Universidad de La Laguna
E. S. Ingeniería y Tecnología
Departamento de Ingeniería Informática y de Sistemas
Programación Científica

PRACTICA 3: Módulos y funciones en Python

3.1. Objetivos

- Estudio de módulos y funciones en Python

3.2. Práctica

Todos los programas que desarrolle en esta práctica han de tomar su entrada por línea de comandos.

Si el usuario del programa escribe el nombre del mismo sin pasarle parámetros, el programa escribirá en pantalla un texto explicativo del modo de uso del mismo, en el que se indiquen los parámetros que han de pasarse y el significado de cada uno.

1. Inicie una sesión en Linux y abra un terminal/consola.
2. Sitúese en el directorio que acaba de crear para la asignatura (`cd PC`).
3. Cree un nuevo directorio para la práctica (`mkdir p03_Funciones`).
4. Sitúese en el directorio de la práctica (`cd p03_Funciones`) y cree la estructura de directorios que le permita tener una subcarpeta para el código a utilizar y otra para el resto de documentos relacionados con la práctica (enunciado, soluciones, etc.). Es decir, se creará un subdirectorio `src` y un subdirectorio `docs`.
5. Guarde el fichero pdf que contiene el enunciado de esta práctica en el directorio `docs`.
6. En el directorio `docs` cree un fichero `respuestas.txt` para almacenar las respuestas a las cuestiones que se plantean en los ejercicios.
7. Escriba una función llamada `area_circle` que, a partir del radio de un círculo, devuelva el valor de su área. Utilice el valor $3{,}1416$ como aproximación de π y tenga en cuenta que el área de un círculo es πr^2.
8. Escriba una función llamada `area_circle` que, a partir del radio de un círculo, devuelva el valor de su área. En este caso, importe el valor de π que encontrará en el módulo `math`.

9. Escriba una función que reciba una cadena y devuelva cierto si empieza por minúscula y falso en caso contrario.

10. Escriba una función que indique si un número dado es o no es perfecto. Se dice que un número es perfecto si es igual a la suma de todos sus divisores excluido él mismo. Por ejemplo, 28 es un número perfecto, pues sus divisores (excepto él mismo) son 1, 2, 4, 7 y 14, que suman 28. La función se llamará `es_perfecto`, recibirá un sólo parámetro (el número sobre el que queremos saber si es perfecto o no) y devolverá un valor booleano.

11. Escriba una función llamada `letra_dni` que, dado un número de DNI, devuelva la letra que le corresponde. La última letra del DNI puede calcularse a partir del número. Para ello, hay que dividir el número por 23 y quedarse con el resto, que será un número entre 0 y 22. La letra que corresponde a cada número se representa en la siguiente tabla:

Resto	Letra
0	T
1	R
2	W
3	A
4	G
5	M
6	Y
7	F
8	P
9	D
10	X
11	B
12	N
13	J
14	Z
15	S
16	Q
17	V
18	H
19	L
20	C
21	K
22	E

12. ¿Son equivalentes las siguientes funciones?

```
def mayor_de_edad(edad):
    if edad < 18:
        resultado = False
    else:
        resultado = True
    return resultado
```

```
def mayor_de_edad(edad):
    if edad < 18:
        return False
    else:
        return True


def mayor_de_edad(edad):
    if edad < 18:
        return False
    return True


def mayor_de_edad(edad):
    return edad >= 18
```

13. Considerando el siguiente código, haga una traza para la llamada `maximo([4, 10, -3, 0, 2, 6])`. ¿Qué ocurriría si se llamara a la función con una lista vacía? ¿Cómo se podría solucionar el problema? Tenga en cuenta que en Python hay un valor predefinido `None` que se utiliza para denotar "ausencia de valor".

```
def maximo(lista):
    candidato = lista[0]
    for elemento in lista:
        if elemento > candidato:
            candidato = elemento
    return candidato
```

14. Escriba una función que reciba una lista de números y devuelva la media de dichos números. Tener en cuenta que para la lista vacía la media es cero.

15. Escriba una función que, dada una lista de cadenas, devuelva la cadena más larga. Si dos o más cadenas miden lo mismo y son las más largas, la función devolverá una cualquiera de ellas.

16. Escriba una función que calcule $\sum_{i=a}^{b} i$. Si a es mayor que b la función deberá devolver el valor 0.

17. Escriba una función que reciba un número de DNI y una letra. La función devolverá `True` si la letra corresponde a ese número de DNI, y `False` en caso contrario. La función debe llamarse `comprueba_letra_dni`. Si se considere conveniente, desde esta función se puede llamar a la función `letra_dni`, desarrollada en uno de los apartados anteriores.

18. Escriba una función que determine (mediante la devolución de `True` o `False`) si dos números dados son amigos. Una pareja de enteros positivos a y b se llaman números amigos si a es la suma de los divisores propios de b y b es la suma de los

divisores propios de a. Los divisores propios de un número incluyen la unidad pero no al propio número. A modo de ejemplo, 220 y 284, son números amigos.

19. Escriba una función llamada `menu_generico` que reciba una lista con opciones. Cada opción se asociará a un número entre 1 y la talla de la lista y la función mostrará por pantalla el menú con el número asociado a cada opción. El usuario deberá introducir por teclado una opción. Si la opción es válida, se devolverá su valor, y si no, se le advertirá del error y se solicitará nuevamente la introducción de un valor. Un ejemplo de llamada a la función sería: `menu_generico(['Saludar', 'Despedirse', 'Salir'])`. Al ejecutarla, deberíamos obtener en pantalla el siguiente texto:

```
1) Saludar
2) Despedirse
3) Salir
Por favor, escoja una opción:
```

20. Escriba una función sin argumentos que devuelva un número aleatorio mayor o igual que $-10{,}0$ y menor que $10{,}0$. Además de utilizar las funciones específicas que proporciona el módulo `random` también escriba una versión de la función que tenga en cuenta que para generar un número en el intervalo $[MIN, MAX]$ se puede utilizar la fórmula siguiente:

(random() * (MAX - MIN + 1)) + MIN

21. Para diseñar un juego de tablero sería conveniente disponer de un "dado electrónico". Escriba una función Python sin argumentos, que se llame `dado` y que devuelva un número entero aleatorio entre 1 y 6.

22. Escriba una función que muestre, en orden inverso, las cifras de un número entero positivo. Por ejemplo, si el procedimiento recibe el número 324, deberá mostrar por pantalla el 4, el 2 y el 3 (en líneas diferentes).

23. Escriba una función `es_primo` que determine si un número es primo (devolviendo `True`) o no (devolviendo `False`). Diseña a continuación un procedimiento `muestra_primos` que reciba un número y muestre por pantalla todos los números primos entre 1 y dicho número.

24. ¿Cómo se podría mejorar el siguiente código?

```
def muestra_nota_de_alumno(alumnos, notas, alumno_buscado):
    encontrado = False
    for i in range(len(alumnos)):
        if alumnos[i] == alumno_buscado:
            print alumno_buscado, notas[i]
            encontrado = True
    if not encontrado:
        print 'El alumno %s no pertenece al grupo' %alumno_buscado
```

25. ¿Qué aparecerá por pantalla al ejecutar este programa?

```
a = 1
b = 2
[a, b] = [b, a]
print a, b
```

26. ¿Qué aparecerá por pantalla al ejecutar el siguiente programa?

```
from math import sqrt

def area_triangulo(a, b, c):
    s = (a + b + c) / 2.0
    return sqrt(s * (s-a) * (s-b) * (s-c))

s = 4
print area_triangulo(s-1, s, s+1)
print s
print a
```

27. ¿Es correcto el siguiente programa?

```
def sin_repetidos(lista):
    resultado = []
    for elemento in lista:
        if elemento not in resultado:
            resultado.append(elemento)
    return resultado

sin_repetidos([9, 3, 7, 0, 9, 1, 0, 3, 0])
print resultado
```

28. ¿Qué muestra por pantalla el siguiente programa al ser ejecutado?

```
def modifica_parametros(x, y):
    x = 1
    y[0] = 1

a = 0
b = [0, 1, 2]
modifica_parametros(a, b)

print a
print b
```

29. Diseñar un módulo en Python que facilite el trabajo con conjuntos. Recuerde que un conjunto es una lista en la que no hay elementos repetidos. Las funciones que se deberían implementar son las siguientes:

    - `list_to_set(list)`: devuelve un conjunto con los mismos elementos que hay en lista, pero sin repeticiones.
    - `union(A, B)`: devuelve el conjunto resultante de unir los conjuntos A y B.

- `intersection(A, B)`: devuelve el conjunto cuyos elementos pertenecen a A y a B.
- `difference(A, B)`: devuelve el conjunto de elementos que pertenecen a A y no a B.
- `equal(A, B)`: devuelve cierto si ambos conjuntos tienen los mismos elementos, y falso en caso contrario. (Nota: es importante tener cuenta que los conjuntos representados por las listas [1, 3, 2] y [2, 1, 3] son iguales)

Una vez diseñado el módulo, escribir un programa desde el que se importe el módulo y se utilicen al menos dos de sus funciones.

30. Situado en el directorio de la asignatura, es decir, en el directorio `PC` comprima las actividades de la práctica:
    `tar cvfz p03_Funciones.tar.gz p03_Funciones`

31. Compruebe que se ha creado el fichero `p03_Funciones.tar.gz` correctamente en el directorio actual:
    `tar tvfz p03_Funciones.tar.gz *`

32. Suba el fichero `p03_Funciones.tar.gz` a la tarea habilitada en el campus virtual.

33. Cierre la sesión.