1. Otro concepto muy importante en Python es el de *listas*. Las listas son similares a las cadenas, excepto que cada elemento puede ser de un tipo diferente. La sintaxis para crear listas en Python es [..., ..., ...]. Por ejemplo, ejecute:

```
lista = [1, "hola", 1.0, 1-1j, True]
type(lista)
print(lista)
```

   Como puede ver, la variable `lista` es un nuevo tipo de objeto: 'list'. En este caso, es una lista cuyos elementos son un entero, un string, un float, un complejo, y un booleano. Para verificar esto, imprima el valor y el tipo de cada elemento de la lista. Por ejemplo,

```
print(lista[0],type(lista[0]))
print(lista[1],type(lista[1]))
```

   Este ejemplo también muestra que los índices de cada elemento de la lista son numerados de la misma manera que en un string:

```
print(lista[0:3])
print(lista[::2])
```

2. Los elementos de una lista pueden tener cualquier tipo reconocido por Python, por ejemplo, pueden ser otra lista!:

```
superlista = ["cool",lista]
print(superlista)
```

   Imprima el valor y el tipo de cada elementos de esta lista. ¿Cuántos elementos tiene la lista `superlista`? (respuesta, use la función `len()`).

3. Existen diversas funciones en Python que crean listas. La función `list()` crea una lista, por ejemplo, a partir de un string. Usando el string `x` definido anteriormente, ejecute

```
y = list(x)
print(y)
print(type(y))
```

4. Otra función que crea listas útiles, esta vez de números *enteros*, es `range(inicio,fin,paso)`, que crea una lista de valores desde `inicio` (cerrado) hasta `fin` (abierto!!), con paso `paso`. Ejecute,

```
z = range(2,26,3)
print(z)
```

   Nuevamente, `(inicio,fin,paso)` funcionan de forma similar a los índices de un string o una lista[1]

```
print(range(2,26))
print(range(26,2,-1))
```

   Bonus track: ¿Qué hacen los siguientes comandos?, ¿Modifican el valor de `x` y/o `lista`?

[1] En Python 3, para imprimir la lista generada por `range` es necesario agregar el comando `list`, por ejemplo `print(list(range(2,26)))`

```
x.split(" ")
x.split("e")
lista.append("chao")
lista.insert(2,"cool")
```

5. Otro concepto fundamental (en todo lenguaje de programación) es el de *ciclos*, es decir, comandos que definen tareas que se repiten un cierto número de veces. En Python, una forma simple de definir un ciclo es usando el comando `for`, cuya sintaxis general es de la forma:

```
for variable in lista:
        comando1
        comando2
comando_fuera_del_ciclo
```

   Un código de este tipo repite los comandos "indentados", es decir, escritos más a la derecha que los anteriores ya sea pulsando la tecla TAB o bien con cuatro espacios (en este caso, `comando1` y `comando2`) tantas veces como elementos tenga la lista `lista`. La primera ocasión que se ejecutan estos comandos la variable `variable` toma el valor `lista[0]`, la segunda vez el valor `lista[1]`, etc., hasta la última repetición donde `variable` toma el valor correspondiente al último elemento de la lista `lista` (es decir, `lista[-1]`).

   Por ejemplo, el código siguiente:

```
for palabra in ["computación", "científica", "con", "Python", 2018]:
    print(palabra)
print("terminamos con el ciclo")
```

   imprime cada uno de los elementos de la lista `['computación', 'científica', 'con', 'Python', 2018]`. Luego de completar el ciclo, el programa imprime el string `'terminamos con el ciclo'`. Verifique lo anterior escribiendo este código en un archivo .py y ejecutándolo en la terminal.

6. Escriba ahora un programa almacenado en un archivo .py con el siguiente código:

```
for x in range(-5,5):
    print(x)
    print(x**2)
```

   Verifique que entiende qué tareas realiza este sencillo programa, y por qué.

7. Una partícula realiza un movimiento vertical bajo la influencia de la gravedad de modo que su altura $z(t)$ respecto al suelo es dado por la siguiente ecuación de la trayectoria,

$$z(t) = z_0 + v_0 t - \frac{1}{2} g t^2, \tag{1}$$

   con $g = 9.8$ m/s^2. Considere el caso en que $z_0 = 1$ m y $v_0 = 24$ m/s.

   Escriba un programa en Python que, usando un ciclo `for`, calcule e imprima el valor de la altura $z(t)$ para los siguientes valores de tiempo (en segundos): $t = 0, 0.1, 0.2, \ldots 5.0$ (51 valores distintos de tiempo).

8. Modifique el programa anterior, para que ahora éste pregunte al usuario los valores de z_0 y v_0. Para esto, use el comando `input` que aprendió en su trabajo con la guía 07.

9. Escriba un programa en Python (que use un ciclo `for`) que calcule e imprima la suma de los primeros 1000 números enteros, es decir, el valor de

$$1 + 2 + 3 + 4 + \cdots + 999 + 1000. \tag{2}$$