



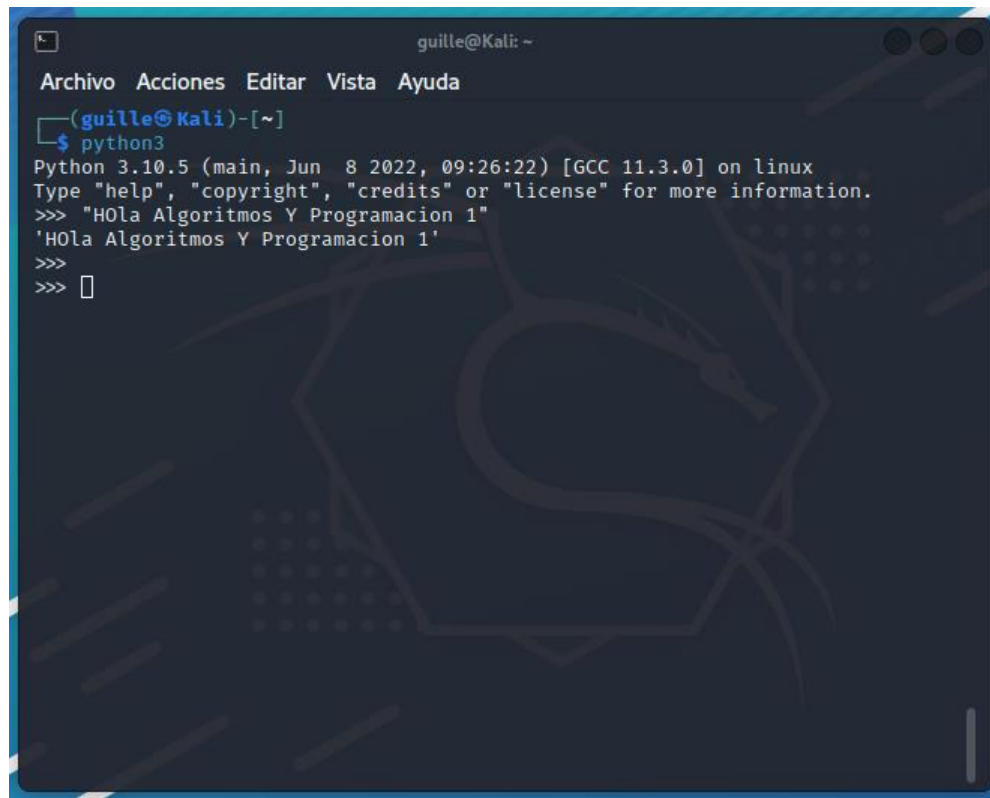
# Ej1-Área de un Polígono

ALGORITMOS Y PROGRAMACION 1 ESSAYA

Guillermo Silva|109777|Practica Alan | Martin Maddalena

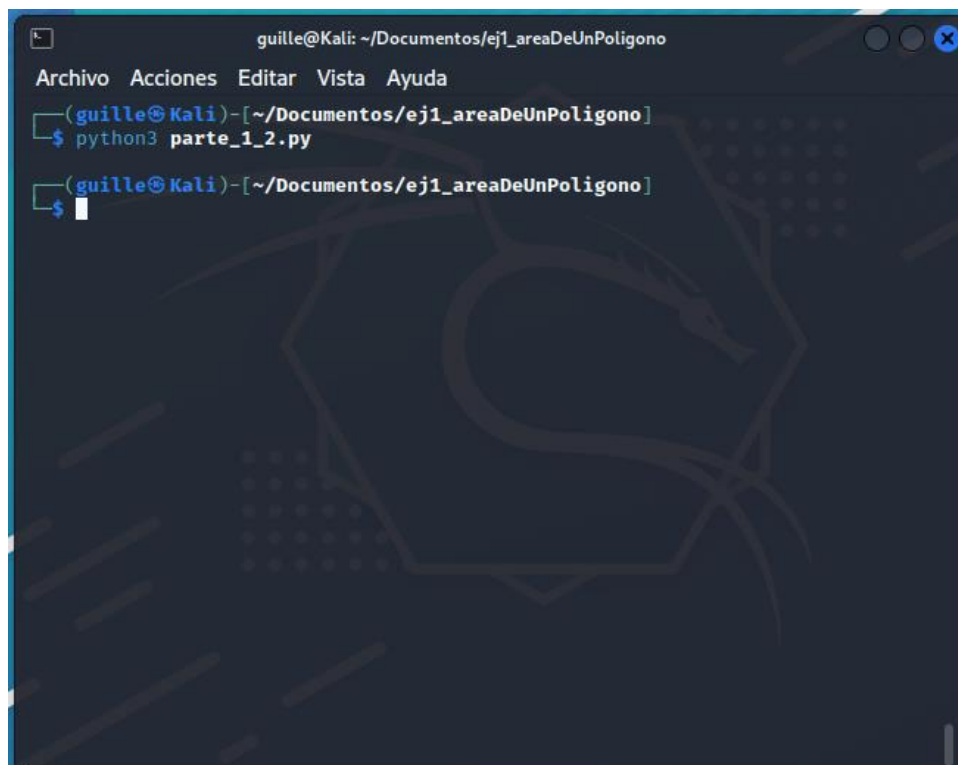
## PARTE 1: ENTORNO DE TRABAJO

### 1.1



```
guille@Kali: ~  
Archivo Acciones Editar Vista Ayuda  
(guille@Kali)-[~]  
$ python3  
Python 3.10.5 (main, Jun  8 2022, 09:26:22) [GCC 11.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> "Hola Algoritmos Y Programacion 1"  
'Hola Algoritmos Y Programacion 1'  
>>>  
>>> 
```

### 1.2



```
guille@Kali: ~/Documentos/ej1_areaDeUnPoligono  
Archivo Acciones Editar Vista Ayuda  
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]  
$ python3 parte_1_2.py  
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]  
$ 
```

## 1.5

- ¿Qué función debo usar para conseguir el mismo resultado que en la parte 1?1?

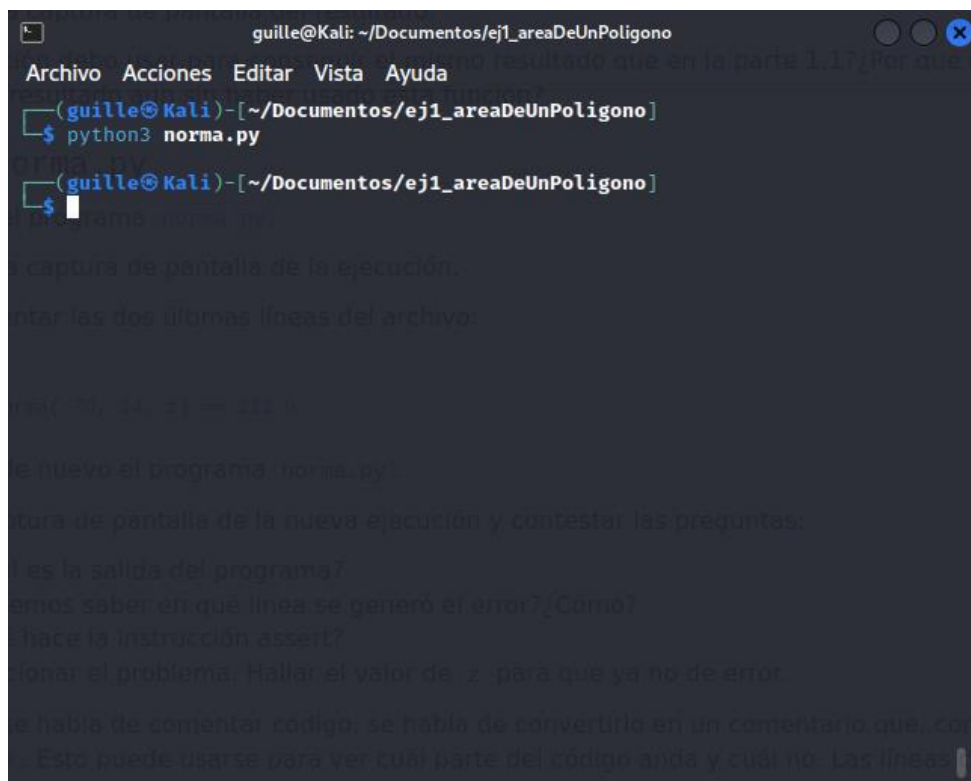
**Rta:** Para obtener el mismo resultado que el ítem 1.1 lo que debería hacer es imprimirlo con la función print, quedándose de la siguiente manera: print ("Hola Algoritmos y Programación I")

- ¿Por qué en la parte 1?1 vemos el resultado aun sin haber usado esta función?

**Rta:** Porque estamos en el intérprete de Python, el mismo ejecuta las instrucciones directamente.

## PARTE 2: NORMA.PY

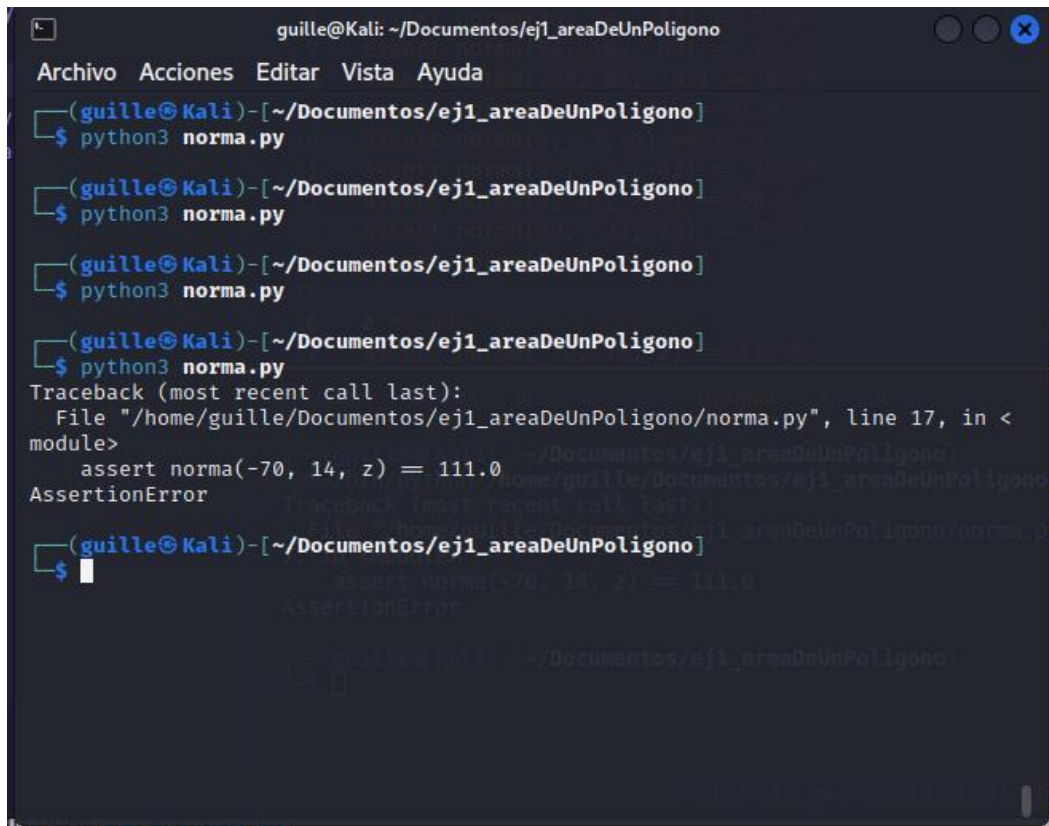
### 2.2



```
guille@Kali: ~/Documentos/ej1_areaDeUnPoligono
Archivo Acciones Editar Vista Ayuda
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]
$ python3 norma.py
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]
$
```

### 2.5

- Tomar captura de pantalla de la nueva ejecución y contestar las preguntas:



```
guille@Kali: ~/Documentos/ej1_areaDeUnPoligono
Archivo Acciones Editar Vista Ayuda
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]
$ python3 norma.py
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]
$ python3 norma.py
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]
$ python3 norma.py
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]
$ python3 norma.py
Traceback (most recent call last):
  File "/home/guille/Documentos/ej1_areaDeUnPoligono/norma.py", line 17, in <
module>
    assert norma(-70, 14, z) == 111.0
AssertionError
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]
$
```

- ¿Cuál es la salida del programa?

**Rta:** La salida del programa es un ERROR, el cual me está diciendo mediante la función Assert que la norma obtenida con el valor de z dado, no es el esperado, se evaluó un false.

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

**Rta:** Si podemos saber en qué línea se produjo el error, la cual es la 17 y la misma se nos muestra en la terminal junto al error.

- ¿Qué hace la instrucción Assert?

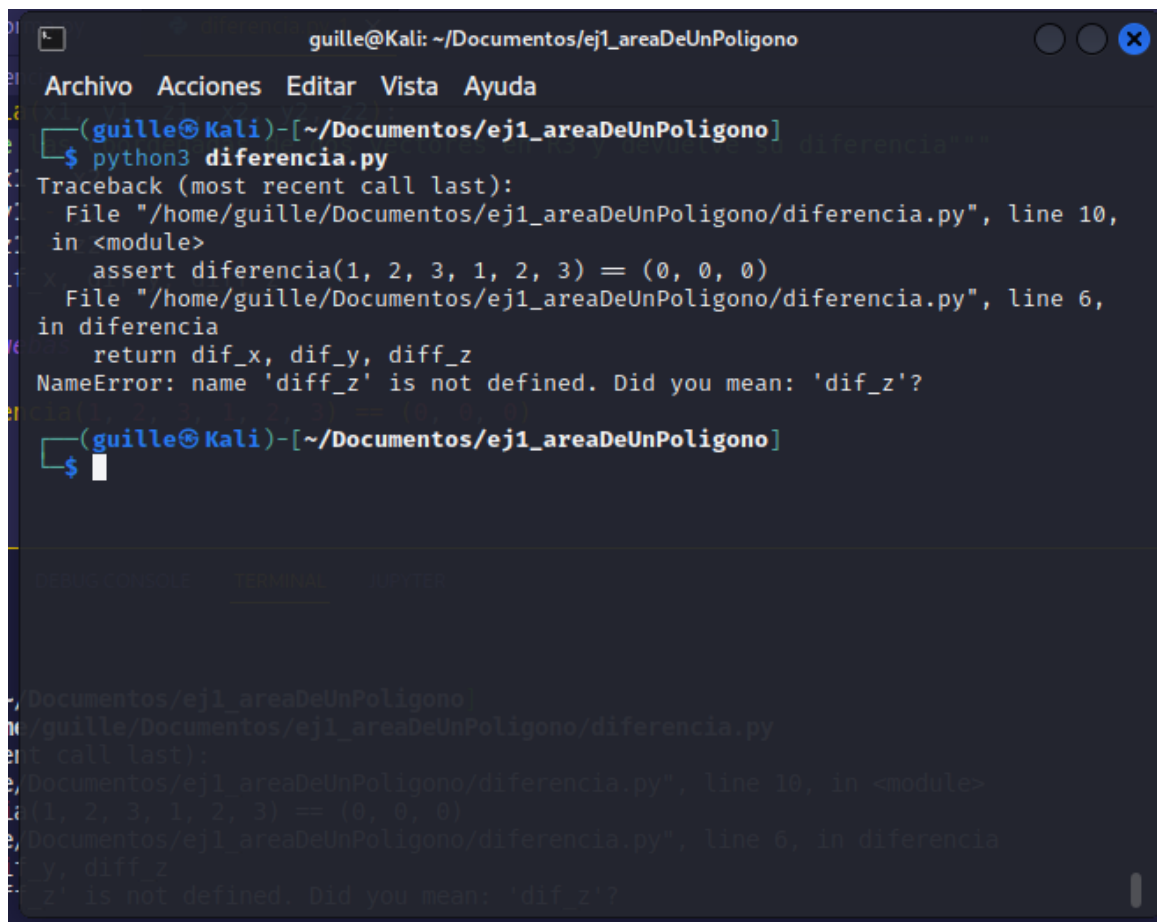
**Rta:** La instrucción Assert recibe una instrucción a verificar (de tipo booleana), si la condición es true, la instrucción no hace nada en caso contrario se produce un error.

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

**Rta:** El valor de z para que el resultado de la norma sea 111.0 es  $|z|=85$ .

## PARTE 3: DIFERENCIA.PY

### 3.3



```
guille@Kali: ~/Documentos/ej1_areaDeUnPoligono
Archivo Acciones Editar Vista Ayuda
(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]
$ python3 diferencia.py
Traceback (most recent call last):
  File "/home/guille/Documentos/ej1_areaDeUnPoligono/diferencia.py", line 10,
in <module>
    assert diferencia(1, 2, 3, 1, 2, 3) == (0, 0, 0)
  File "/home/guille/Documentos/ej1_areaDeUnPoligono/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'?

(guille@Kali)-[~/Documentos/ej1_areaDeUnPoligono]
$
```

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

A simple vista ya se ve el error, el cual también no los dice el error en la terminal, el mismo se produce al escribir incorrectamente la variable `diff_z` en vez de la correcta asignada arriba `dif_z`.

En mi caso en la línea 6 me tiraba un error de `nameError` ya que estaba mal escrita una variable, y me sugiere una posible solución

## PARTE 4: DEPURACIÓN

### 4.4

- ¿Qué error muestra? ¿En qué línea?

El error producido es el mismo que el ítem 2.5.1 un error del tipo `AssertError`, en la línea 10, luego también podemos ver que el cálculo del producto de la segunda variable que corresponde al versor `k` podemos ver que aparece un `**` en vez de un `*` cosas totalmente distintas



```

guille@Kali: ~/Documentos/ej1_areaDeUnPoligono
Archivo Acciones Editar Vista Ayuda
def calculoProductoVectorial(x1, y1, z1, x2, y2, z2):
    """Recibe las coordenadas de dos vectores en R3 y devuelve el producto vectorial"""
    versori = y1*z2 - z1*y2
    versorj = z1*x2 - x1*z2
    versork = x1*y2 - y1*x2
    return versori, versorj, versork

assert calculoProductoVectorial(54, 12, 29, 1, 11, 12) == (-175, -619, 582)
assert calculoProductoVectorial(71, 52, 24, 1, 11, 6) == (48, -402, 729)
assert calculoProductoVectorial(726, 434, 110, 488, 962, 820) == (250060, -541640, 486620)
assert calculoProductoVectorial(62, 12, 198, 380, 334, 490) == (-60252, 44860, 16148)
assert calculoProductoVectorial(-85, 807, 964, 462, 101, 474) == (285154, 485658, -381419)
assert calculoProductoVectorial(746, 466, 396, 910, 138, 289) == (80026, 144766, -321112)
assert calculoProductoVectorial(192, 362, 397, 249, 598, 50) == (-219306, 89253, 24678)
assert calculoProductoVectorial(781, 520, 996, 348, 68, 215) == (44072, 178693, -127852)
assert calculoProductoVectorial(459, 971, 201, 582, 569, 703) == (568244, -205695, -303951)
assert calculoProductoVectorial(754, 968, 956, 231, 901, -31) == (-891364, 244210, 455746)
"prodvectorial.py" 17L, 1131B
1,1 Todo

```

➤ ¿Se puede escribir *el cuerpo* de la función en una línea? ¿Cómo? Si se puede g

```

guille@Kali: ~/Documentos/ej1_areaDeUnPoligono
Archivo Acciones Editar Vista Ayuda
def calculoProductoVectorial(x1, y1, z1, x2, y2, z2):
    """Recibe las coordenadas de dos vectores en R3 y devuelve el producto vectorial"""
    return y1*z2 - z1*y2, z1*x2 - x1*z2, x1*y2 - y1*x2

assert calculoProductoVectorial(54, 12, 29, 1, 11, 12) == (-175, -619, 582)
assert calculoProductoVectorial(71, 52, 24, 1, 11, 6) == (48, -402, 729)
assert calculoProductoVectorial(726, 434, 110, 488, 962, 820) == (250060, -541640, 486620)
assert calculoProductoVectorial(62, 12, 198, 380, 334, 490) == (-60252, 44860, 16148)
assert calculoProductoVectorial(-85, 807, 964, 462, 101, 474) == (285154, 485658, -381419)
assert calculoProductoVectorial(746, 466, 396, 910, 138, 289) == (80026, 144766, -321112)
assert calculoProductoVectorial(192, 362, 397, 249, 598, 50) == (-219306, 89253, 24678)
assert calculoProductoVectorial(781, 520, 996, 348, 68, 215) == (44072, 178693, -127852)
assert calculoProductoVectorial(459, 971, 201, 582, 569, 703) == (568244, -205695, -303951)
assert calculoProductoVectorial(754, 968, 956, 231, 901, -31) == (-891364, 244210, 455746)
~
~
~
-- VISUAL --
51 3,55 Todo

```

## PARTE 5: REUTILIZANDO FUNCIONES

### 5.4

- ¿Cuál es la importancia de reutilizar funciones?

**Rta:** La importancia de reutilizar códigos son varias, a continuación, voy a decir algunas, una es que escribís menos y reducís el tiempo de desarrollo, tu código no queda tan largo y queda más prolijo y entendible. También facilita la documentación

En conclusión, siempre es recomendable que seas un desarrollador low-cost y que escribas lo justo y necesario y que reutilices lo más que puedas.