**Actividad 2.1**

**Operaciones con números**

**1.- Ejecutar en PyCharm** la sentencia print(123+45), y comprobar que el resultado es entero (no tiene punto decimal).

Repetir en cada uno de los siguientes, reemplazando + (adición) por el operador indicado y ver el tipo de resultado que se obtiene (si tiene o no coma decimal).

1. - (resta, es decir, calcular 123 - 45),
2. \* (multiplicación, en matemáticas «123 × 45»),
3. \*\* (exponenciación, en matemáticas « 12345»)
4. / (división),
5. // (división con cociente entero),
6. % (resto de la división con cociente entero, ¡no confundir con porcentaje!).

Observar que, excepto el caso de la división 123 / 45, todos los resultados son enteros (no tienen coma decimal).

**2.- Introducir en PyCharm** los apartados del ejercicio anterior considerando 12.3 y 4.5 (en vez de, respectivamente, 123 y 45), y ver si los resultados tienen o no coma decimal.

Observar que el resultado de 12.3 // 4.5 es 2.0 y no 2.

**3.- Introducir en PyCharm** las siguientes operaciones. ¿Qué pasa si ponemos cualquiera de las siguientes?.

*1.* 12 / 0 2. 34.5 / 0 3. 67 // 0

**4.-** ¿Cuánto es 00 (cero a la cero) según las matemáticas? ¿Y según Python?. **Ejecutar en PyCharm** la operación anterior.

**5.- Ejecutar en PyCharm** las operaciones siguientes. Si tenemos más de una operación, podemos agrupar con paréntesis como hacemos en matemáticas. Ejecutar las siguientes instrucciones, comprobando que los resultados son los esperados.

1. print(4-(3+2)) 2. print((4-3)+2)

* Cuando no usamos paréntesis, tenemos que tener cuidado con la *precedencia* (cuál se aplica primero) de los operadores. Por ejemplo, si ponemos 2 + 3 × 4, sabemos que tenemos que calcular primero 3 × 4 y a eso agregarle 2, o sea, « × » tiene mayor precedencia que « + ».
* No es claro cómo evaluar 2 − 3 + 4 − 5. Parte del problema es que el signo « − » se usa tanto para indicar la resta como para indicar el inverso aditivo de un número. También ayuda a la confusión el que la suma es conmutativa (2 + 3 = 3 + 2), pero la resta no (2 − 3 ≠ 3 − 2).
* Algo similar pasa con la expresión 3/4 × 5: no queda claro si nos referimos a (3/4) × 5 o a 3/(4 × 5).
* Ante la ausencia de paréntesis, Python evalúa expresiones como 2 - 3 + 4 - 5 y 3 / 4 \* 5 de izquierda a derecha, esto es, como si hiciera los reemplazos antes de la evaluación.

*a* − *b* → *a* + (−*b*) y *a/b* → *a* × *b*−1

**6.- Introducir en PyCharm** esta operación. -3\*\*-4, ¿es equivalente en matemáticas a (−3)−4 o a −3−4?

¿Cuáles son las precedencias de Python en este caso?

En el curso trataremos de evitar construcciones como la de los ejercicios anteriores, agregando paréntesis aunque sean redundantes.

**7.- Ejecutar en PyCharm** las siguientes operaciones. Desde las matemáticas, ¿es √3 entero?, ¿y √4? ¿De qué tipo son 3\*\*(1/2) y 4\*\*(1/2)? (Razonar las respuesta)

**8.-** En un triángulo rectángulo, un cateto mide 12 y el otro 5. **Calcular cuánto mide la hipotenusa usando Python.**

**9.-** Compré 4 botellas a 100,23€ cada una y 7 a 60,25€ cada una.

Resolver con lápiz y papel (o mentalmente) **y luego hacer las cuentas con Python:**

1. ¿Cuántas botellas compré?
2. ¿Cuánto gasté en total?
3. En promedio, ¿cuál fue el costo por botella?

**10.- Ejecutar en PyCharm y las siguientes funciones**. Además de las operaciones entre números, podemos usar algunas funciones como el valor absoluto de *x*, *|x|*, que se escribe abs(x) en Python, o el redondeo de decimal a entero, round(x), que nos da el entero más próximo a *x*.

1. **Ejecutar en PyCharm** absy luego repetir para round.
2. Resolver y analizar el resultado: (No hace falta comentar nada sino darse cuenta de cómo funcionan estas funciones en PyCharm)

1. print(abs(12)) 2. print(round(12.3)) 3. print(round(-5.67)) 4. print(abs(-3.21))

1. **Evaluar:**

1. abs 2. round

1. y observar que al poner una función (como abs) sin argumento, Python responde diciendo que es una función (en este caso, propia).
2. Python diferencia entre mayúsculas y minúsculas. Probar con los siguientes, viendo que da error:

1. print(Abs(2)) 2. print(ABS(2))

**11.-** roundadmite un argumento opcional indicando la cantidad de decimales deseados para el redondeo.

**Evaluar en PyCharm:**

1. 1. print(round(123.4567, 3)) 2. print(round(765.4321, 2))
2. Escribir el promedio calculado en el ejercicio 1.9.*c*) redondeado a céntimos.
3. El precio de cierto auto 0 km es de 23400€: expresarlo redondeado a unidades de 1000 (mil) y de 10000 (diez mil) usando round.

**12.-** Cuando *x*es un número, type(x)nos dice si *x* es entero (int) o decimal (float) según Python.

**Ver los resultados de las siguientes instrucciones ejecutándolas en PyCharm:**

*a*) print(type(12)) *b*) print(type(12.3))

*c*) print(round(12.3)) *d*) print(type(round(12.3)))

*e*) print(98 // 76)) *f*) print(type(98 // 76))

*g*) print(98.0 // 76.0) *h*) print(type(98.0 // 76.0)**)**

- intviene de integer, o entero. floatviene de floating point (*punto flotante*), el nombre de la codificación para números decimales se estudiarán más adelante.

Es posible pasar de uno a otro tipo de número usando int (para pasar de float a int) o float (para pasar de int a float). Claro que al pasar de decimal a entero perdemos los decimales después de la coma.

**13.-** Analizar los resultados en PyCharm ejecutando las instrucciones de: (Comentar por qué se producen)

*a*) print(int(12.3)) *b*) print(type(int(12.3)))

*c*) print(float(-45) ) *d*) print(type(float(-45)))

**14.-** Python puede trabajar con enteros de cualquier tamaño, pero los tamaños de los decimales es limitado.

1. **Calcular** print(12\*\*34) y luego print(12.0\*\*34), y observar que Python representa al decimal con una variante de *notacióncientífica*incorporando la letra e para indicar una potencia de 10:
2. 4.922235242952027e+36 ↔ 4.922235242952027 × 1036.
   1. *¡No debe confundirse este* e *con el número e* = 2.71828 . . .*!*
3. Del mismo modo, poner print(1 / 12\*\*34) y ver cómo se representa en Python.
4. Escribir en la notación de Python: 12.3 × 1045 y 12.3 × 10−45 (sin usar \* ni \*\* sino e), observando cómo los representa Python.
5. Poner print(123\*\*456) viendo que obtenemos un número muy grande.
6. Poner print(123.0\*\*456.0) viendo que obtenemos un error (el mensaje de error indica Result too largeo *resultado demasiado grande*.)

**Operaciones con cadenas**

1. Escribir por pantalla cada carácter de una cadena introducida por teclado.
2. Pide una cadena y un carácter por teclado y muestra cuantas veces aparece el carácter en la cadena.
3. Realizar un programa que dada una cadena de caracteres, genere otra cadena con todos sus caracteres en minúscula.
4. Realizar un programa que dada una cadena de caracteres, genere otra cadena con todos sus caracteres en mayúscula.
5. Realizar un programa que dada una cadena de caracteres, genere otra cadena con todos sus caracteres en capital.
6. Realizar un programa que cuente cuantas veces aparece un carácter en dicha cadena pasada por teclado.
7. Realizar un programa que compruebe si todos los caracteres de una cadena pasada por teclado son dígitos.
8. Realizar un programa que compruebe si todos los caracteres de una cadena pasada por teclado son números.
9. Realizar un programa que compruebe si todos los caracteres de una cadena pasada por teclado son números decimales.
10. Realizar un programa que compruebe si todos los caracteres son alfanuméricos de una cadena pasada por teclado.
11. Realizar un programa que compruebe si todos los caracteres son alfabéticos de una cadena pasada por teclado.
12. Realizar un programa que compruebe si todas las letras son minúsculas de una cadena pasada por teclado.
13. Realizar un programa que compruebe si todas las letras son Mayúsculas de una cadena pasada por teclado.
14. Realizar un programa que compruebe si la cadena contiene todos caracteres imprimibles de una cadena pasada por teclado.
15. Realizar un programa que compruebe si la cadena contiene solo espacios de una cadena pasada por teclado.
16. Realizar un programa que cambie las mayúsculas por minúsculas y viceversa de una cadena pasada por teclado.
17. Realizar un programa que cambie las vocales por números: a=1, e=2, i=3, o=4 y u=5, de una cadena pasada por teclado.
18. Pide una cadena y dos caracteres por teclado, sustituye la aparición del primer carácter en la cadena por el segundo carácter.
19. Realizar un programa que compruebe si una cadena **contiene** una subcadena. Las dos cadenas se introducen por teclado.
20. Realizar un programa que introduzca un guion por un espacio en blanco de una frase pasada por teclado.
21. Realizar un programa que separe en una lista una frase pasada por teclado por los espacios en blanco.
22. Unir la lista del ejercicio anterior con un guion alto