Boletín de ejercicios para Navidad

Programación — DAW

Ricardo Pérez López
IES Doñana

Curso 2021/2022

1. Problemas prácticos

La mayoría de los problemas siguientes se pueden realizar de dos formas:

- Mediante un programa que reciba los datos de entrada a través del teclado e imprima los resultados a la pantalla.
- Mediante un subprograma (en este caso, una función) que reciba los datos a través de sus parámetros y devolviendo los resultados como su valor de retorno.

---

1. Escribe un programa que salude al usuario con el mensaje «Hola. Encantado de conocerle.».
2. Escribe un programa que pida un número al usuario y a continuación lo muestre.
3. Escribe un programa que pida al usuario su edad y muestre la que tendrá el año que viene.
4. Escribe un programa que pida el año actual y el de nacimiento del usuario. Debe calcular su edad, suponiendo que en el año en curso el usuario ya ha cumplido años.
5. Escribe un programa que calcule la media aritmética de dos notas enteras. Hay que teneren cuenta que la media puede contener decimales.

6. Escribe un programa que calcule la longitud y el área de una circunferencia. Para ello, el usuario debe introducir el radio (que puede contener decimales).

   Recordemos:

$$longitud = 2\pi \cdot radio$$
$$área = \pi \cdot radio^2$$

7. Pide dos números al usuario: a y b. Deberá mostrar `True` si ambos números son iguales y `False` en caso contrario.
8. Escribe un programa que solicite al usuario su edad y le indique si es mayor de edad (mediante un literal booleano `True` o `False`).
9. Escribe un programa que solicite al usuario un número y le indique si es par (mediante un literal booleano `True` o `False`).
10. Escribir un programa que nos indique si podemos salir a la calle. Existen aspectos que influirán en esta decisión: si está lloviendo y si hemos terminado nuestras tareas. Solo podremos salir a la calle si no está lloviendo y hemos finalizado nuestras tareas. Existe una opción en la que, indistintamente de lo anterior, podremos salir a la calle: el hecho de que tengamos que ir a la biblioteca (para realizar algún trabajo, entregar un libro, etc.). Solicitar al usuario (mediante un booleano) si llueve, si ha finalizado las tareas y si necesita ir a la biblioteca. El programa deberá mostrar mediante un booleano (`True` o `False`) si es posible que se le otorgue permiso para ir a la calle.
11. Un frutero necesita calcular los beneficios anuales que obtiene de la venta de manzanas y peras. Por este motivo, es necesario diseñar un programa que solicite las ventas (en kilos) de cada semestre para cada fruta. El programa mostrará el importe total sabiendo que el precio del kilo de manzanas está fijado en 2.35 € y el kilo de peras en 1.95 €.
12. Escribe un programa que pida un número al usuario y muestre su valor absoluto.
13. Escribe un programa que solicite las notas del primer, segundo y tercer trimestre (notas enteras que se solicitarán al usuario). El programa debe mostrar la nota media del curso como se utiliza en el boletín de calificaciones (sólo la parte entera) y como se usa en el expediente académico (con decimales).
14. Escribir un programa que pida como entrada un número decimal y lo muestre redondeado al entero más próximo.
15. Un economista te ha encargado un programa para realizar cálculos con el IVA. El programa debe solicitar la base imponible y el IVA que se debe aplicar. Muestra en pantalla el importe correspondiente al IVA y al total.
16. Escribe un programa que tome como entrada un número entero e indique qué cantidad hay que sumarle para que el resultado sea múltiplo de 7. Un ejemplo:

    - A 2 hay que sumarle 5 para que el resultado ($2 + 5 = 7$) sea múltiplo de 7.

- A 13 hay que sumarle 1 para que el resultado (13 + 1 = 14) sea múltiplo de 7.

Si proporcionas el número 2 o el 13, la salida del programa debe ser 5 ó 1, respectivamente.

Indicación: Usar el operador módulo.

17. Modifica el ejercicio anterior para que, indicando dos números n y m, diga qué cantidad hay que sumarle a n para que sea múltiplo de m.

18. Escribe un programa que pida la base y la altura de un triángulo y muestre su área.

$$área = \frac{base \times altura}{2}$$

19. Dado el siguiente polinomio de segundo grado:

$$y = ax^2 + bx + c$$

escribe un programa que pida los coeficientes a, b y c, así como el valor de x, y calcule el valor correspondiente de y.

20. Escribe un programa que solicite al usuario que introduzca una cantidad de segundos. El programa deberá mostrar cuántas horas, minutos y segundos hay en el número de segundos introducidos por el usuario.

21. Solicita al usuario tres distancias:

    - La primera, medida en milímetros.
    - La segunda, medida en centímetros.
    - La última, medida en metros.

    Escribe un programa que muestre la suma (en centímetros) de las tres longitudes introducidas.

22. Un biólogo está realizando un estudio de distintas especies de invertebrados y necesita un programa que le ayude a contabilizar el número de patas que tienen en total todos los animales capturados durante una jornada de trabajo. Para ello, te ha solicitado que escribas un programa al que hay que proporcionar:

    - El número de hormigas capturadas (6 patas).
    - El número de arañas capturadas (8 patas).
    - El número de cochinillas capturadas (14 patas).

    El programa deberá mostrar el número total de patas.

23. Una empresa que gestiona un parque acuático te solicita un programa que les ayude a calcular el importe que hay que cobrar en la taquilla por la compra de una serie de entradas (cuyo número será introducido por el usuario). Existen dos tipos de entrada: infantil, que cuestan 15.50 €; y de adultos, que cuestan 20 €.

    En el caso de que el importe total sea igual o superior a 100 €, se aplicará automáticamente un bono descuento del 5 %.

24. Solicita al usuario un número real y calcula su raíz cuadrada. Realiza dos versiones del programa:

    - Importando la función *raíz cuadrada* del módulo correspondiente.
    - Importante el propio módulo.

25. La FILA (Federación Internacional de Lanzamiento de Algoritmo) realiza una competición donde cada participante escribe un algoritmo en un papel y lo lanza, ganando quien consiga lanzarlo más lejos. La peculiaridad del concurso es que la longitud del lanzamiento se mide en metros (con tantos decimales como se desee), pero para el ranking solo se tiene en cuenta la longitud en centímetros (sin decimales). Por ejemplo, para un lanzamiento de 12.3456 m (que son 1234.56 cm) solo se contabilizarán 1234 cm.

    Escribe un programa que solicite la longitud (en metros) de un lanzamiento, y muestre la parte entera correspondiente en centímetros.

26. Escribir un programa que solicite al usuario un número e indique si es par o impar.

27. Pedir dos números enteros y decir si son iguales o no.

28. Solicitar dos números distintos y mostrar cuál es el mayor.

29. Solicitar dos números (que pueden ser iguales o distintos) y mostrar cuál es el mayor.

30. Implementar un programa que pida por teclado un número decimal e indique si es un número casi-cero, que son aquellos números (positivos o negativos) que se acercan a cero por menos de 1 unidad (aunque, curiosamente, el 0 no se considera un número casi-cero). Ejemplos de números casi-cero son el 0.3, el -0.99 o el 0.123. En cambio, algunos números que no se consideran casi-cero son 12.3, el 0 o el -1.

31. Pedir dos números y mostrarlos ordenados en orden decreciente.

32. Pedir tres números y mostrarlos ordenados de mayor a menor.

33. Pedir los coeficientes de una ecuación de segundo grado y mostrar sus soluciones reales. Si no existen, habrá que indicarlo. Hay que tener en cuenta que las soluciones de una ecuación de segundo grado $ax^2 + bx + c = 0$ son:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

34. Escribir un programa que indique cuántas cifras tiene un número entero introducido por teclado, que estará comprendido entre 0 y 99 999.

35. Pedir una nota entera de 0 a 10 y mostrarla de la siguiente forma:

    - Insuficiente (de 0 a 4).
    - Suficiente (5).
    - Bien (6).
    - Notable (7 y 8).
    - Sobresaliente (9).
    - Matrícula de honor (10).

36. Escribir un programa que solicite al usuario un número comprendido entre 1 y 7, correspondiente a un día de la semana. Se debe mostrar el nombre del día de la semana al que corresponde. Por ejemplo, el número 1 corresponde a «lunes» y el 6 a «sábado».

37. Escribir un programa que solicite al usuario un número comprendido entre 1 y 7, correspondiente a un día de la semana. Se debe mostrar el nombre del día de la semana al que corresponde. Por ejemplo, el número 1 corresponde a «lunes» y el 6 a «sábado».

38. Pedir el día, mes y año de una fecha e indicar si la fecha es correcta. Hay que tener en cuenta que existen meses con 28, 30 y 31 días (no se considerará los años bisiestos).

39. Hacer el mismo ejercicio anterior pero considerando los años bisiestos.

    Un año es bisiesto si cumple las siguientes condiciones:

    - Es bisiesto si es divisible entre 4.
    - Pero no es bisiesto si es divisible entre 100.
    - Pero sí es bisiesto si es divisible entre 400. (2000 y 2400 sí son bisiestos porque son divisibles entre 100 pero también entre 400. 1900, 2100, 2200 y 2300 no lo son porque solo son divisibles entre 100).

40. Escribir un programa que solicite al usuario una fecha (día, mes y año) y muestre la fecha correspondiente al día siguiente.

41. Escribir un programa que pida una hora de la siguiente forma: hora, minutos y segundos. El programa debe mostrar qué hora será un segundo más tarde.

    Por ejemplo: hora actual: `10:41:59` → hora actual + 1 segundo: `10:42:00`.

42. Escribir un programa que muestre, para cada número introducido por teclado, si es par, si es positivo y su cuadrado. El proceso se repetirá hasta que se introduzca un 0.

43. Escribir un programa para calcular datos estadísticos de las edades de los alumnos de un centro educativo. Se introducirán datos hasta que uno de ellos sea negativo, y se mostrará: la suma de todas las edades introducidas, la media, el número de alumnos y cuántos son mayores de edad.

44. Codificar el juego «el número secreto», que consiste en acertar un número entre 1 y 100 (generado automáticamente). Para ello se introduce por teclado una serie de números, para los que se indica: «mayor» o «menor», según sea mayor o menor con respecto al número secreto. El proceso termina cuando el usuario acierta o cuando se rinde (introduciendo un -1).

45. Un centro de investigación de la flora urbana necesita una aplicación que muestre cuál es el árbol más alto. Para ello, se introducirá por teclado la altura (en centímetros) de cara árbol (terminando la introducción de datos cuando se utilice -1 como altura). Los árboles se identifican mediante etiquetas con números únicos correlativos, comenzando en 0. Escribir un programa que resuelva el problema planteado.

46. Desarrollar un juego que ayude a mejorar el cálculo mental de la suma. El jugador tendrá que introducir la solución de la suma de dos números aleatorios comprendidos entre 1 y 100. Mientras la solución introducida sea correcta, el juego continuará. En caso contrario, el programa terminará y mostrará el número de operaciones realizadas correctamente.

47. Escribir un programa para aprender a contar, que pedirá un número n y mostrará todos los números del 1 al n.

48. Escribir todos los múltiplos de 7 menores que 100.

49. Pedir diez números enteros por teclado y mostrar la media.

50. Implementar un programa que pida al usuario un número comprendido entre 1 y 10. Hay que mostrar la tabla de multiplicar de dicho número, asegurándose de que el número introducido se encuentra en el rango establecido.

51. Diseñar un programa que muestre las tablas de multiplicar del 1 al 10.

52. Diseñar un programa que muestre la suma de los 10 primeros números impares.

53. Pedir un número y calcular su factorial.

54. Pedir 5 calificaciones de alumnos y decir al final si hay algún suspenso.

55. Dadas 6 notas, escribir la cantidad de alumnos aprobados, condicionados (nota igual a 4) y suspensos.

56. Pedir por consola un número n y dibujar un triángulo rectángulo de n elementos de lado, utilizando para ello asteriscos ($*$). Por ejemplo, para $n = 4$:

```
* * * *
* * *
* *
*
```

57. Escribir un programa que convierta un número decimal en su representación binaria. Hay que tener en cuenta que desconocemos cuántas cifras tiene el número que introduce el usuario.

58. Escribir un programa que convierta un número binario en su representación decimal.

59. Escribir un programa que incremente la hora de un reloj. Se pedirán por teclado la hora, minutos y segundos, así como cuántos segundos se desea incrementar la hora introducida. El programa mostrará la nueva hora. Por ejemplo, si las `13:59:51` se incrementan en 10 segundos, resultan las `14:00:01`.

60. Escribe un programa que nos pida un número n y nos diga cuántos números primos hay entre 1 y n.

    Por ejemplo, para $n = 8$, comprobamos todos los números del 1 al 8: 1 → primo ; 2 → primo ; 3 → primo ; 4 → no primo ; 5 → primo ; 6 → no primo ; 7 → primo ; 8 → no primo

61. Diseña un programa que dibuje el triángulo de Pascal (también llamado de Tartaglia), para n filas. Numerando las filas y elementos desde 0, la fórmula para obtener el m-ésimo elemento de la n-ésima fila es:

$$E(n, m) = \frac{n!}{m!(n-m)!}$$

    Un ejemplo de triángulo de Pascal con 5 filas ($n = 4$) es:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

62. Solicita al usuario un número n y dibuja un triángulo de base y altura n, de la siguiente forma (para $n = 4$):

```
   *
  * *
 * * *
* * * *
```

63. Para dos números dados, a y b, es posible buscar el máximo común divisor (el número más grande que divide a ambos números) mediante un algoritmo ineficiente pero sencillo: desde el menor de a y b, ir buscando de forma decreciente el primer número que divide a ambos simultáneamente. Realiza un programa que calcule el máximo común divisor de dos números.

64. De forma similar al ejercicio anterior, implementa un algoritmo que calcule el mínimo común múltiplo de dos números dados.

65. Escribir una función que calcule el máximo común divisor de tres números (ver los dos ejercicios anteriores).

66. Calcula la raíz cuadrada de un número natural mediante aproximaciones. En el caso de que no sea exacta, muestra el resto. Por ejemplo, para calcular la raíz cuadrada de 23, probamos $1^2 = 1$, $2^4 = 4$, $3^2 = 9$, $4^2 = 16$, $5^2 = 25$ (nos pasamos), resultando 4 la raíz cuadrada de 23 con un resto $(23 - 16)$ de 7.

67. Escribe un programa que solicite al usuario las distintas cantidades de dinero de las que dispone. Por ejemplo, la cantidad de dinero que tiene en el banco, en una hucha, en un cajón y en los bolsillos. El programa mostrará la suma total de dinero de la que dispone el usuario. La manera de especificar que no se desea introducir más cantidades es mediante el cero.

68. Diseñar la función `eco` a la que se le pasa como argumento un número entero n y muestra por pantalla n veces el mensaje «`Eco...`».

69. Escribir una función a la que se le pasen dos enteros y muestre todos los números comprendidos entre ellos.

70. Escribir una función a la que se le pasen dos enteros y devuelva en una lista todos los números comprendidos entre ellos.

71. Escribir una función que calcule y devuelva el área o el volumen de un cilindro, según se especifique. Para distinguir un caso de otro, se le pasará como opción un número: 1 (para el área) o 2 (para el volumen). Además, hay que pasarle a la función el radio de la base y la altura.

$$\begin{aligned} área &= 2\pi \cdot radio \cdot (altura + radio) \\ volumen &= \pi \cdot radio^2 \cdot altura \end{aligned}$$

72. Diseñar una función que recibe como argumentos dos números enteros y devuelve el máximo de ambos.

73. Crear una función que, mediante un booleano, indique si el carácter que se le pasa como argumento corresponde a una vocal.

74. Implementar la función definida según la siguiente especificación:

$$\begin{cases} \textbf{Pre}: & n \geq 0 \\ & \texttt{es\_primo(}n\texttt{ : int) -> bool} \\ \textbf{Post}: & \texttt{es\_primo(}n\texttt{)} = \begin{cases} \texttt{True} & \text{si } n \text{ es primo} \\ \texttt{False} & \text{en caso contrario} \end{cases} \end{cases}$$

75. Escribir una función a la que se le pase un número entero y devuelva el número de divisores primos que tiene.

76. Diseñar la función `calculadora`, a la que se le pasan dos números reales (los operandos) y qué operación se desea realizar con ellos. Las operaciones disponibles son: sumar, restar, multiplicar o dividir. Las operaciones se especifican mediante un número: 1 para la suma, 2 para la resta, 3 para la multiplicación y 4 para la división. La función devolverá el resultado de la operación en forma de número real.

    Indicación: Los números se codifican las operaciones disponibles se pueden representar más adecuadamente usando *constantes*.

77. Diseñar una función que calcule y muestre la superficie y el volumen de una esfera.

$$superficie = 4\pi \cdot radio^2$$
$$volumen = \frac{4\pi}{3} \cdot radio^3$$

78. Implementar la función `distancia` que, con la siguiente especificación:

$$\begin{cases} \textbf{Pre}: & \texttt{True} \\ & \texttt{es\_primo(}x_1\texttt{ : float, }y_1\texttt{ : float, }x_2\texttt{ : float, }y_2\texttt{ : float) -> float} \\ \textbf{Post}: & \texttt{es\_primo(}x_1\texttt{, }y_1\texttt{, }x_2\texttt{, }y_2\texttt{)} = \text{la distancia euclídea de} \\ & \qquad\qquad\qquad\qquad\qquad \text{los puntos } (x_1, y_1) \text{ y } (x_2, y_2) \end{cases}$$

    La fórmula para calcular la distancia euclídea es:

$$distancia = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

79. Escribir la función `numeros_pares` que muestre por la consola los n primeros números pares.

80. Escribir una función a la que se le pase como argumentos una cantidad de días, horas y minutos. La función calculará y devolverá el número de segundos que existen en los datos de entrada.

81. Escribir la función `divisores_primos` que devuelva una lista con todos los divisores primeros del número entero que se le pasa como argumento.

82. Escribir una función que decida si dos números enteros positivos son *amigos*. Dos números a y b son amigos si la suma de los divisores propios (distintos de él mismo) de a es igual a b, y viceversa.

    Para probar, se pueden usar los números 220 y 284, que son amigos.

83. Crear una función que devuelva una lista de números aleatorios enteros. Los parámetros de la función serán: la cantidad de números aleatorios que se mostrarán y los valores mínimos y máximos que estos pueden tomar.

84. Crear tres listas de cinco elementos: la primera con enteros, la segunda con reales y la tercera con booleanos.

85. Crea una lista de 10 elementos del tipo que desees. Define diferentes variables que referencien a la lista creada. Comprueba que todas las variables contienen la misma referencia.

86. Crea una lista de longitud 10 que se inicializará con números aleatorios comprendidos entre 1 y 100. Mostrar la suma de todos los números aleatorios que se guardan en la lista.

87. Escribir un programa en el cual el usuario puede introducir por teclado un número n; a continuación, se solicita que teclee n números, y muestra la media de los números positivos, la media de los negativos y cuenta el número de ceros introducidos.

88. Crear un programa que solicite al usuario que introduzca por teclado 5 números decimales. A continuación, mostrar los números en el mismo orden en que se han introducido.

89. Escribir un programa que solicite al usuario cuántos números desea introducir. A continuación, el usuario podrá introducir esa cantidad de números enteros y, por último, el programa mostrará esos números en orden inverso al introducido.

90. Diseñar una función que devuelva el valor máximo contenido en una lista.

91. Escribir la función:

$$\begin{cases} \textbf{Pre}: & longitud \geq 0 \land fin \geq 2 \\ & \texttt{rellena\_pares(}longitud\texttt{ : int, }fin\texttt{ : int) -> List[int]} \end{cases}$$

    que crea y devuelve una lista ordenada de la longitud especificada, rellena con números pares aleatorios comprendidos en el rango desde 2 hasta fin (inclusive).

92. Escribir la función:

$$\begin{cases} \textbf{Pre}: & \texttt{True} \\ & \texttt{buscar(}lista\texttt{ : List[int], }clave\texttt{ : int) -> int} \end{cases}$$

    que busca de forma secuencial el valor *clave* en la lista *lista*. En caso de encontrarlo, devuelve en qué posición lo encuentra; y en caso contrario, devolverá -1.

93. Definir una función que tome como argumentos dos listas, la primera con los 6 números de una apuesta de la primitiva, y la segunda con los 6 números de la combinación ganadora. La función devolverá el número de aciertos.

94. Definir la función `sin_repetidos` que construye y devuelve una lista de longitud apropiada, con los elementos de la lista original donde se han eliminado los datos repetidos.

95. Leer y almacenar n números enteros en una lista, a partir de la que se construirán otras dos listas con los elementos con valores pares e impares de la primera, respectivamente. Las listas con valores pares e impares deberán mostrarse ordenadas.

96. Escribir la función

    Pre : `True`

    `eliminar_mayores(`*lista* `: List[int],` *valor* `: int) -> List[int]`

    que crea y devuelve una copia de la lista *lista* donde se han eliminado todos los elementos que son mayores que *valor*.

97. Crear una función que realice el borrado de un elemento de una lista ordenada, sin usar ningún método de borrado de listas como `remove`.

98. El «número de la suerte» de una persona se puede calcular a partir de sus números favoritos. De entre ellos, se seleccionan dos diferentes al azar, que se eliminarán de la lista, pero en su lugar se añade la media aritmética de los dos eliminados a la lista de números favoritos. El proceso se repite hasta que sólo quede un número, que resultará el número de la suerte para esa persona.

    Escribir un programa que solicite al usuario sus números favoritos y calcule su número de la suerte.

99. Desarrollar el juego «la cámara secreta», que consiste en abrir una cámara mediante su combinación secreta, que está formada por una combinación de dígitos del 1 al 5. El jugador especificará cuál es la longitud de la combinación; a mayor longitud, mayor será la dificultad del juego. La aplicación genera, de forma aleatoria, una combinación secreta que el usuario tendrá que acertar. En cada intento se muestra como pista, para cada dígito de la combinación introducido por el jugador, si es mayor, menor o igual que el correspondiente en la combinación secreta.

100. Crear una matriz bidimensional de tamaño 5×5 mediante una lista de listas y rellenarla de la siguiente forma: el elemento de la posición `[`n`][`m`]` debe contener el valor $10 \cdot n + m$. Después se debe mostrar su contenido.

101. Escribir la función:

    Pre : `True`

    `buscar_todos(`*lista* `: List[int],` *clave* `: int) -> List[int]`

    que crea y devuelve una lista con todos los índices de los elementos donde se encuentra la clave de búsqueda *clave*. En el caso de que *clave* no se encuentre en *lista*, la función devolverá una lista vacía.

102. El ayuntamiento de Sanlúcar te ha encargado una aplicación que ayude a realizar encuestas estadísticas para conocer el nivel adquisitivo de los sanluqueños. Para ello, tendrás que preguntar el sueldo a cada persona encuestada. *A priori*, no se conoce el número de encuestados. Para finalizar la entrada de datos, introduce un sueldo con valor -1.

    Una vez terminada la entrada de datos, muestra la siguiente información:

    - Todos los sueldos introducidos, ordenados de forma decreciente.
    - El sueldo máximo y mínimo.
    - La media de los sueldos.

103. Debes desarrollar una aplicación que ayude a gestionar las notas de un centro educativo. Los alumnos se organizan en grupos compuestos por 5 personas. Leer las noas (números enteros) del primer, segundo y tercer trimestres de un grupo. Debes mostrar al final la nota media del grupo en cada trimestre y la media del alumno que se encuentra en una posición dada (que el usuario introduce por teclado).

104. En un juego de rol, el mapa puede implementarse como una matriz donde las filas y las columnas representan lugares (*lugar 0*, *lugar 1*, *lugar 2*, etc.) que estarán conectados. Si desde el lugar x podemos ir hacia el lugar y, entonces la matriz en la posición `[x][y]` valdrá `True`; en caso contrario, valdrá `False`.

    Escribe una función que, dados dos lugares y una matriz que representa el mapa, indique si es posible viajar desde el primer lugar al segundo **directamente o pasando por lugares intermedios**.

105. Escribir la función:

$$\begin{cases} \textbf{Pre}: \texttt{True} \\ \qquad \texttt{suma(}\mathit{lista}\texttt{ : List[int], }\mathit{num\_elem}\texttt{ : int) -> List[int]} \end{cases}$$

    que crea y devuelve una lista con las sumas de los *num_elem* elementos consecutivos de *lista*.

    Por ejemplo, sea *lista* = `[10, 1, 5, 8, 9, 2]`. Si los elementos de *lista* se agrupan de 3 en 3, se harán las sumas:

    - $10 + 1 + 5 = 16$.
    - $1 + 5 + 8 = 14$.
    - $5 + 8 + 9 = 22$.
    - $8 + 9 + 2 = 19$.

    Por lo tanto, la función devolverá la lista `[16, 14, 22, 19]`.

106. Escribir un programa que solicite los elementos de una matriz de tamaño 4×4. La aplicación debe decidir si la matriz introducida corresponde a un *cuadrado mágico*, que es aquel en el que la suma de los elementos de cualquier fila, columna, diagonal principal y diagonal secundaria valen lo mismo.

107. Hacer el mismo programa del ejercicio anterior pero considerando una matriz cuadrada de cualquier tamaño ($N \times N$).

108. Introducir por teclado dos frases e indicar cuál de ellas es la más corta, es decir, la que tiene menos caracteres.

109. Diseñar el juego «acierta la contraseña». La mecánica del juego es la siguiente: el primer jugador introduce la contraseña sin que la vea el segundo jugador; a continuación, el segundo jugador debe teclear palabras hasta que la acierte. El programa deberá indicar en cada intento si la palabra introducida es mayor o menor (alfabéticamente) que la contraseña.

110. Hacer el mismo programa del ejercicio anterior pero con la siguiente variante: en lugar de indicar si la palabra es mayor o menor que la contraseña, deberá mostrar la longitud de la contraseña y una cadena con los caracteres acertados en sus lugares respectivos y asteriscos en los no acertados.

111. Diseñar un programa que pida al usuario que introduzca una frase por teclado e indique cuántos espacios en blanco tiene. Hacer dos versiones: una recorriendo la cadena y otra sin recorrido.

112. Diseñar una función a la que se le pase una cadena de caracteres y la devuelva invertida. Por ejemplo, la cadena «Hola mundo» quedaría «odnum aloH».

113. Escribir un programa que pida el nombre completo al usuario y lo muestre sin vocales (mayúsculas, minúsculas y acentuadas). Por ejemplo, «Álvaro Pérez» se mostraría «lvr Prz».

114. Los habitantes de Pythonlandia tienen un idioma algo extraño: cuando hablan, siempre comienzan sus frases con «Pythonín, pythonón», para después hacer una pausa más o menos larga (la pausa se representa mediante espacios en blanco o tabuladores) y a continuación expresan el mensaje. Existe un dialecto que no comienza sus frases con la muletilla anterior, pero siempre las terminan con un silencio, más o menos prolongado, y la coletilla «pythonén, nen, nen». Se pide diseñar un traductor que, en primer lugar, nos diga si la frase introducida está escrita en el idioma de Pythonlandia (en cualquiera de sus dialectos) y, en caso afirmativo, nos muestre sólo el mensaje sin muletillas.

115. Introducir por teclado una frase palabra a palabra, y mostrar la frase completa separando las palabras introducidas con espacios en blanco. Terminar de leer la frase cuando alguna de las palabras introducidas sea la cadena «fin» esritsa con cualquier combinación de mayúsculas y minúsculas. La cadena «fin» no aparecerá en la frase final.

116. Escribir un programa que lea una frase del teclado y nos indique si es palíndroma, es decir, que la frase sea igual leyendo de izquierda a derecha, que de derecha a izquierda, sin tener en cuenta los espacios ni las tildes. Por ejemplo: «Dábale arroz a la zorra el abad».

117. Se dispone de las siguientes secuencias de caracteres:

    - Secuencia 1: `e i k m p q r s t u v`
    - Secuencia 2: `p v i u m t e r k q s`

    Con ellas, es posible codificar un texto convirtiendo cada letra de la secuencia 1 en su correspondiente de la secuencia 2. El resto de los caracteres no se modifican. Las secuencias se utilizan tanto para codificar mayúsculas como minúsculas, mostrando siempre la codificación en minúsculas.

    Por ejemplo, la palabra «PaquiTo» se codifica como «matqvko».

    Escribir un programa que codifique un texto. Para ello, se debe implementar la siguiente función:

    $$\begin{cases} \textbf{Pre}: & \texttt{len}(c) = 1 \\ & \texttt{codifica}(sec1\texttt{ : str, }sec2\texttt{ : str, }c\texttt{ : str) -> str} \\ \textbf{Post}: & \texttt{len(codifica}(sec1\texttt{, }sec2\texttt{, }c\texttt{))} = 1 \end{cases}$$

    que devuelve el carácter c codificado según las secuencias 1 y 2 que se le pasan.

118. Un *anagrama* es una palabra que resulta del cambio del orden de los caracteres de otra. Ejemplos de anagramas para la palabra *roma* son: *amor*, *ramo* o *mora*. Escribir un programa que solicite al usuario dos palabras e indique si son anagramas una de otra.

119. Implementar el juego del anagrama, que consiste en que un jugador escribe una palabra y el programa muestra un anagrama suyo generado al azar. A continuación, otro jugador tiene que acertar cuál es el texto original. El programa no debe permitir que el texto introducido por el primer jugador sea la cadena vacía. Por ejemplo, si el primer jugador escribe «teclado», el programa muestra como pista un anagrama al azar, digamos, «etcloda».

120. Modificar el programa anterior para que indique al segundo jugador cuántas letras coinciden (son iguales y están en la misma posición) entre el texto introducido por él y el original.

121. Escribir un programa que lea del teclado una frase e indique, para cada carácter que aparece en la frase, cuántas veces aparece. Se consideran iguales las letras mayúsculas y minúsculas para realizar la cuenta. Por ejemplo:

```
Frase: En un lugar de La Mancha.
Resultado:
a: 4 veces
c: 1 vez
d: 1 vez
e: 2 veces
...
```

122. Realizar el juego del ahorcado. Las reglas del juego son:

    a. El jugador A teclea una palabra, sin que el jugador B la vea.
    b. Ahora se le muestran tantos guiones como letras tenga la palabra secreta. Por ejemplo, para «hola» se mostraría «`----`».
    c. El jugador B intentará acertar, letra a letra, la palabra secreta.
    d. Cada acierto muestra la letra en su lugar, y las letras no acertadas seguirán ocultas como guiones. Siguiendo con el ejemplo anterior, y suponiendo que se ha introducido la «o», la «j» y la «a», se mostrará «`-o-a`».
    e. El jugador B sólo tiene 7 intentos.
    f. La partida terminará al acertar todas las letras que forman la palabra secreta (gana el jugador B) o cuando se agoten todos los intentos (gana el jugador A).

123. Escribir un programa que pida un entero por la consola, leyéndolo del teclado, y lo imprime. Si la cadena introducida por consola no tiene el formato correcto, muestra un mensaje de error y vuelve a pedirlo.

124. Escribir un programa que lea un archivo de texto (cuyo nombre lo solicitará por teclado al usuario) y muestre su contenido por pantalla.

125. Crear el archivo de texto «`numeros_reales.txt`» en el directorio de trabajo actual que contenga una sola línea de texto con números reales separados por espacios. A continuación, escribir un programa que abre ese archivo, lea los números que contiene y calcule la suma y la media aritmética, mostrando los resultados por pantalla.

126. Pedir por teclado el nombre, la edad (un entero) y la estatura en metros (un real) de un deportista. Si algún dato tiene un formato incorrecto, deberá indicarse. En caso contrario, se deberá mostrar todos los datos en pantalla.

127. Crear un archivo de texto con una colección de números reales, uno por línea. A continuación, escribir un programa que:

    a. Abra el archivo para lectura.
    b. Lo lea línea a línea.
    c. Muestre finalmente la suma de todos ellos.

128. Crear un programa que escriba en un archivo de texto, línea a línea, frases introducidas por el teclado hasta que se introduzca la cadena «fin».

129. Escribir un programa que duplique el contenido de un archivo cuyo nombre se pide al usuario. El archivo copia tendrá el mismo nombre con el prefijo «`copia_de_`».

130. Escribir un programa que solicite al usuario el nombre de un archivo de texto y muestre su contenido en pantalla. Si no se proporciona ningún nombre de archivo, el programa usará por defecto `prueba.txt`.

131. Hacer el mismo ejercicio anterior, pero recogiendo el nombre del archivo desde la línea de órdenes del sistema operativo. (*Indicación*: usar `sys.argv`).

132. Escribir un programa que pida al usuario su nombre y su edad. Esos datos deben guardarse en el archivo `datos.txt`. Si ese archivo existe, debe añadirse al final en una nueva línea, y en caso de no existir, debe crearse.

133. Escribir un programa que lea dos listas de números enteros no ordenados de sendos archivos con un número por línea, los reúna en una lista única y los guarde en orden creciente en un tercer archivo, de nuevo uno por línea.

134. Escribir un programa que lea un archivo de texto llamado `carta.txt`. Tenemos que contar los caracteres, las líneas y las palabras. Para simplificar, supondremos que cada palabra está separada de otra por un único espacio en blanco o por un salto de línea.

135. En el archivo `numeros.txt` disponemos de una serie de números (uno por línea). Diseñar un programa que procese el archivo y nos muestre el menor y el mayor.

136. Un libro de firmas es útil para recoger los nombres de todas las personas que han pasado por un determinado lugar. Escribir un programa que permita mostrar el libro de firmas e insertar un nuevo nombre (comprobando que no se encuentre repetido). El archivo se deberá llamar `firmas.txt`.

137. En los sistemas Unix (como GNU/Linux) disponemos del comando `more`, al que se le pasa un archivo y nos lo muestra poco a poco, cada 24 líneas. Implementar un programa que funcione de forma similar.

138. Escribir la función:

$$\begin{cases} \textbf{Pre}: & \texttt{True} \\ & \texttt{lee\_enteros(}\textit{texto}\texttt{ : str) -> List[int]} \end{cases}$$

    a la que se le pasa una cadena y devuelve una lista con todos los enteros que aparecen en ella.

139. Dado un documento XML similar a éste (es decir, con la misma estructura pero no necesariamente con el mismo contenido) y almacenado en el archivo `club.xml`:

```xml
<?xml version="1.0"?>
<club>
    <nombre>Diógenes</nombre>
    <socios>
        <socio id="1">
            <nombre>Sherlock Holmes</nombre>
            <direccion>221B Baker St</direccion>
            <alta>1890-12-14</alta>
        </socio>
        <socio id="51">
            <nombre>Winston Churchill</nombre>
            <direccion>10 Downing St</direccion>
            <alta>1942-02-13</alta>
        </socio>
    </socios>
</club>
```

escribir un programa que muestre los socios del club de forma similar a la siguiente:

```
[1] Sherlock Holmes
[51] Winston Churchill
```

140. Dado el documento XML del ejercicio anterior, escribir un programa que cuente cuántos socios tiene el club y lo muestre por pantalla.

141. Dado el documento XML del ejercicio anterior, escribir un programa que cambie la dirección de todos los socios por «`Avda. de Huelva, s/n`» y guarde los cambios en el mismo archivo.

142. Dado el documento XML del ejercicio anterior, escribir un programa que cambie la dirección del socio cuyo `id` sea `1` por «`Calle Ancha, 35`» y guarde los cambios en el mismo archivo.

143. Dado el documento XML del ejercicio anterior, escribir un programa que elimine al socio cuyo `id` sea `51` y guarde los cambios en el mismo archivo.

144. Dado el documento XML del ejercicio anterior, escribir un programa que añada el teléfono del socio cuyo `id` sea `1` creándole al socio un nodo hijo que sea `<telefono>666555444</telefono>` y guarde los cambios en el mismo archivo.

145. Dado el documento XML del ejercicio anterior, escribir un programa que muestre el nombre de los socios por orden cronológico según su fecha de alta, de más antiguo a más reciente.

2. Tests

(Selecciona y arrastra con el ratón en la línea donde pone *Solución* para desvelar la solución.)

1. Al evaluar la expresión `2 < 1 or 2 != 1`, el resultado es:

    a. `1`.
    b. `2`.
    c. `True`.
    d. `False`.

    Solución: .

2. ¿Qué instrucción es equivalente a `i += 1`?

    a. `i = i + 1`.
    b. `i + 1`.
    c. `1 += i`.
    d. `i = i + i`.

    Solución: .

3. ¿Qué valor devuelve la siguiente expresión: `3 if 1 < 2 else 4`?

    a. `1`.
    b. `2`.
    c. `3`.
    d. `4`.

    Solución: .

4. Selecciona la expresión cuya evaluación resulta `3`:

    a. `3 + 2 * 6 / 5`.
    b. `(3 + 2) * 6 / 5`.
    c. `(3 + 2 * 6) / 5`.
    d. `3 + 2 * (6 / 5)`.

    Solución: .

5. Los operadores lógicos operan con valores booleanos, resultando:

    a. Valores enteros.
    b. Valores enteros y booleanos.
    c. Otros tipos de valores.
    d. Sólo valores booleanos.

    Solución: .

6. La evaluación de una operación relacional puede generar un valor de tipo:

    a. Entero.
    b. Real.
    c. Booleano.
    d. Todos los anteriores.

    Solución: .

7. La expresión `3 == 3 and 2 < 3 and 1 != 2` devuelve:

    a. `True`.
    b. `False`.
    c. No se puede evaluar.
    d. No genera un booleano, ya que la expresión es aritmética.

    Solución: .

8. La siguiente expresión, donde interviene la variable booleana `a`:

    ```
    3 != 3 or a or 1 < 2
    ```

    resulta:

    a. Dependerá del valor de `a`.
    b. `True`.
    c. `False`.
    d. No se puede evaluar.

    Solución: .

9. Elige los valores de las variables enteras (`a`, `b` y `c`) que permiten que la evaluación de la siguiente expresión sea cierta: `a < b and b != c and b <= c`:

    a. `a` = 1, `b` = 1, `c` = 2.
    b. `a` = 2, `b` = 1, `c` = 2.
    c. `a` = 1, `b` = 2, `c` = 2.
    d. `a` = 1, `b` = 2, `c` = 3.

    Solución: .

10. El bloque de instrucciones de una sentencia `if` se ejecutará:

    a. Siempre.
    b. Nunca.
    c. Dependerá de la evaluación de la expresión utilizada.
    d. Todas las respuestas anteriores son correctas.

    Solución: .

11. En una sentencia `if-else`, los bloques de instrucciones (bloque `True` y bloque `False`) pueden ejecutarse:

    a. Simultáneamente.
    b. Es posible, dependiendo de la condición utilizada, que no se ejecute ninguno.
    c. Siempre se ejecutará al menos uno y son excluyentes.
    d. Todas las anteriores son incorrectas.

    Solución: .

12. La instrucción que permite detener completamente las iteraciones de un bucle, es:

    a. `stop`.
    b. `break`.
    c. `continue`.
    d. `finish`.

    Solución: .

13. La instrucción que permite detener la iteración actual de un bucle, continuando con la siguiente (si procede) es:

    a. `stop`.
    b. `break`.
    c. `continue`.
    d. `finish`.

    Solución: .

14. ¿Cuántas veces se ejecutará la instrucción del bucle más interno en el siguiente fragmento de código?

```
for i in range(1, 11):
    for i in range(1, 6):
        print("Hola")
```

    a. 10 veces.
    b. 5 veces.
    c. 50 veces.
    d. Infinitas veces.

    Solución: .

15. Analiza el siguiente código y busca qué valores de `a` y `b` implican un menor número de iteraciones:

```
for i in range(a, a + b + 1):
    for j in range(a + b, -1, -1):
        pass
```

a. `a` = 1 y `b` = 3.
b. `a` = 3 y `b` = 1.
c. `a` = 1 y `b` = 1.
d. `a` = 3 y `b` = 3.

Solución: .

16. Una variable que se crea dentro de un ámbito sólo se podrá utilizar:

a. En cualquier parte del programa.
b. En todos los bucles.
c. Dentro del ámbito donde se ha creado.
d. Todas las opciones anteriores son correctas.

Solución: .

17. Una variable que se crea dentro de una función sólo se podrá utilizar:

a. En cualquier lugar del código.
b. Fuera de cualquier función.
c. Sólo en la función donde se ha creado.
d. Ninguna de las opciones anteriores es correcta.

Solución: .

18. ¿Qué instrucción permite a una función imperativa devolver un valor?

a. `value`.
b. `return`.
c. `def`.
d. `=`.

Solución: .

19. El tipo devuelto por todas las funciones definidas en nuestro programa tiene que ser siempre:

a. `int`.
b. `float`.
c. `bool`.
d. Ninguna de las opciones anteriores es correcta.

Solución: .

20. El paso de argumentos a una función en Python siempre es:

a. Por valor.
b. Por copia.
c. Por asignación.

d. Por referencia.

Solución: .

21. ¿Cuáles de las siguientes operaciones se pueden implementar fácilmente mediante funciones recursivas?

    a. $a^n = a \cdot a^{n-1}$.
    b. $es\_par(n) = es\_impar(n-1)$ y $es\_impar(n) = es\_par(n-1)$.
    c. $suma(a, b) = suma(a+1, b-1)$.
    d. Todas las anteriores respuestas son correctas.

    Solución: .

22. En los identificadores de las funciones en Python, al igual que en los de las variables, el convenio a usar es:

    a. *Snake case*: `suma_notas_alumnos`.
    b. Todo junto en minúsculas: `sumanotasalumnos`.
    c. *Pascal case*: `SumaNotasAlumnos`.
    d. *Camel case*: `sumaNotaAlumnos`.

    Solución: .

23. En Python, una lista puede almacenar datos de distintos tipos, como por ejemplo enteros, booleanos, reales, etc.

    a. Cierto, las listas siempre pueden almacenar datos de distintos tipos.
    b. Falso, las listas sólo pueden almacenar datos de un único tipo.
    c. Pueden almacenar datos de distintos tipos siempre que sean numéricos.
    d. Pueden almacenar datos de distintos tipos siempre que la longitud de los datos sea idéntica.

    Solución: .

24. En Python, la numeración de los índices que determina la identificación de cada elemento de una lista comienza en:

    a. Cero.
    b. Uno.
    c. Depende del tipo de dato de la lista.
    d. Es configurable por el usuario.

    Solución: .

25. Si en una lista de 10 elementos intentamos acceder al elemento con índice 11 (que se encuentra fuera de rango):

a. Al salirse del rango de la longitud, Python redimensiona la lista de forma automática.
b. No es posible y lanza una excepción.
c. Los índices tienen un comportamiento circular y utilizar el índice 11 es equivalente a utilizar el índice 1.
d. Ninguna de las anteriores respuestas es correcta.

Solución: .

26. La forma de conocer la longitud de una lista `l` es mediante:

a. `l.size`.
b. `len(l)`.
c. `l.len`.
d. `List.size(l)`.

Solución: .

27. En Python, al igual que en otros muchos lenguajes de programación, las secuencias de escape se escriben mediante:

a. Dos puntos (`:`).
b. El carácter u mayúscula (`U`).
c. El carácter u minúscula (`u`).
d. Una barra invertida (`\`).

Solución: .

28. Señala qué opción es cierta en Python:

a. `'a'` es una cadena de caracteres.
b. `"a"` es una cadena de caracteres.
c. `"""a"""` es una cadena de caracteres.
d. Todas las opciones anteriores son ciertas.

Solución: .

29. La forma de extraer en Python el cuarto carácter de la cadena almacenada en la variable `cad` es:

a. `cad.index(4)`.
b. `cad.get(4)`.
c. `cad[3]`.
d. `cad.char(3)`.

Solución: .

30. El método que permite eliminar los caracteres blancos del principio y el final de una cadena es:

a. `is_white_space`.
b. `rstrip`.
c. `strip_leading`.
d. `strip`.

Solución: .

31. ¿Qué método permite convertir una lista de cadenas (por ejemplo, `['ab', 'cd', 'ef']`) en una cadena formada por la concatenación de los elementos de la cadena (en este caso, `'abcdef'`)?

a. `join`.
b. `concat`.
c. `str`.
d. `split`.

Solución: .

32. ¿Cuál de los siguientes modos de apertura de archivos se debe usar para abrir un archivo de texto en sólo lectura?

a. `r+`.
b. `wb`.
c. `a`.
d. `r`.

Solución: .

33. ¿Cuál de las siguientes preguntas es correcta?

a. Escribir en un archivo supone siempre hacer crecer el tamaño del archivo.
b. Añadir a un archivo supone siempre hacer crecer el tamaño del archivo.
c. Para leer de un archivo primero tiene que estar cerrado.
d. Siempre que se escribe en un archivo, si no existe, lo crea.

Solución: .

34. ¿Cuál es la primera acción que se debe realizar sobre un archivo?

a. Guardarlo.
b. Cerrarlo.
c. Abrirlo.
d. Escribirlo.

Solución: .

35. ¿Cuál es la última acción que se debe realizar sobre un archivo?

a. Guardarlo.

b. Cerrarlo.
c. Abrirlo.
d. Escribirlo.

Solución: .

36. ¿Cuál es el tipo de los datos leídos desde un archivo de texto?

a. Booleanos.
b. Enteros.
c. Cadenas.
d. Reales.

Solución: .

37. ¿Cuáles de las siguientes afirmaciones son correctas respecto a los modos de apertura de un archivo? (Se pueden señalar varias)

a. Cuando se abre un archivo sólo para lectura, si el archivo no existe, se produce un error.
b. Cuando se abre un archivo sólo para escritura, si el archivo no existe, se produce un error.
c. Cuando se abre un archivo sólo para lectura, si el archivo no existe, el programa abrirá un archivo vacío.
d. Cuando se abre un archivo sólo para escritura, si el archivo no existe, se creará un archivo nuevo.
e. Cuando se abre un archivo sólo para escritura, si el archivo existe, se sobreescribirá con un archivo nuevo.

Solución: .

38. ¿Cuál de las siguientes instrucciones permiten obtener el contenido completo de un archivo en forma de cadena?

a. `f.read(n)`
b. `f.read()`
c. `f.readline()`
d. `f.readlines()`

Solución: .

39. Una excepción en Python:

a. Se produce cuando se rompe el monitor del ordenador.
b. Es un valor único de una variable.
c. Se lanza cuando se produce una condición anómala durante la ejecución de un programa.
d. Tiene lugar cuando un código es sintácticamente incorrecto.

Solución: .

40. La palabra reservada **`finally`**:

    a. Termina la ejecución de un programa.
    b. Termina la ejecución de una función, forzando el **`return`**.
    c. En una estructura **`try ... except`**, fuerza la ejecución de su bloque antes de que se ejecute una sentencia **`return`** e independientemente de si se produce o no una excepción.
    d. Indica el final de una función.

    Solución: .

41. Nos tenemos que asegurar de que todos los flujos abiertos deben cerrarse antes de que termine el programa…

    a. Porque se quedarían abiertos hasta que se apague el ordenador.
    b. Porque otro programa podría alterarlos.
    c. Porque se deben liberar los recursos asociados, como los archivos. Además, podrían quedar caracteres del búfer sin escribir.
    d. Porque se pueden borrar datos de una archivo.

    Solución: .

42. Los flujos se cierran:

    a. Con el método `close`.
    b. Apagando el ordenador.
    c. Abortando el programa.
    d. Con el atributo `closed`.

    Solución: .

43. Los gestores de contexto:

    a. Permiten abrir flujos asociados con varios archivos a la vez.
    b. Es abrir archivos recurrendo a una lista.
    c. Consiste en abrir flujos sin peligro de que se produzcan excepciones.
    d. Permiten gestionar automáticamente lo que ocurre al entrar o salir de un bloque de código, y se pueden usar junto con la orden **`with`** para asegurar el cierre de un archivo previamente abierto al ejecutar el **`with`**.

    Solución: .

44. ¿De qué forma podemos importar el módulo `ElementTree` para manipular documentos XML?

    a. `import xml.etree.ElementTree as ET`
    b. `import xml.etree.ElementTree`

c. Ambas respuestas son correctas.
d. Ninguna respuesta es correcta.

Solución: .

45. Dado el siguiente documento XML:

```
<?xml version="1.0"?>
<club>
    <nombre>Diógenes</nombre>
    <socios>
        <socio id="1">
            <nombre>Sherlock Holmes</nombre>
            <direccion>221B Baker St</direccion>
            <alta>1890-12-14</alta>
        </socio>
        <socio id="51">
            <nombre>Winston Churchill</nombre>
            <direccion>10 Downing St</direccion>
            <alta>1942-02-13</alta>
        </socio>
    </socios>
</club>
```

y suponiendo que la variable `raiz` (de tipo `Element`) contiene el nodo raíz del árbol correspondiente a dicho documento, ¿qué expresión nos devuelve una lista con los nodos `<nombre>` de todos los socios del club?

a. `raiz.findall('.//nombre')`.
b. `raiz.findall('socios/*/nombre')`.
c. `raiz.iter('nombre')`.
d. `raiz.find('nombre')`.

Solución: .

46. Siendo `socio` un nodo `<socio>` del documento XML del ejercicio anterior (es decir, una variable de tipo `Element`), ¿qué expresión nos devuelve el nombre del socio en forma de cadena?

a. `socio.find('nombre').text`
b. `socio.findall('nombre').text`
c. `socio.find('nombre')`
d. `socio.nombre`

Solución: .