Guía de Ejercicios 1: Tipos de datos, expresiones y asignaciones

Objetivos:

- Ejercitar el manejo de valores y expresiones de los tipos `int`, `float` y `str`.
- Entender cómo funcionan las conversiones de tipo, tanto implícitas como explícitas.
- Advertir sobre los errores de representación de números reales en la computadora.
- Introducir los conceptos de variable, asignación y espacio de memoria de un programa.

Ejercicio 1. Evaluación de expresiones.

(a) Para cada una de las siguientes expresiones, determinar cuál es su tipo y evaluarla a mano, realizando las conversiones de tipos que sean necesarias:

(I) `11 + 2`
(II) `11 / 2`
(III) `11 // 2`
(IV) `11 % 2`
(V) `1 / 0`
(VI) `1 % 0`
(VII) `11.0 + 2.0`
(VIII) `'ho' + 'la'`
(IX) `'ja' * 3`
(X) `len('hola') + 6`
(XI) `123 + '123'`
(XII) `123 + 123.0`
(XIII) `123 + int('123')`
(XIV) `str(123) + '123'`

(b) Revisar las respuestas del punto anterior, esta vez evaluando las expresiones en una consola `ipython` y averiguando su tipo con la operación `type()`.

(c) ¿Qué diferencia hay entre los valores y los tipos de las expresiones (I) y (VII)?

(d) Comparar el resultado de evaluar las expresiones (II) y (III). ¿Cuál símbolo corresponde a la división entera, y cuál a la división de `float`? ¿Qué rol tiene el símbolo `%` respecto de la división entera?

(e) ¿A qué se deben los errores de las expresiones (V), (VI) y (XI)?

(f) ¿Qué ocurrió exactamente al evaluar las expresiones (XII), (XIII) y (XIV)?

Ejercicio 2. Representación de números reales en punto flotante.

(a) Evaluar las siguientes expresiones, primero a mano y después en una consola `ipython`:

(I) `(10 / 3 - 3) * 3 - 1`
(II) `0.00000001 + 1000000000 - 1000000000`

(b) ¿Los resultados son los esperados? ¿A qué se debe esto?

(c) En (II), reemplazar el primer valor literal (`0.00000001`) por valores cada vez más grandes (ej.: (`0.0000001`, `0.000001`, ..., hasta `0.1`), reevaluar y estudiar los resultados.

Ejercicio 3. Teniendo en cuenta la existencia de errores de representación de números reales en punto flotante, ¿qué problema puede surgir al comparar con == dos valores de tipo `float`? ¿Cómo se puede solucionar? ¿Cómo podría afectar a las comparaciones `!=`, >= y <=?

Ejercicio 4. La operación `round()` redondea un número `float` al entero más cercano. Por ejemplo, `round(1.9)` devuelve 2. Evaluar `round(0.5)`, `round(1.5)`, `round(2.5)`, `round(3.5)`, etc. Buscar documentación de `round` y averiguar a qué se debe el comportamiento observado.

Ejercicio 5. A partir del string `'123.45'`, escribir expresiones que arrojen como resultado los siguientes valores:

(a) El float `123.45`.

(b) El entero `123`.

(c) El float `0.45`.

Ejercicio 6. Cadenas de caracteres: tipo `str`.

(a) Escribir el siguiente programa en un archivo `holamundo.py`:

```
1 print("hola mundo")
```

Ejecutar el programa y observar su salida, ya sea con el botón ▶ en Spyder, o bien con el comando "`python holamundo.py`" en la consola del sistema operativo (*Terminal*, *Command Prompt*, etc.).

(b) Reemplazar el código por el siguiente y observar qué sucede al ejecutarlo.

```
1 print("hola
2 mundo")
```

(c) Reemplazar el código por el siguiente y observar la salida de su ejecución.

```
1 print("hola\nmundo\nchau\tmundo")
```

(d) Reemplazar el texto del programa por el siguiente y observar la salida de su ejecución.

```
1 # Esto es un comentario.
2 print("""En Python se pueden
3 escribir strings en varias líneas
4 """)
```

¿Qué se puede concluir de los símbolos """ y # en Python?

Ejercicio 7. En el ejercicio anterior apareció el carácter \. Se conoce a la barra invertida como *carácter de escape*, y su función dentro de una cadena de caracteres es darle un significado especial al carácter que figura inmediatamente a continuación. Por ejemplo, \n se usa como carácter de nueva línea. Escribir las siguientes instrucciones en la consola interactiva `ipython`:

```
print("\"\n")
print("""Cuidado con las "comillas"."""")
print('Tengamos "mucho cuidado" por favor.')
```

Ejercicio 8. Importar la biblioteca `math` y escribir un programa para imprimir los valores de las constantes matemáticas π y e, redondeados al cuarto número decimal. Para ello, usar el operador módulo (`%`) de strings. Opcionalmente, conseguir el mismo resultado usando el método `str.format()`. Links útiles:

- `https://docs.python.org/3/library/math.html` (documentación de la biblioteca `math`)
- `https://www.geeksforgeeks.org/python-output-formatting/` (breve tutorial sobre formateo de strings)

Ejercicio 9. ¿Qué valor se imprime en la última instrucción del siguiente programa?

```
a:int = 1
b:int = a
a = 2
print(b)
```

¿Por qué no se imprime 2, si en la segunda instrucción indicamos que `b` valiera lo mismo que `a`?

Ejercicio 10. Escribir un programa en Python que intercambie el valor de dos variables `m` y `n` de tipo entero. Es decir, si al comenzar el programa `m` vale un valor m_0, y `n` vale n_0; al finalizar, `m` deberá valer n_0, mientras que `n` deberá valer m_0. Primero, programarlo usando una variable auxiliar; luego, sin usar variables auxiliares (difícil). (En ambos casos, no se permite usar la asignación simultánea de Python.)

Ejercicio 11. Completar el siguiente código, de manera que convierta una temperatura expresada en grados Fahrenheit a grados Celsius.

```
fahr:float = 80.0
cel:float = ...
print(...)
```

Cuando `fahr` se inicializa en 80.0, la salida por pantalla debe ser exactamente:

```
80.00 grados Fahrenheit equivalen a 26.67 grados Celsius.
```

Prestar especial atención al formato de los números. Este programa debe funcionar para cualquier valor inicial de `fahr`, no solo para 80. :-)