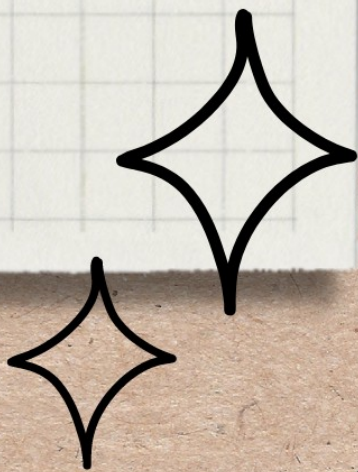





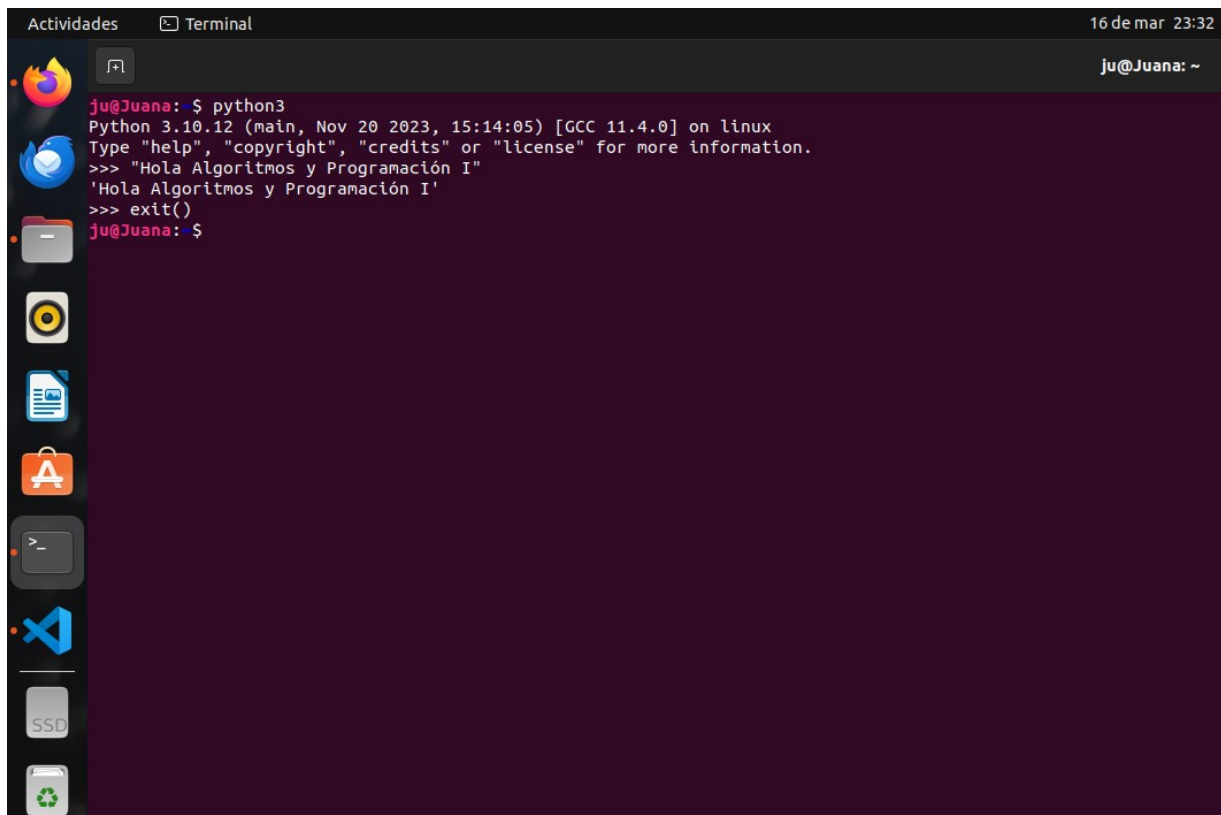
FUNDAMENTOS DE PROGRAMACIÓN

ALUMNA: JUANA REHL
MATERIA: FUNDAMENTOS DE
PROGRAMACIÓN
CÁTEDRA: DIEGO ESSAYA
PRACTICA: ALAN
PADRÓN 112185

Juana Rehl



Parte 1,1



A terminal window titled "Terminal" with a dark background. The prompt is "ju@Juana: ~". The user has entered "python3", which has started the Python 3.10.12 interpreter. The interpreter shows the version and date, then prompts for help. The user has entered a multi-line string: ">>> \"Hola Algoritmos y Programación I\"". The interpreter has printed the string and then prompted for exit. The user has entered "exit()", and the prompt has returned to "ju@Juana: \$".

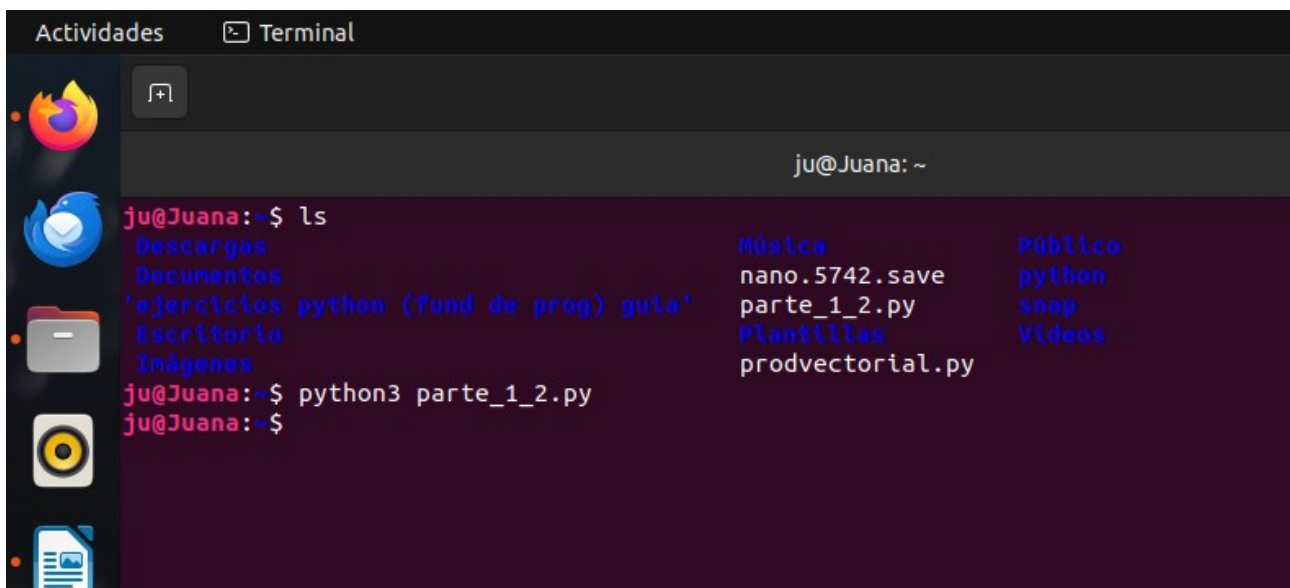
```
ju@Juana:~$ python3
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> "Hola Algoritmos y Programación I"
'Hola Algoritmos y Programación I'
>>> exit()
ju@Juana:~$
```

Parte 1,2



A terminal window titled "Terminal" with a dark background. The prompt is "ju@Juana: ~". The user has entered ">parte_1_2.py".

```
ju@Juana:~$ >parte_1_2.py
```



A terminal window titled 'Terminal' showing a user named 'ju' at a machine named 'Juana'. The prompt is 'ju@Juana: ~'. The user enters the command 'ls', which lists the contents of the home directory. The output is as follows:

Column 1	Column 2	Column 3
Descargas	Música	Público
Documentos	nano.5742.save	python
'ejercicios python (fund de prog) guia'	parte_1_2.py	snap
Escritorio	Plantillas	Videos
Imágenes	prodvectorial.py	

After the listing, the user enters 'python3 parte_1_2.py' and the prompt returns to 'ju@Juana: ~\$'.

(las capturas están separadas ya que modifique el archivo usando nano, para las demás actividades use Visual Studio Code)

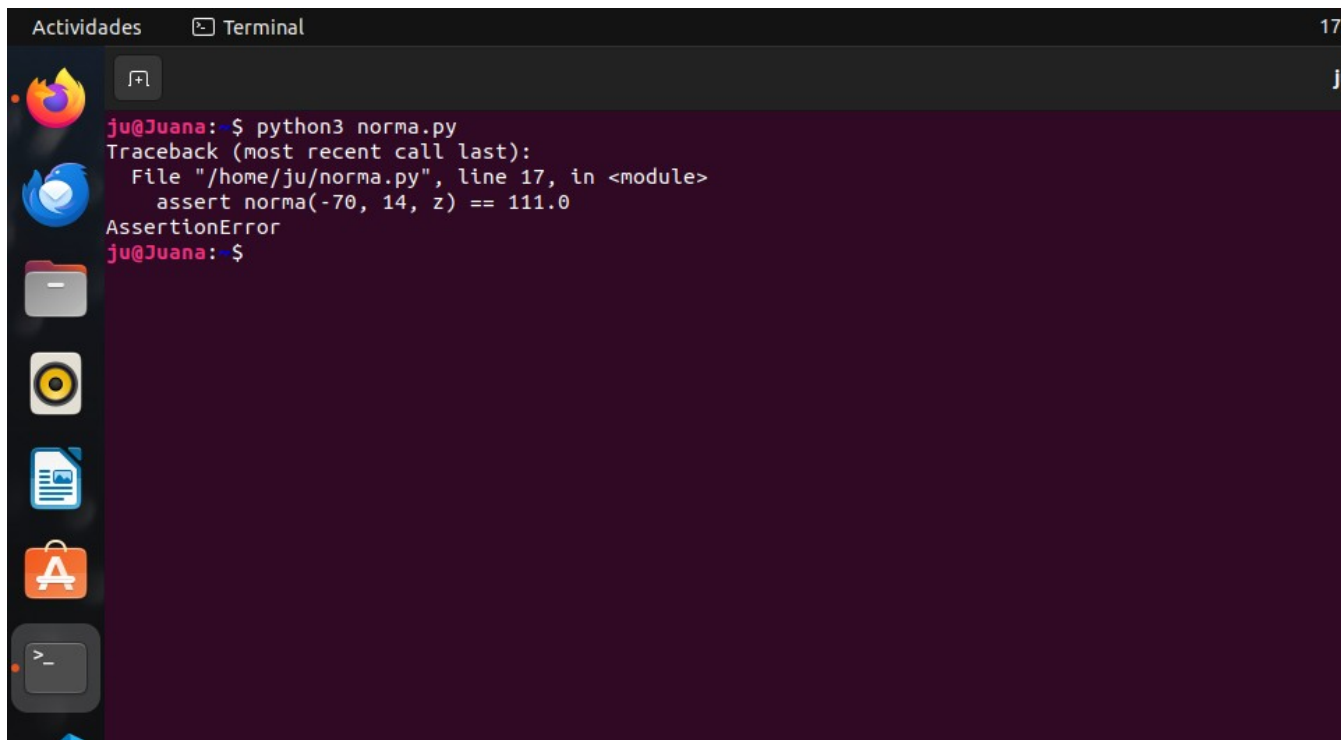
5-¿Qué función debo usar para conseguir el mismo resultado que en la parte 1.1? ¿Por qué en la parte 1.1 vemos el resultado aun sin haber usado esta función?

Para conseguir el mismo resultado que en la parte 1.1 debo usar el comando print para que se muestre el texto en pantalla. En la parte 1.1 podemos ver el resultado sin haber utilizado la función ya que nos encontrábamos dentro del programa Python. En cambio, en la parte 1.2 se agrego el texto a un archivo sin utilizar ningún comando.

Parte 2



A terminal window titled 'Terminal' showing a user named 'ju' at a machine named 'Juana'. The prompt is 'ju@Juana: ~'. The user enters the command 'python3 norma.py', and the prompt returns to 'ju@Juana: ~\$'.



```
ju@Juana:~$ python3 norma.py
Traceback (most recent call last):
  File "/home/ju/norma.py", line 17, in <module>
    assert norma(-70, 14, z) == 111.0
AssertionError
ju@Juana:~$
```

1. ¿Cuál es la salida del programa?

La salida del programa es un error, mas preciso, en la linea 17 cuando realiza el assert.

2. ¿Podemos saber en qué línea se generó el error? ¿Cómo?

Como dije en el ejercicio anterior, si se puede saber la linea en que se genero el error (en este caso es la 17). Podemos saber esto ya que dice “line 17, in <module>” lo cual nos indica la linea del error. Además nos damos cuenta porque el assert que nos dice que tiene error coincide con el assert de la linea 17.

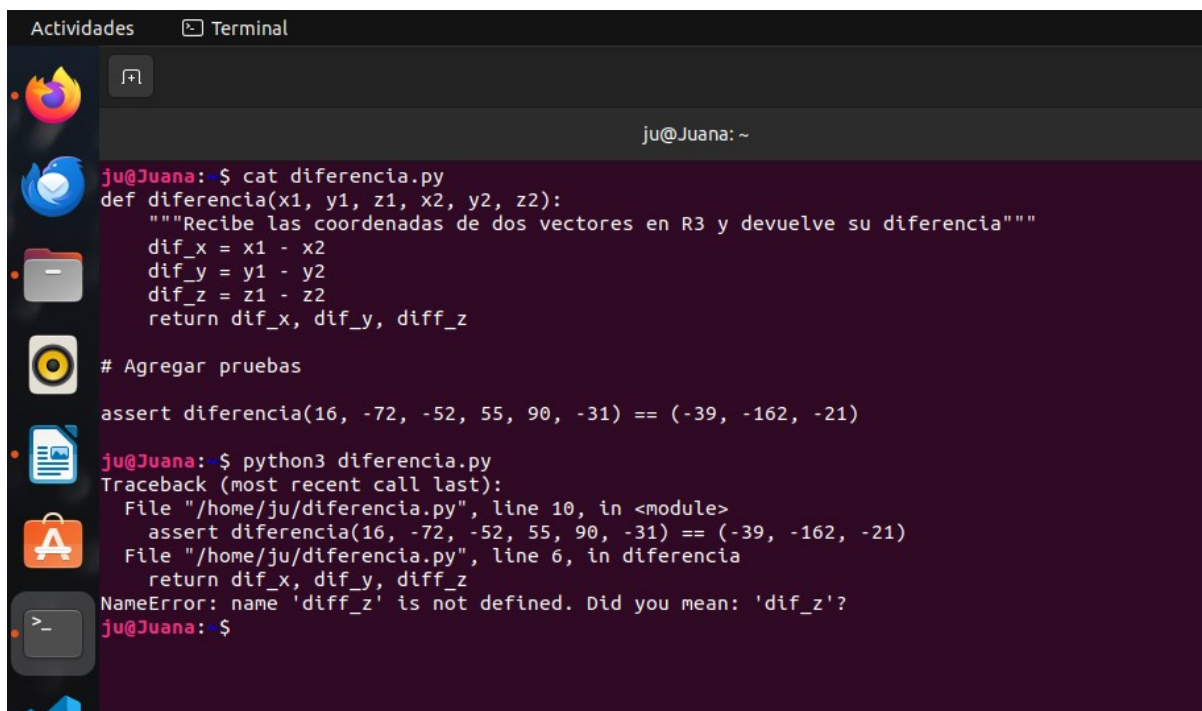
3. ¿Qué hace la instrucción assert?

El assert nos permite verificar si lo que le estamos dando (una condición) es verdadera. En caso que no lo sea nos informa sobre el error.

4. Solucionar el problema. Hallar el valor de z para que ya no de error.

Para que no de error el valor de z debe ser 85. Cambiando z a este valor no nos da error.

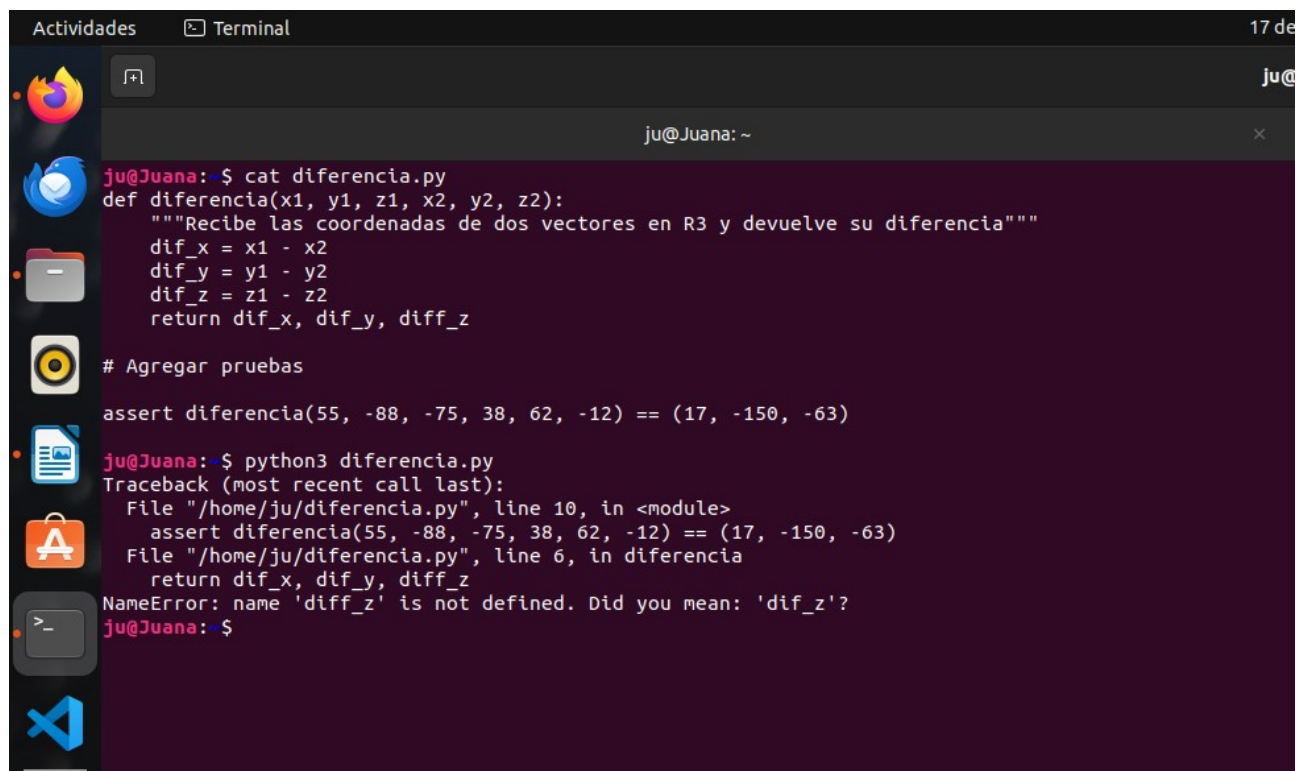
Parte 3



```
Actividades Terminal
ju@Juana: ~
ju@Juana:~$ cat diferencia.py
def diferencia(x1, y1, z1, x2, y2, z2):
    """Recibe las coordenadas de dos vectores en R3 y devuelve su diferencia"""
    dif_x = x1 - x2
    dif_y = y1 - y2
    dif_z = z1 - z2
    return dif_x, dif_y, diff_z

# Agregar pruebas
assert diferencia(16, -72, -52, 55, 90, -31) == (-39, -162, -21)

ju@Juana:~$ python3 diferencia.py
Traceback (most recent call last):
  File "/home/ju/diferencia.py", line 10, in <module>
    assert diferencia(16, -72, -52, 55, 90, -31) == (-39, -162, -21)
  File "/home/ju/diferencia.py", line 6, in diferencia
    return dif_x, dif_y, diff_z
NameError: name 'diff_z' is not defined. Did you mean: 'dif_z'?
ju@Juana:~$
```



```
Actividades Terminal 17 de
ju@Juana: ~
ju@Juana:~$ cat diferencia.py
def diferencia(x1, y1, z1, x2, y2, z2):
    """Recibe las coordenadas de dos vectores en R3 y devuelve su diferencia"""
    dif_x = x1 - x2
    dif_y = y1 - y2
    dif_z = z1 - z2
    return dif_x, dif_y, diff_z

# Agregar pruebas
assert diferencia(55, -88, -75, 38, 62, -12) == (17, -150, -63)

ju@Juana:~$ python3 diferencia.py
Traceback (most recent call last):
  File "/home/ju/diferencia.py", line 10, in <module>
    assert diferencia(55, -88, -75, 38, 62, -12) == (17, -150, -63)
  File "/home/ju/diferencia.py", line 6, in diferencia
    return dif_x, dif_y, diff_z
NameError: name 'diff_z' is not defined. Did you mean: 'dif_z'?
ju@Juana:~$
```

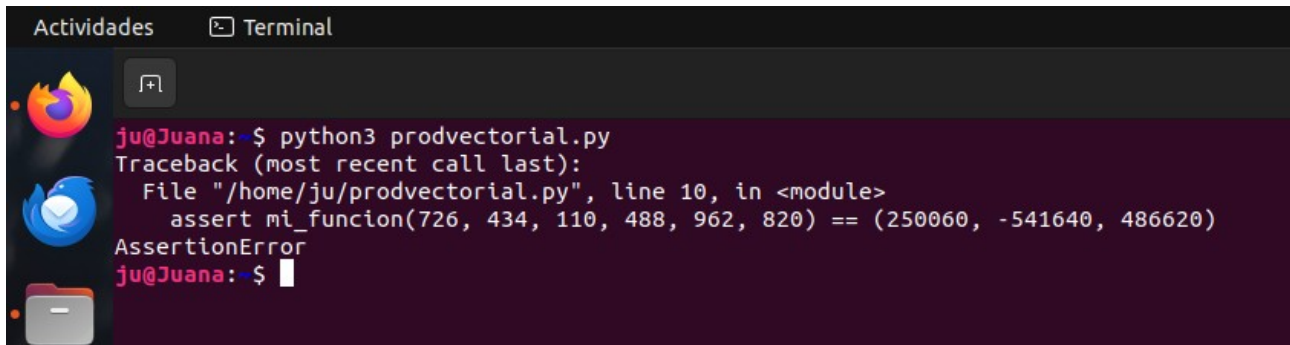
4. ¿Se detectó algún error? ¿Cuál era? ¿Qué significa? ¿Qué línea estaba fallando?

Si se detecto un error. El error es uno de tipo “NameError” lo que significa que algún nombre, ya sea por ejemplo de una variable, esta mal escrito y el programa no lo puede encontrar. La línea que estaba fallando es la 6: “line6, in diferencia...”

Parte 4

4. ¿Qué error muestra? ¿En qué línea?

Muestra un error de tipo “AssertionError”, es el error de assert explicado en el ejercicio parte 2. Se da en la linea 10: “line 10, in <module>”

A screenshot of a terminal window titled "Terminal" with a dark background. The prompt is "ju@Juana:~\$". The user has run "python3 prodvectorial.py". The output shows a traceback: "Traceback (most recent call last):", "File \"/home/ju/prodvectorial.py", line 10, in <module>", "assert mi_funcion(726, 434, 110, 488, 962, 820) == (250060, -541640, 486620)", and "AssertionError". The prompt returns to "ju@Juana:~\$". On the left side of the terminal window, there are icons for Firefox, a mail client, and a file manager.

```
ju@Juana:~$ python3 prodvectorial.py
Traceback (most recent call last):
  File "/home/ju/prodvectorial.py", line 10, in <module>
    assert mi_funcion(726, 434, 110, 488, 962, 820) == (250060, -541640, 486620)
AssertionError
ju@Juana:~$
```

7. Renombrar la función y las variables de forma que sus nombres sean representativos. ¿Por qué es importante hacer esto?

Es importante tener nombres representativos para saber como funciona la función y que significan las variables. Es más fácil de entender una función si los nombres de esta y sus variables están relacionados con su función en el programa.

(Parte 5 en los archivos.py enviados.)