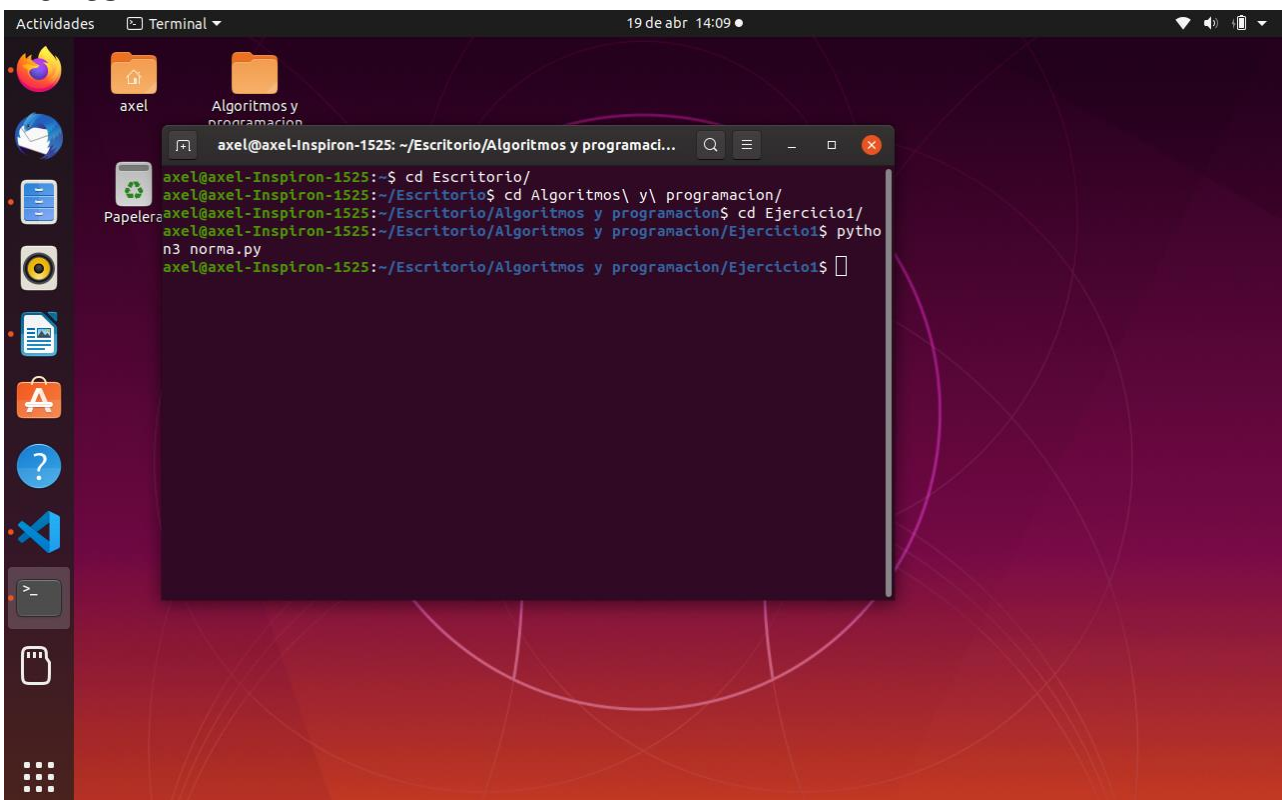
```
axel@axel-Inspiron-1525: ~  
axel@axel-Inspiron-1525:~$ python3  
Python 3.7.5 (default, Nov 20 2019, 09:21:52)  
[GCC 9.2.1 20191008] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> "Hola Algoritmos y Programación I"  
'Hola Algoritmos y Programación I'  
>>> quit()  
axel@axel-Inspiron-1525:~$
```

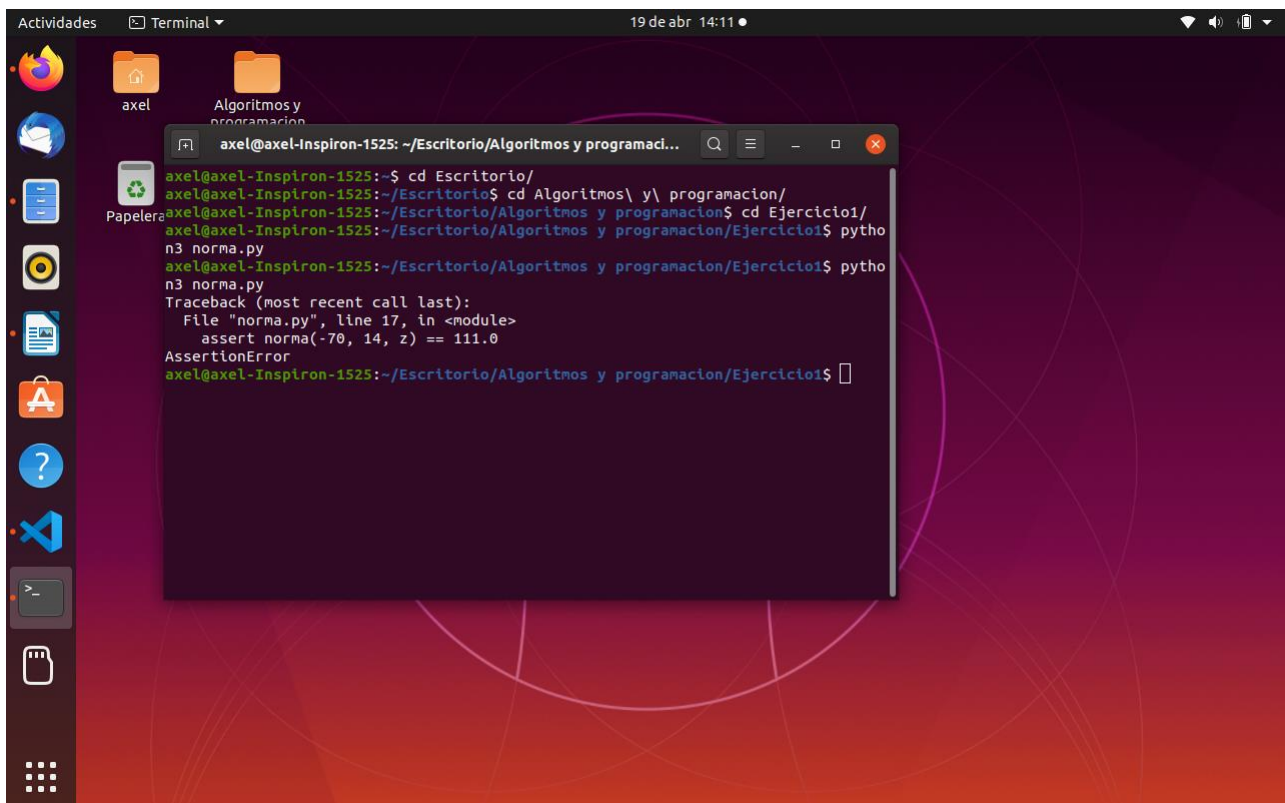
Parte1.2



5) Para que sea parecido al punto 1.1 del ejercicio se debe usar la función print, en la parte 1.1 podemos ver el resultado sin usar la función ya que utilizamos el intérprete en la terminal el cual podemos introducir las órdenes o instrucciones las cuales interpreta con el lenguaje seleccionado, en este caso python, y repite lo que introdujimos. En cambio en el ejercicio 1.2 se trata de correr un programa, y esta al no tener ninguna función definida o instrucción, solo hay una cadena de texto, el intérprete no sabe qué hacer, para eso debemos usar la función print si queremos mostrarle el mensaje al usuario.

Parte 2



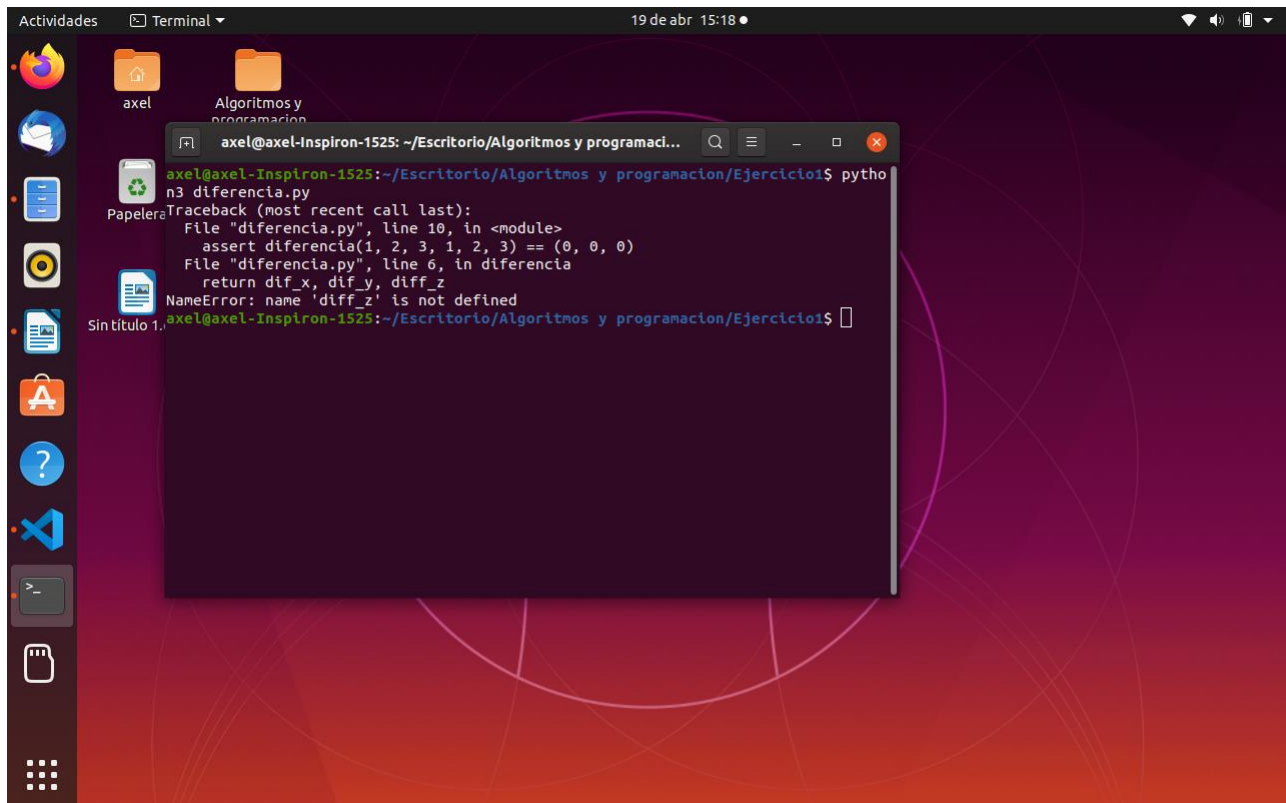


The screenshot shows a terminal window on a Linux desktop. The user 'axel' is in the directory '~/Escritorio/Algoritmos y programacion/Ejercicio1'. They run 'python3 norma.py', which results in an 'AssertionError'. The error message indicates that the assertion 'norma(-70, 14, z) == 111.0' failed at line 17 of the script. The terminal output is as follows:

```
axel@axel-Inspiron-1525:~$ cd Escritorio/
axel@axel-Inspiron-1525:~/Escritorio$ cd Algoritmos\ y\ programacion/
axel@axel-Inspiron-1525:~/Escritorio/Algoritmos y programacion$ cd Ejercicio1/
axel@axel-Inspiron-1525:~/Escritorio/Algoritmos y programacion/Ejercicio1$ python3 norma.py
axel@axel-Inspiron-1525:~/Escritorio/Algoritmos y programacion/Ejercicio1$ python3 norma.py
Traceback (most recent call last):
  File "norma.py", line 17, in <module>
    assert norma(-70, 14, z) == 111.0
AssertionError
axel@axel-Inspiron-1525:~/Escritorio/Algoritmos y programacion/Ejercicio1$
```

- 1) La salida del programa es un mensaje de error de código.
- 2) Si podemos saber en qué línea se generó el error, ya que en el intérprete en la segunda línea del mensaje de error muestra que es en la línea 17 "File "norma.py", line 17, in <module>"
- 3) La función assert en una instrucción de python que te permite definir condiciones que deban cumplir siempre, sirve para comprobar valores de variables, expresiones y estados del programa. En caso de expresiones booleanas si es true la instrucción no hace nada, en caso contrario si es false dispara una excepción.
- 4) El valor de z para que el resultado sea 111.0 es de 85 o -85
- 5) Es incorrecto cambiar el valor del resultado ya que en el ejercicio, el parámetro z de una función assert no está definida por un valor, además de que el resultado 111.0 no es parámetro ni variable, por eso se cambia el valor de z y no el resultado. En la función se introducen parámetros, y lo que se busca con assert es que la condición sea verdadera, es decir, queremos que los parámetros introducidos y su relación con el resultado sean correctos.

Parte 3



```
axel@axel-Inspiron-1525: ~/Escritorio/Algoritmos y programacion/Ejercicio1$ python3 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
axel@axel-Inspiron-1525: ~/Escritorio/Algoritmos y programacion/Ejercicio1$
```

4) Si, se detectó un error, es un error de definición provocado por un error de tipeo de la variable. Mientras que en la función utilizamos la variable `dif_z` en la devolución está escrito `diff_z` que tiene doble f en vez de una y al tener distintos nombres el intérprete devuelve un error en el que la variable `diff_z` no está definida en la función `diferencia`. Este error se encuentra en la línea 6 del código.

Parte 4

- 4) Se muestra un error de la función `Assert`, “`AssertionError`”, en la línea 10 del código.
- 6) Es importante que el nombre de la función y las variables sean representativas porque es necesario que otra persona pueda leer el código fácilmente, además si se quiere reutilizar el código es importante que pueda entender que hace cada función y que variables son las que utilizan, en síntesis el código debe ser legible. En caso contrario, si el código no es muy deductivo y confuso, será más difícil entender que es lo que se quiere lograr con el programa.
- 7) Si se puede escribir el cuerpo de la función en una sola línea, en lugar de utilizar las variables se devuelve directamente la operación, es decir, en lugar de devolver, por ejemplo, la variable “`variable_x` se” se devuelve directamente la operación “`y1*z2 - z1*y2`”.

Parte 5

- 4) Es importante reutilizar las funciones ya que te permite escribir un código más simple y acortado, en lugar de estar escribiendo la misma función varias veces, y además de que se pueden utilizar de distintas formas dependiendo de lo que quiera hacer el usuario.