

Algoritmos y Programación I, curso 04,
Essaya

Informe del Ejercicio 1 Área de Polígonos

Alumno: Gianni Antonio
Boccazzi

Práctica: Barbara

Padrón: 109304

Ayudante a cargo: Florencia
Sardella

Parte 1.1

1. Ingresar al intérprete de Python con el comando:

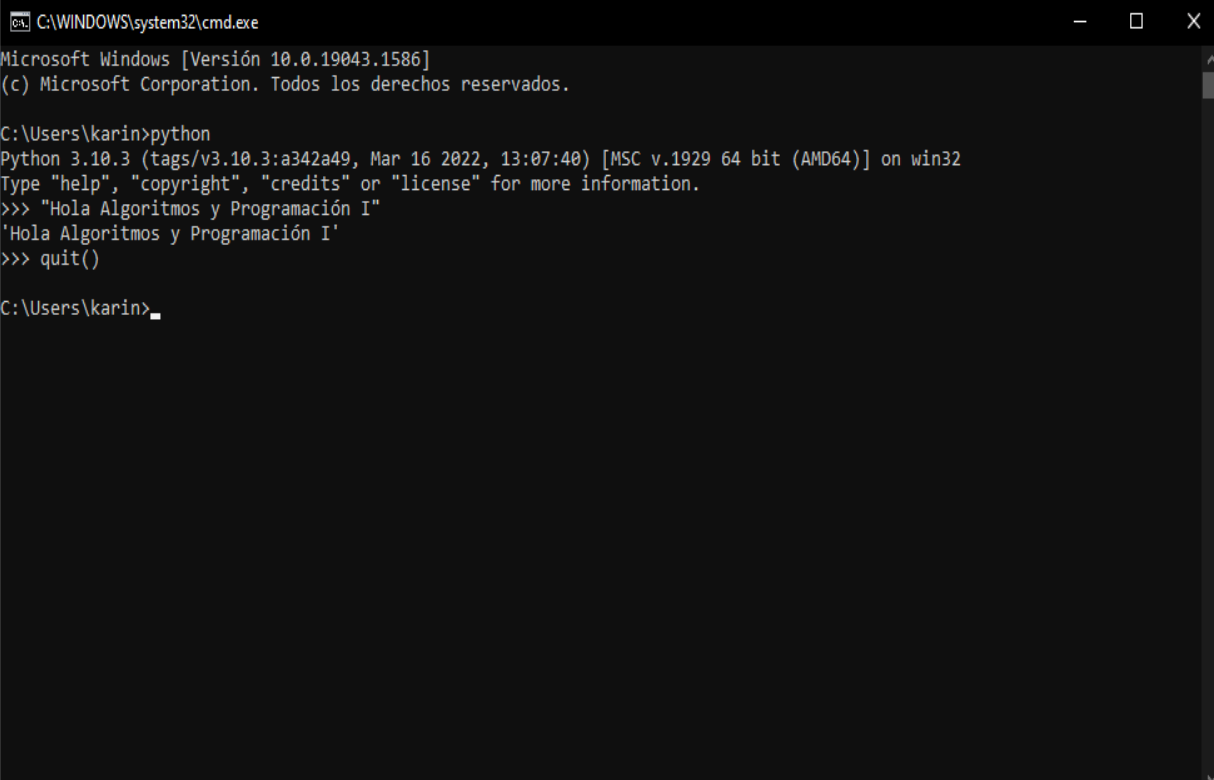
2. `$ python3`

3. Ejecutar la siguiente línea de código:

4. `>>> "Hola Algoritmos y Programación I"`

5. Salir del intérprete.

6. Tomar una captura de pantalla del resultado.



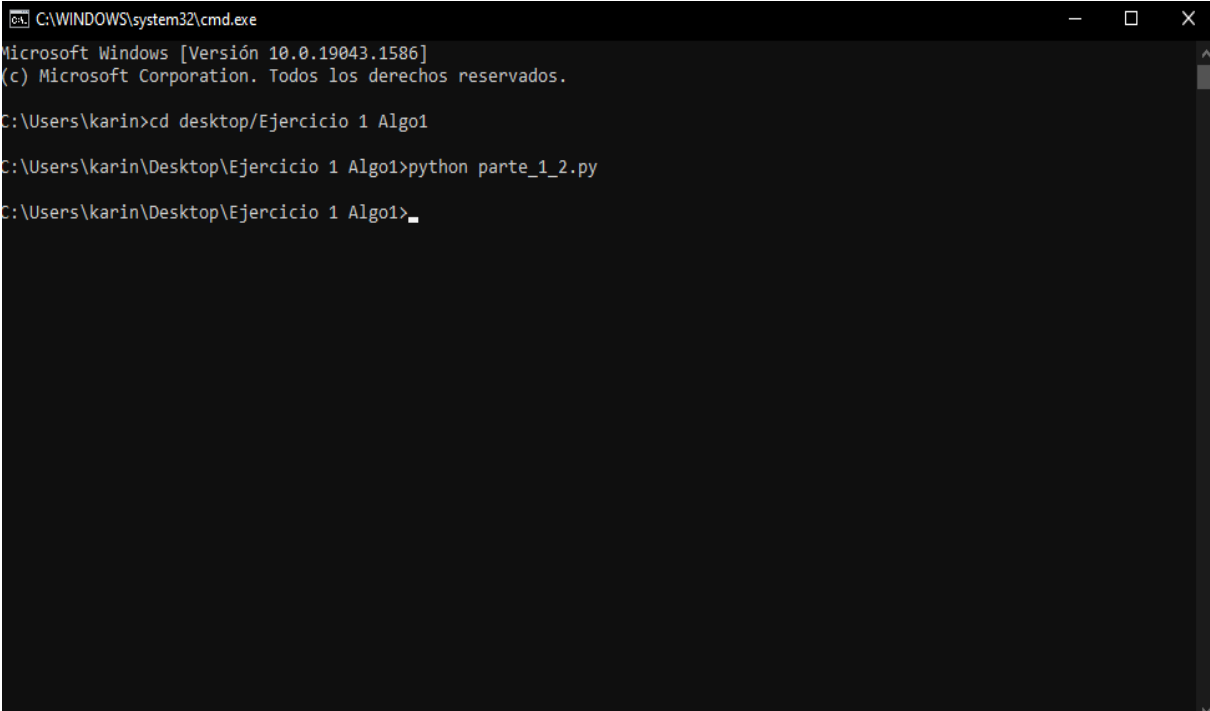
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1586]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\karin>python
Python 3.10.3 (tags/v3.10.3:a342a49, Mar 16 2022, 13:07:40) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> "Hola Algoritmos y Programación I"
'Hola Algoritmos y Programación I'
>>> quit()

C:\Users\karin>
```

Parte 1.2

1. Crear un archivo de texto plano llamado `parte_1_2.py`.
2. Agregar la siguiente línea al archivo:
3. "Hola Algoritmos y Programación I"
4. Ejecutar el programa con el comando:
5. `$ python3 parte_1_2.py`
6. Tomar una captura de pantalla del resultado.
7. ¿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?



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1586]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\karin>cd desktop/Ejercicio 1 Algo1

C:\Users\karin\Desktop\Ejercicio 1 Algo1>python parte_1_2.py

C:\Users\karin\Desktop\Ejercicio 1 Algo1>_
```

7. Para conseguir el mismo resultado que la parte 1.1, debemos usar la función `Print`, que imprime en la pantalla de la terminal texto o variables. En la Parte 1.1 se pudo ver el resultado de la función porque estábamos en el modo Interactivo de Python. Este modo nos sirve para escribir expresiones en el lenguaje Python, y Python nos devuelve el resultado inmediato, lo cual lo convierte en interactivo.

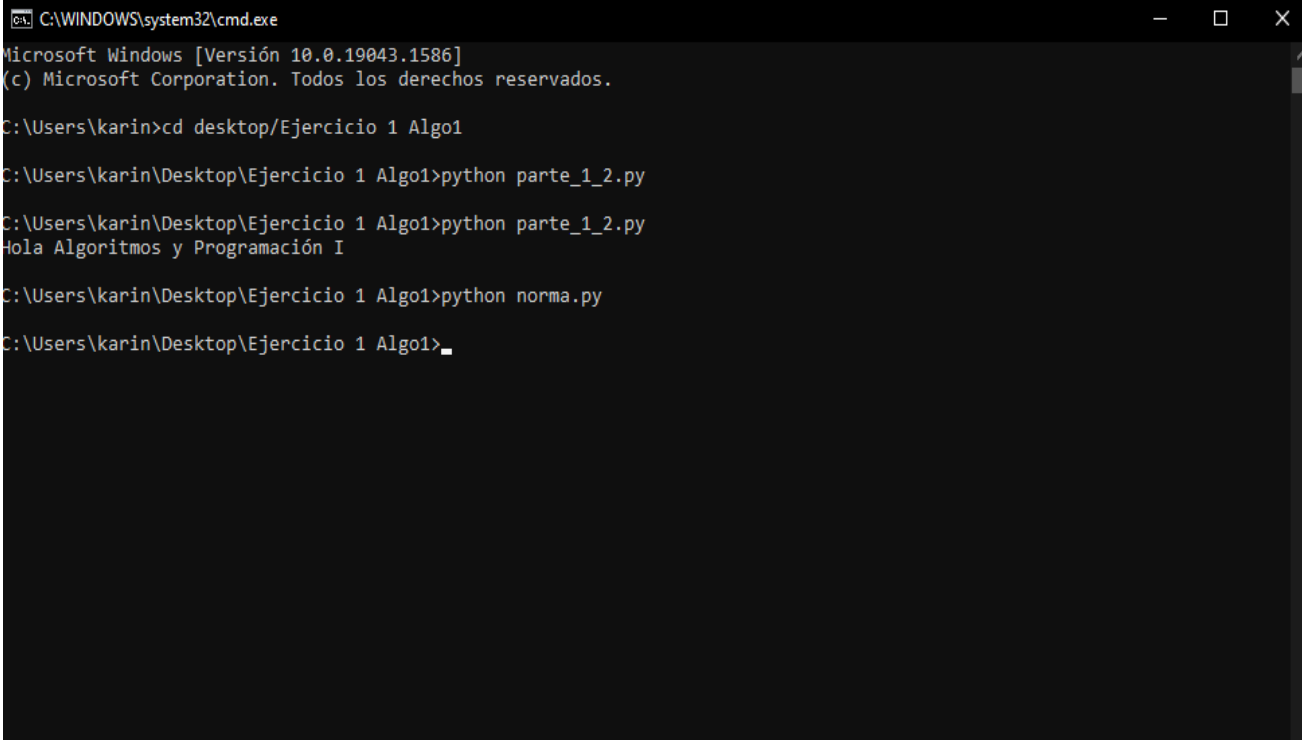
Parte 2: `norma.py`

1. Ejecutar el programa `norma.py`.
2. Tomar una captura de pantalla de la ejecución.
3. Des-comentar las dos últimas líneas del archivo:

```
z = -80  
  
assert norma(-70, 14, z) == 111.0
```

4. Ejecutar de nuevo el programa `norma.py`.
5. Tomar captura de pantalla de la nueva ejecución y contestar las preguntas:
 - a. ¿Cuál es la salida del programa?
 - b. ¿Podemos saber en qué línea se generó el error? ¿Cómo?
 - c. ¿Qué hace la instrucción `assert`?
 - d. Solucionar el problema. Hallar el valor de `z` para que ya no de error.

2.



```
C:\WINDOWS\system32\cmd.exe  
Microsoft Windows [Versión 10.0.19043.1586]  
(c) Microsoft Corporation. Todos los derechos reservados.  
  
C:\Users\karin>cd desktop/Ejercicio 1 Algo1  
  
C:\Users\karin\Desktop\Ejercicio 1 Algo1>python parte_1_2.py  
  
C:\Users\karin\Desktop\Ejercicio 1 Algo1>python parte_1_2.py  
Hola Algoritmos y Programación I  
  
C:\Users\karin\Desktop\Ejercicio 1 Algo1>python norma.py  
  
C:\Users\karin\Desktop\Ejercicio 1 Algo1>_
```

5.

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19043.1586]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\karin>cd desktop/Ejercicio 1 Algo1

C:\Users\karin\Desktop\Ejercicio 1 Algo1>python norma.py
Traceback (most recent call last):
  File "C:\Users\karin\Desktop\Ejercicio 1 Algo1\norma.py", line 17, in <module>
    assert norma(-70, 14, z) == 111.0
AssertionError

C:\Users\karin\Desktop\Ejercicio 1 Algo1>
```

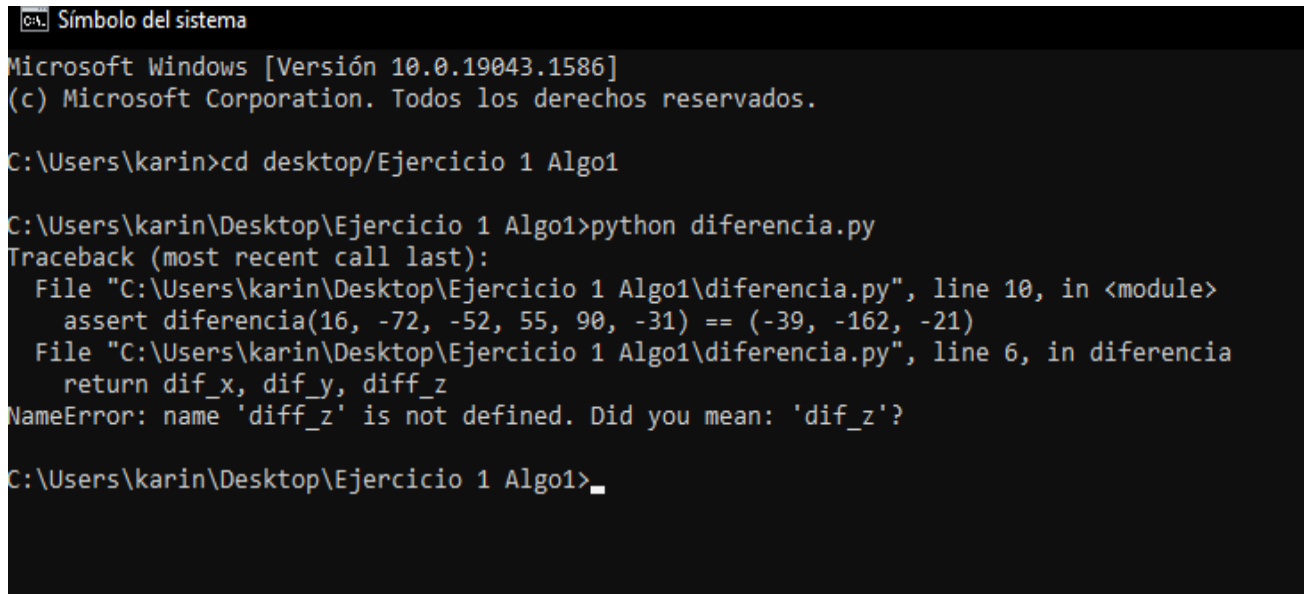
La salida del programa es un Traceback (un reporte, o rastreo) de un error del programa. En este Traceback que hace Python, se muestra dónde está el error y qué tipo de error es. En este caso, el error está en la línea 17, y es un AssertionError. Esto se debe a que la instrucción Assert nos permite comprobar si una condición del código que escribimos es cierta. En caso de serla, el controlador sigue por la siguiente línea, tal como pasó hasta la línea 14. Cuando la condición es falsa, el programa se frena y devuelve el AssertionError. Para que se solucione el error, la condición tiene que ser cierta, es decir que $z=-80$ no es correcto para que el resultado de 111.0. Entonces, haciendo los cálculos correctos vemos que $z=85$ o $z=-85$, y de esa forma se soluciona el AssertionError.

Parte 3: `diferencia.py`

1. Abrir el archivo `diferencia.py`
2. Utilizando la instrucción `assert`, probar los siguientes casos:
 - o La diferencia de los vectores $A = (16, -72, -52)$ y $B = (55, 90, -31)$ es $(-39, -162, -21)$
 - o La diferencia de los vectores $A = (55, -88, -75)$ y $B = (38, 62, -12)$ es $(17, -150, -63)$
3. Tomar captura de pantalla de la terminal mostrando su ejecución.

4. ¿Se detectó algún error? ¿Cuál era? ¿Qué significa? ¿Qué línea estaba fallando?
5. Solucionar el problema.

3.



```
C:\Users\karin\Desktop\Ejercicio 1 Algo1>python diferencia.py
Traceback (most recent call last):
  File "C:\Users\karin\Desktop\Ejercicio 1 Algo1\diferencia.py", line 10, in <module>
    assert diferencia(16, -72, -52, 55, 90, -31) == (-39, -162, -21)
  File "C:\Users\karin\Desktop\Ejercicio 1 Algo1\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'?

C:\Users\karin\Desktop\Ejercicio 1 Algo1>_
```

4. En este caso, se detecta un NameError (error de nombre), en la línea 6, diciendo que “diff_z” no está definido. Esto se debe a que en esa línea se utiliza el comando return a la variable “diff_z”, la cual, si prestamos atención, no está definida en ninguna línea. Pero sí está definida “dif_z”, que nos hace suponer que fue un error al tipear. Incluso, el Traceback de Python nos ayuda a solucionar el error, preguntando si en “diff_z” nos referíamos a “dif_z”. Para solucionarlo, simplemente borramos la “f” de más para que sea la misma variable que la línea 5.

Parte 4: Depuración

1. Leer el ejercicio 3.4.c de la guía de ejercicios.
2. Abrir el archivo `prodvect.py`, que corresponde a una solución del ejercicio 3.4.c
3. Ejecutar el programa.
4. ¿Qué error muestra? ¿En qué línea?
5. Depurar el programa.

o Ayuda: se pueden incluir llamadas a `print()` donde sea necesario. Por ejemplo:

o `print("DEBUG=== VALOR DE X: ", x)`

6. Renombrar la función y las variables de forma que sus nombres sean representativos. ¿Por qué es importante hacer esto?
 7. ¿Se puede escribir *el cuerpo* de la función en una línea? ¿Cómo?
4. Muestra un AssertionError en la línea 10. Imprimiendo las variables “var1, var2, var3” en la función definida, podemos ver que en “var2” da un número enorme. Al ver atentamente código, se puede ver que hay un error de tipo en la variable ya que hay doble “*”, lo cual no es una multiplicación, sino la potencia. Borrando una de las “*” se soluciona el problema.
6. Renomé la función a producto_vectorial, y las variables las cambié a “resultado” y la coordenada. Es importante que los nombres sean representativos porque esto nos ayuda a recordar y entender fácilmente lo que estamos haciendo, u otra persona está haciendo. Si no usamos nombres representativos, probablemente en algún momento no vamos a recordar para qué sirve cierta función/variable, y vamos a perder tiempo intentando averiguarlo.
7. Sí, en este caso la función va a recibir ciertas variables y va a devolver solamente tres operaciones entre ellas mismas, por lo cual no es necesario nombrar una variable “resultado”, para cada una de las operaciones. Con simplemente usar “return” con las operaciones, logramos escribir el cuerpo de la función en una línea.

Parte 5: Reutilizando funciones

1. Tomar todas las funciones anteriores y ponerlas en un archivo vectores.py.
2. Crear el archivo area_triángulo.py y, en él, solucionar el ejercicio 3.4.d de la guía de ejercicios **utilizando las funciones del módulo vectores.py**.
3. Crear al menos 3 pruebas que demuestren su uso.
4. ¿Cuál es la importancia de reutilizar funciones?

4. Es importante reutilizar funciones porque nos permite ahorrarnos tiempo en escribir funciones que anteriormente ya determinamos, también nos ayuda a que el código sea más conciso, ya que muchas veces los programas son largos y pueden confundirnos, e incluso al usar funciones anteriores, va a ser más fácil depurar el programa.