1. Abra una consola y ejecute el intérprete de Python (es decir, ejecute el comando `python`).
   (a) ¿Qué versión de Python está instalada?
   (b) Ejecute los siguientes comandos en forma consecutiva:

   (c) Ahora ejecute:

   ¿Qué tipo de variable es `sx`? Entonces, ¿qué hace la función `str()`?
   (d) Ahora ejecute:

   ¿Qué diferencia observa en el resultado?
   (e) Conozca la función `len()`, para esto ejecute:

   ¿Qué valor entrega la función `len()` aplicada a un string[1]?, ¿Qué tipo de variable suministra? (pruebe aplicándola a otros strings).

2. Existen diversas operaciones definidas entre distintos tipos de variables. Para aprender cómo funcionan algunas de ellas defina primero las siguientes variables y verifique su tipo:

   A continuación imprima el valor y el tipo del resultado de las siguientes operaciones: `a+b, a+d, a+e, b+c, b+d, b+e, f+e, e+f, a*b, a*d, a*e, b*c, b*d, c*e, e*f, a**b, a**d, a**e, b**c, e**a, e**b, e**f, a/b, a/d, a/e, b/c, b/d, b/e, c/b, d/a, d/b, e/a, e/b, e/f, a*g, b*g, not(g), g and False, g and True, g or False, g or True`. ¿Cuáles de estas operaciones no están definidas?¿Qué pasó en los casos `b/c` y `c/b`?. Busque en las referencias sugeridas la explicación de este comportamiento. También existen operaciones que transforman el tipo de variable. Por ejemplo, como continuación del ejercicio anterior, calcule y verifique el tipo de las siguientes operaciones: `int(a), float(b), d.real, d.imag, a==b, a>b`. Cree un programa Python llamado `cuadratica.py` e incluya como primeras líneas el siguiente código:

   ▪

   Este pequeño programa Python, al ser ejecutado con el comando `python test.py`, pregunta al usuario por los valores de las variables a, b y c, que son asignadas como valores decimales (float). Ahora modifique el programa para que además *calcule e imprima* las dos soluciones de la ecuación cuadrática, es decir, los valores

   $$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{1}$$

   Para calcular la raiz cuadrada involucrada eleve el valor correspondiente la potencia 0.5, es decir, use el hecho que $\sqrt{\alpha} = \alpha^{0.5}$.

[1] "String" es el nombre usado comúnmente para una *cadena* de caracteres alfanuméricos.

3. Los caracteres individuales que forman una cadena alfanumérica (string) pueden ser accesados usando el número del *índice* correspondiente. En Python **el valor de los índices siempre comienza en 0**, luego 1, 2, etc. Para ilustrar esto, abra un intérprete Python y ejecute los siguientes comandos:

4. Como comprobó anteriormente la función `len()` entrega el largo del string, es decir, el número de caracteres que contiene. Por lo tanto

   imprime el último caracter del string, cuyo índice es `len(x)-1`, debido que el índice del primer caracter es 0. El mismo resultado, puede ser conseguido usando

   Similarmente,

   imprime el penúltimo caracter, y así sucesivamente. Por ejemplo, ejecute

   para comprobar que se refieren al mismo caracter.

5. También es posible acceder a un subconjunto de caracteres del string usando, en nuestro caso, `x[inicio:fin:paso]`, donde `inicio` y `fin` son los índices de los caracteres iniciales y finales y `paso` es un entero que define el paso. Si `paso` no es ingresado, el intérprete considera que `paso = 1` . Por ejemplo, ejecute y verifique qué hacen los siguientes comandos

   Note que el caracter correspondiente al índice `fin` NO es desplegado. En lenguaje matemático podríamos decir que `x[inicio:fin]` suministra los caracteres de `x` con índices en el intervalo desde `inicio` *cerrado* hasta `fin` *abierto*.

6. Además, si no se especifica `inicio` se asume el valor 0 (inicio del string) y si no se especifica `fin` se asume el valor `len(x)` (fin del string). Verifique esto ejecutando:

7. ¿Qué hace cada uno de los siguientes comandos?, ¿Modifican el valor de `x`?

8. 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:

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,

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

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

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

10. 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

11. 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,

12. ¿Qué hacen los siguientes comandos?, ¿Modifican el valor de `x` y/o `lista`?

13. 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,

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

    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).

14. 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.

15. 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

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

16. Escriba un programa que al ejecutarlo pregunte al usuario un número e imprima su valor absoluto. Recuerde que el valor absoluto (o módulo) $|x|$ de un valor real x es definido por

$$|x| := \begin{cases} x, & \text{si } x \geq 0 \\ -x, & \text{si } x < 0 \end{cases}. \tag{4}$$

17. Usando lo que aprendió sobre el comando `if` y asociados, modifique el programa `cuadratica.py` que creó y que resuelve la ecuación cuadrática $ax^2 + bx + c = 0$, para que ahora el programa informe que existen dos soluciones reales, y las imprima, si el discriminante $b^2 - 4ac$ es positivo, o que informe que no existe solución real (si el discriminante es negativo), o bien que informe que existe sólo una solución real, y la imprima (si el discriminante es nulo).

18. Escribir un programa que pregunte al usuario el valor algún número natural e imprima todos los números primos que hay hasta ese número. Por ejemplo, si se ingresa el número 8, el programa debe imprimir los números 2, 3, 5 y 7.

19. Modifique el programa anterior para que ahora el cálculo de todos los números primos menores que un cierto valor `n` sean retornados como una lista al llamar a la función `primos(n)`.

20. El factorial de un número entero positivo n, denotado por $n!$ es definido por

$$n! = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n. \tag{5}$$

Por ejemplo, $3! = 1 \cdot 2 \cdot 3 = 6$ y $10! = 3628800$. Escriba un programa en Python que pregunte al usuario por el valor de n, y que calcule e imprima su factorial, es decir, $n!$.

21. Modifique ahora el código que acaba de crear para calcular el factorial de un número entero para que ahora su programa verifique, antes de calcular el factorial, que el número suministrado es realmente un entero positivo, y sólo calcule el factorial en ese caso, y que en caso contrario informe al usuario que el número ingresado no es apropiado.

22. Modifique el programa anterior para que ahora el cálculo del factorial de un cierto valor `n` sea retornado al llamar a la función `mifactorial(n)`.

23. Escriba un programa que use su nuevo función `mifactorial(n)` y evalúe e imprima (una aproximación de) el número π. Para esto, use la siguiente expresión en serie (desarrollada por Leonhard Euler),

$$\pi = \sum_{n=0}^{\infty} \frac{2^{n+1}(n!)^2}{(2n+1)!} = \left[2^1 \frac{(0!)^2}{1!} + 2^2 \frac{(1!)^2}{3!} + 2^3 \frac{(2!)^2}{5!} + 2^4 \frac{(3!)^2}{7!} + \cdots \right]. \tag{6}$$

Para esto, trunque la serie infinita para incluir sólo un número finito de términos. Lo anterior es un ejemplo de un método con el que se puede calcular el valor de π, con precisión cada vez mayor al agregar más y más términos.