UNIVERSITAT JAUME I

1.º Ingeniería Informática
1.º Matemática Computacional
1.º Inteligencia Robótica
EI1003/MT1003/IR2103 Programación I
Práctica 3

En los ejercicios de este boletín vas a empezar a definir tus propias funciones para utilizarlas en tus programas. Esto supone la aparición de un nuevo tipo de requisito entre los que ha de cumplir tu código:

- Hasta ahora, lo principal (y de lo que debes seguir preocupándote) era que tus programas tuvieran el comportamiento de entrada/salida solicitado: mostrar por pantalla lo que correspondiera teniendo en cuenta las entradas por teclado del usuario.
- A partir de ahora, además, *cada una de tus funciones debe comportarse exactamente como especifique el enunciado*, devolviendo los valores que correspondan según la información recibida en sus parámetros (¡no accedas directamente a variables globales!) y sin modificar ningún objeto, leer nada de teclado ni mostrar nada por pantalla salvo que el enunciado así lo indique.

Por otra parte, aunque lo normal sea reutilizar funciones no solamente en diferentes puntos de un mismo programa, sino también una misma función en diferentes programas mediante el mecanismo de la importación, *para resolver los ejercicios de este boletín, te pedimos que cada uno de ellos lo soluciones incluyendo en un único fichero todo tu código Python.* Así pues, si una misma función definida por ti has de utilizarla en varios ejercicios, simplemente escríbela en el primero de ellos y pégala en los demás.

Ejercicio 1

Escribe una función, `área_círculo`, que reciba como parámetro el radio de un círculo y devuelva como resultado el área de ese círculo (puedes copiarla del ejercicio 21 de la práctica 1).

Escribe otra función, `longitud_circunferencia`, que reciba como parámetro el radio de una circunferencia y devuelva como resultado la longitud de esa circunferencia.

Escribe un programa que lea el radio de un círculo y, utilizando las dos funciones anteriores, calcule su área y la longitud de su circunferencia. Muestra solo dos decimales en cada resultado. Un ejemplo de ejecución del programa es:

```
Introduce el radio: 23
Área: 1661.90
Longitud: 144.51
```

Ejercicio 2

Escribe una función, `es_triángulo`, que reciba como parámetros tres números y devuelva `True` cuando los números formen un triángulo y `False` en caso contrario.

Modifica una copia del programa del ejercicio 2 de la práctica 2 para que use la función definida. Recuerda que el programa que allí se te pedía debía leer tres números, a, b y c, y escribir "Es un triángulo" o "No es un triángulo". He aquí un ejemplo de ejecución del programa en el que la llamada a la función debe devolver `True` para que el programa principal escriba "Es un triángulo":

```
Introduce el número a: 7
Introduce el número b: 5
Introduce el número c: 3
Es un triángulo
```

Y he aquí otro ejemplo, en el que `es_triángulo` habrá tenido que devolver `False`:

```
Introduce el número a: 7
Introduce el número b: 4
Introduce el número c: 1
No es un triángulo
```

Ejercicio 3

Escribe una función, `tipo_triángulo`, que reciba como parámetros tres números y devuelva `None` cuando los números no formen un triángulo y una cadena con el tipo de triángulo apropiado (`'equilátero'`, `'isósceles'` o `'escaleno'`) cuando sí lo formen. Esta función debe llamar a la función `es_triángulo` definida en el ejercicio anterior.

Modifica una copia del programa del ejercicio 19 de la práctica 2 para que use la función `tipo_triángulo` en su programa principal. Recuerda que el programa que allí se te pedía debía leer repetidamente tres números, a, b y c, hasta que los valores leídos formasen un triángulo y, entonces, escribir de qué tipo era ese triángulo. Observa el ejemplo de ejecución del programa que se te ofrecía entonces:

```
Introduce el lado a: 7
Introduce el lado b: 4
Introduce el lado c: 1
No es un triángulo
Introduce el lado a: 2
Introduce el lado b: 3
Introduce el lado c: 5
No es un triángulo
Introduce el lado a: 3
Introduce el lado b: 4
Introduce el lado c: 5
Es escaleno
```

En esta ejecución, `tipo_triángulo` habrá tenido que devolver `None` en sus dos primeras llamadas y `'escaleno'` en la última.

Ejercicio 4

Escribe una función, `calcular_cadena_repetida`, que reciba como parámetros una cadena c, un entero n y una cadena s y devuelva una cadena con n repeticiones de la cadena c, utilizando la cadena s como separador. Por ejemplo, la llamada `calcular_cadena_repetida('patata', 3, '+')` debe devolver la cadena `'patata+patata+patata'`. **Pista:** Utiliza el operador de repetición `*` sobre cadenas.

Escribe un programa principal que lea una cadena, un separador y un número máximo de repeticiones, m, y llame m veces a la función `calcular_cadena_repetida` como se muestra a continuación, variando el número de repeticiones de la cadena entre 1 y m:

```
Introduce una cadena: ==o==
Introduce un separador: +++
Introduce un máximo de repeticiones: 4
==o==
==o==+++==o==
==o==+++==o==+++==o==
==o==+++==o==+++==o==+++==o==
```

Ejercicio 5

Convierte la función `calcular_cadena_repetida` del ejercicio anterior en el procedimiento `mostrar_cadena_repetida`. Recuerda que un procedimiento es una función sin sentencia `return`. El procedimiento debe recibir los mismos parámetros que la función pero en lugar de *devolver* la cadena calculada como retorno de función, debe *mostrarla* por la pantalla.

Modifica una copia del programa principal del ejercicio anterior para que, en lugar de utilizar la función, utilice el nuevo procedimiento `mostrar_cadena_repetida`. Ten en cuenta que el nuevo programa, desde el punto de vista del usuario, debe comportarse exactamente igual que el del ejercicio anterior. **Pista:** Fíjate en que mientras en el ejercicio anterior la función devolvía una cadena que tu programa principal debía mostrar usando `print`, ahora el procedimiento ya realiza esa tarea y, por tanto, este programa, en lugar de usar `print`, debe simplemente llamar al procedimiento.

© Universitat Jaume I 2024-2025

Ejercicio 6

Escribe una función, `contar_divisores`, que reciba como parámetro un número entero y devuelva como resultado la cantidad de divisores de ese número.

Modifica una copia del programa del ejercicio 24 de la práctica 2 para que use la función definida. Recuerda que el programa que allí se te pedía debía leer un número entero y escribir precisamente cuántos divisores tenía, como ilustraba el siguiente ejemplo:

```
Introduce un número entero: 100
El número 100 tiene 9 divisores
```

Ejercicio 7

Modifica una copia del programa del ejercicio 43 de la práctica 2 para que use la función `contar_divisores` definida en el ejercicio anterior. Recuerda que el programa que se te pedía en aquel ejercicio 43 debía leer un número entero, n, y escribir el mayor número comprendido entre 1 y n, ambos inclusive, con más divisores, como ilustraba el siguiente ejemplo:

```
Introduce un número entero: 100
El número con más divisores es 96 (12 divisores)
```

Y he aquí otro ejemplo:

```
Introduce un número entero: 10
El número con más divisores es 10 (4 divisores)
```

Ejercicio 8

Escribe una función, `sumar_divisores_propios`, que reciba como parámetro un número entero y devuelva como resultado la suma de los divisores propios de ese número (recuerda: aunque *el mismo número* será divisor suyo, *no será un divisor propio*).

Escribe otra función, `es_abundante`, que reciba como parámetro un número entero y, haciendo uso de la función anterior, devuelva como resultado `True` cuando el número sea abundante y `False` en caso contrario (recuerda: un número es abundante si es menor que la suma de sus divisores propios).

Modifica una copia del programa del ejercicio 41 de la práctica 2 para que use la función `es_abundante` en su programa principal. Recuerda que el programa que allí se te pedía debía leer un número entero, n, y escribir todos los números abundantes menores que n, como ilustraba el siguiente ejemplo:

```
Introduce un número entero: 25
Los números abundantes menores que 25 son: 12 18 20 24
```

Ejercicio 9

Modifica una copia del programa del ejercicio 42 de la práctica 2 para que use en su programa principal la función `es_abundante` definida en el ejercicio anterior. Recuerda que el programa que se te pedía en aquel ejercicio 42 debía leer un número entero, n, y escribir los n primeros números abundantes, como ilustraba el siguiente ejemplo:

```
Introduce un número entero: 10
Los 10 primeros números abundantes son: 12 18 20 24 30 36 40 42 48 54
```

Ejercicio 10

Escribe una función, `son_amigos`, que reciba como parámetro dos números enteros y, haciendo uso nuevamente de la función `sumar_divisores_propios`, devuelva como resultado `True` cuando los números sean amigos y `False` en caso contrario. Ten en cuenta que dos números son amigos si cada uno de ellos es igual a la suma de los divisores propios del otro.

Escribe un programa que lea repetidamente pares de enteros, a y b, y que, tras la lectura de cada par, informe por pantalla de si esos dos números son amigos o no (y luego pregunte al usuario si desea seguir) como ilustra el siguiente ejemplo:

© Universitat Jaume I 2024-2025

```
Introduce un número entero a: 1210
Introduce un número entero b: 1184
Los dos números son amigos
¿Seguimos comprobando amistades (S/N)? S
Introduce un número entero a: 24
Introduce un número entero b: 36
Los dos números no son amigos
¿Seguimos comprobando amistades (S/N)? S
Introduce un número entero a: 55
Introduce un número entero b: 36
Los dos números no son amigos
¿Seguimos comprobando amistades (S/N)? N
¡Adiós!
```

Ejercicio 11

Escribe una función, `es_primo`, que reciba como parámetro un número entero y devuelva `True` si el número es primo y `False` en caso contrario. Como un número es primo si tiene exactamente dos divisores (la unidad[1] y él mismo), utiliza la función `contar_divisores` del ejercicio 6 para definir `es_primo`.

Escribe un programa que lea dos enteros estrictamente positivos, a y b, y determine qué números comprendidos entre a y b, ambos inclusive, son primos. Observa en el siguiente ejemplo el comportamiento deseado cuando hay primos en el rango especificado por el usuario:

```
Introduce un entero estrictamente positivo: 11
Introduce un entero mayor que 11: 67
Voy a buscar primos entre 11 y 67...
Encontrados: 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67
```

Este otro ejemplo ilustra el comportamiento deseado cuando no los hay:

```
Introduce un entero estrictamente positivo: 2975
Introduce un entero mayor que 2975: 2995
Voy a buscar primos entre 2975 y 2995...
Encontrados: ninguno
```

Ejercicio 12

Escribe una función, `leer_entero_mayor_que`, que reciba un entero como parámetro y devuelva como resultado un entero leído por teclado estrictamente mayor que el valor del parámetro. Si el primer número leído no es estrictamente mayor, la función debe solicitar repetidamente otro hasta que la condición requerida se cumpla y pueda devolverlo.

Modifica una copia del programa del ejercicio anterior para que, haciendo uso de la función que acabas de definir, tu nueva versión del programa no tenga que confiar en que el usuario siempre le proporcionará dos números positivos en el orden correcto. Ten en cuenta que todas las lecturas (así como los mensajes que rechazan un número y piden otro) son responsabilidad de la función. Este ejemplo ilustra el comportamiento conjunto deseado:

```
Introduce un entero mayor que 0: 0
Esperaba entero mayor que 0 y 0 no lo es. Dame otro: -10
Esperaba entero mayor que 0 y -10 no lo es. Dame otro: 4
Introduce un entero mayor que 4: 2
Esperaba entero mayor que 4 y 2 no lo es. Dame otro: 4
Esperaba entero mayor que 4 y 4 no lo es. Dame otro: 40
Voy a buscar primos entre 4 y 40...
Encontrados: 5 7 11 13 17 19 23 29 31 37
```

[1] El número 1 no se considera ni primo ni compuesto.

© Universitat Jaume I 2024-2025

Ejercicio 13

Escribe un programa que lea repetidamente líneas de texto y escriba, tras cada lectura, la longitud de la correspondiente línea. El programa debe despedirse y finalizar su ejecución tras la primera cadena vacía leída. Un ejemplo de ejecución del programa es:

```
Introduce un texto (vacío para acabar): No por mucho madrugar
Su longitud es 21
Introduce otro texto (vacío para acabar): amanece más temprano...
Su longitud es 23
Introduce otro texto (vacío para acabar): ¿o sí?
Su longitud es 6
Introduce otro texto (vacío para acabar):
¡Adiós!
```

Ejercicio 14

Escribe un programa que lea repetidamente líneas de texto hasta que se le introduzca una cadena vacía y que acabe mostrando, de las cadenas leídas (exceptuando la vacía), la más corta y la más larga. Los empates en longitud deben resolverse siempre a favor de la cadena más reciente. Un ejemplo de ejecución del programa es:

```
Introduce un texto (vacío para acabar): Luis
Introduce otro texto (vacío para acabar): mínimos
Introduce otro texto (vacío para acabar): amargo
Introduce otro texto (vacío para acabar): paz
Introduce otro texto (vacío para acabar): trocear
Introduce otro texto (vacío para acabar): ámbar
Introduce otro texto (vacío para acabar): dos
Introduce otro texto (vacío para acabar): patoso
Introduce otro texto (vacío para acabar):
Última cadena de mínima longitud, 3: =>dos<=
Última cadena de máxima longitud, 7: =>trocear<=
```

Pero si la primera línea leída es la cadena vacía, esto es lo que se desea:

```
Introduce un texto (vacío para acabar):
No se ha introducido ningún texto
```

Ejercicio 15

Escribe un programa que lea dos cadenas, interpretables ambas como enteros sin signo, y lleve a cabo con ellas las manipulaciones y operaciones necesarias para acabar mostrando resultados análogos a los del siguiente ejemplo, que pretende ser autoexplicativo:

```
Introduce A: 108
Introduce B: 2
C, concatenación de A y B: 1082
R, repetición de C un número B de veces: 10821082
S, suma de los números representados por R y A: 10821190
```

Y he aquí otro ejemplo de ejecución del mismo programa:

```
Introduce A: 7
Introduce B: 5
C, concatenación de A y B: 75
R, repetición de C un número B de veces: 7575757575
S, suma de los números representados por R y A: 7575757582
```

© Universitat Jaume I 2024-2025

Ejercicio 16

Escribe un programa que lea una cadena de caracteres y responda indicando si todos los caracteres de la cadena son dígitos o si alguno no lo es.

Un ejemplo de ejecución del programa es:

```
Introduce una cadena de caracteres: 12481632641282565121024
Todos los caracteres son dígitos
```

Y otro:

```
Introduce una cadena de caracteres: 123456789X1112y13
No todos los caracteres son dígitos
```

Nota: Una posible solución consiste en utilizar el método `isdigit()`, pero solo te permitimos usarlo si actúa sobre un carácter, no si lo hace sobre una cadena.

Ejercicio 17

Escribe una función, `todos_dígitos`, que reciba como parámetro una cadena y devuelva como resultado `True` cuando todos los caracteres de la cadena sean dígitos y `False` en caso contrario.

Modifica una copia del programa del ejercicio anterior para que use la función definida.

Nota: Al igual que en el ejercicio 16, solo te permitimos utilizar `isdigit()` si actúa sobre un carácter.

Ejercicio 18

Modifica una copia del programa del ejercicio 16 para completar su salida con la información siguiente:

- Si todos los caracteres son dígitos, se ha de indicar cuántos son y cuál es su suma.
- En caso contrario, ha de mostrarse el primer carácter encontrado que no sea dígito e indicar en qué posición se encontró.

Un ejemplo de ejecución del programa es:

```
Introduce una cadena de caracteres: 12481632641282565121024
Todos los caracteres son dígitos
¿Cuántos dígitos? 23
¿Suma de dígitos? 76
```

Y otro:

```
Introduce una cadena de caracteres: 000123456789X1112y13
Primer "no dígito": 'X' en posición 12
```

Ejercicio 19

Escribe un programa que lea repetidamente líneas de texto y escriba, tras cada lectura, si la cadena leída está formada únicamente por letras del alfabeto castellano (es decir, las del alfabeto inglés más la eñe, las vocales acentuadas y la u con diéresis):

- De ser así, el programa debe responder "Podría ser una palabra correcta".
- En caso contrario, debe informar de cuál es el primer carácter no válido encontrado.

Además, el programa debe despedirse y finalizar su ejecución tras la primera cadena vacía leída. Un ejemplo de ejecución del programa es:

© Universitat Jaume I 2024-2025

```
Ve introduciendo palabras, o vacío para acabar...
Nueva palabra: vergüenza
Podría ser una palabra correcta
Nueva palabra: ad hoc
Contiene un carácter que no es del alfabeto castellano: ' '
Nueva palabra: ADSL
Podría ser una palabra correcta
Nueva palabra: afecta2
Contiene un carácter que no es del alfabeto castellano: '2'
Nueva palabra: LaTeX
Podría ser una palabra correcta
Nueva palabra: cómputo
Podría ser una palabra correcta
Nueva palabra: AÑADO
Podría ser una palabra correcta
Nueva palabra: HISTÒRIA
Contiene un carácter que no es del alfabeto castellano: 'Ò'
Nueva palabra: Historia
Podría ser una palabra correcta
Nueva palabra: You&I
Contiene un carácter que no es del alfabeto castellano: '&'
Nueva palabra: Áéíóú
Podría ser una palabra correcta
Nueva palabra: cte.
Contiene un carácter que no es del alfabeto castellano: '.'
Nueva palabra: // 2+2=4
Contiene un carácter que no es del alfabeto castellano: '/'
Nueva palabra:
¡Adiós!
```

Pista: Para averiguar si un carácter es o no una letra del alfabeto castellano, una posible solución es pasarlo a minúscula y comprobar si es mayor o igual que el carácter `'a'` y menor o igual que el carácter `'z'`, o bien está contenido en la cadena `'ñáéíóúü'`.

Ejercicio 20

Escribe una función, `es_letra_castellana`, que reciba como parámetro un carácter y devuelva `True` cuando ese carácter sea una letra del alfabeto castellano y `False` en caso contrario.

Escribe una función, `primer_no_castellano`, que reciba como parámetro una cadena y devuelva como resultado el primer carácter encontrado que no pertenezca al alfabeto castellano, o `None` si todos los caracteres de la cadena son letras de este alfabeto. Esta función debe llamar a `es_letra_castellana`.

Modifica una copia del programa del ejercicio anterior para que use la función `primer_no_castellano` en su programa principal.

Ejercicio 21

Escribe un programa que lea una línea de texto y, a partir de ella, construya una nueva cadena sin guiones ni espacios en blanco iniciales ni finales que sea, por lo demás, idéntica a la leída. Obsérvese el tipo de salida deseado en el ejemplo de ejecución siguiente:

```
Introduce una cadena de caracteres: -- - Blancos y guiones por todas partes... --
Cadena limpiada: =>Blancos y guiones por todas partes...<=
Cadena original: =>-- - Blancos y guiones por todas partes... --<=
```

Otro ejemplo:

```
Introduce una cadena de caracteres: -Vaya, tengo un guion al principio, así: -.
Cadena limpiada: =>Vaya, tengo un guion al principio, así: -.<=
Cadena original: =>-Vaya, tengo un guion al principio, así: -.<=
```

© Universitat Jaume I 2024-2025

Y, por supuesto, el programa debe prever que la cadena original sea vacía, que no tenga ningún carácter que haya que eliminar o que esté íntegramente formada por blancos y guiones.

Ejercicio 22

Escribe un programa que gestione la corrección automática de exámenes de tipo test como muestra el siguiente ejemplo:

```
Introduce la plantilla de respuestas correctas: cbbaabcdcc
Ve introduciendo respuestas de alumnos, o vacío para acabar...
Nuevas respuestas: cbababcd
La cadena introducida es de longitud 8 (se esperaba 10)
Nuevas respuestas: cbababcd**
Resultados: 6 BIEN; 2 MAL; 2 NS/NC
Nuevas respuestas: a*a*a*aaaa
Resultados: 1 BIEN; 6 MAL; 3 NS/NC
Nuevas respuestas: *bbbbbbbbb
Resultados: 3 BIEN; 6 MAL; 1 NS/NC
Nuevas respuestas: cbba*bcdcc
Resultados: 9 BIEN; 0 MAL; 1 NS/NC
Nuevas respuestas: exit
La cadena introducida es de longitud 4 (se esperaba 10)
Nuevas respuestas:
Alumnos corregidos: 4
```

Obsérvese que, tras leer la plantilla de respuestas correctas, el programa debe ir pidiendo respuestas de alumnos, las de cada uno codificadas en una cadena del mismo modo que en la plantilla (un carácter por respuesta). Además, un asterisco en la cadena correspondiente a un alumno debe interpretarse como una pregunta no contestada y contabilizar dentro del apartado NS/NC; las sí contestadas, por su parte, deberán clasificarse como bien o mal contestadas. Y si, por error, el usuario introduce como cadena de un alumno una de longitud incorrecta (es decir, de longitud distinta de la de la plantilla), el programa debe quejarse y no aceptarla como respuestas que corregir. Finalmente, el programa debe informar del número de alumnos a los que se les han corregido sus respuestas y finalizar su ejecución si, esperando las de un nuevo alumno, recibe la cadena vacía.

Ejercicio 23

Escribe, sin hacer uso de ningún método de cadenas, un programa que lea dos cadenas de caracteres y responda indicando si la segunda es un sufijo de la primera. Un ejemplo de ejecución del programa es:

```
Introduce una cadena de caracteres A: desoxirribonucleico
Introduce una cadena de caracteres B: ico
B es sufijo de A
```

Y otro:

```
Introduce una cadena de caracteres A: persiana
Introduce una cadena de caracteres B: pana
B no es sufijo de A
```

Por supuesto, el programa debe prever que las cadenas sean vacías (la cadena vacía es sufijo de cualquiera) o que la segunda sea de mayor longitud que la primera (en cuyo caso no podrá ser sufijo suyo).

Ejercicio 24

Escribe un programa que lea repetidamente líneas de texto y escriba, tras cada lectura, cuántas secuencias de dígitos consecutivos encuentra en la correspondiente línea. El programa debe despedirse y finalizar su ejecución tras la primera cadena vacía leída. Un ejemplo de ejecución del programa es:

© Universitat Jaume I 2024-2025

```
Ve introduciendo cadenas de caracteres, o vacío para acabar...
Nueva cadena: 2, 3, 5, 7, 11...
Secuencias de dígitos encontradas: 5
Nueva cadena: 12481632641282565121024
Secuencias de dígitos encontradas: 1
Nueva cadena: 000123456789X1112y13
Secuencias de dígitos encontradas: 3
Nueva cadena: xx12345007+23/(4-hola*4ever)
Secuencias de dígitos encontradas: 4
Nueva cadena:
¡Adiós!
```

Ejercicio 25

Escribe una función, `contar_secuencias_dígitos`, que reciba como parámetro una cadena y devuelva como resultado la cantidad de secuencias de dígitos consecutivos que hay en la cadena dada.

Modifica una copia del programa del ejercicio anterior para que use la función definida.

© Universitat Jaume I 2024-2025