Iteración sobre secuencias mediante `for` 2

(0)

Describe lo que hace esta función. Intenta ejecutar el código en tu mente, sin tener que usar Python. El ordenador eres tú.

1. ¿Para qué sirve?
2. ¿Qué nombre le darías a la función?
3. ¿Qué nombre le darias a la variable `something` ?

```
def mistery(elements):
    something = []
    for element in elements:
        if type(element) == list:
            something.append(element)
    return something
```

(1)

Crea la función `select_even` que recibe una lista de números y devuelve una lista con *sólo los números pares*, en el mismo orden.

Es decir, si recibe la lista `[1,2,3,4, 5, 6]` , el valor correcto de retorno es `[2,4,6]` . No es correcto devolver `[4, 2, 6]` .

- ¿Qué debe de devolver si no hay ningún numero par?
- ¿Qué debe de devolver si recibe la lista vacía?

(2)

Crea la función `select_strings` que recibe una lista y devuelve otra lista sólo con las cadenas que hubiese en la lista de entrada, en el mismo orden.

(3)

Crea la función `select_bigger` que recibe *2 parámetros*:

1. Un número
2. Una lista de números

devuelve una lista con aquellos que son mayores que el parámetro numérico.

- Recuerda preservar el orden

(4)

Crea la función `select_smaller` que recibe *2 parámetros*:

1. Un número
2. Una lista de números

devuelve una lista con aquellos que son menores que el parámetro numérico.

- Recuerda preservar el orden
- ¿Qué devolverías si recibes una lista vacía?
- ¿En qué se distingue de `select_bigger` ?

(5)

Crea la función `select_mult_5` . Recibe una lista de números y devuelve una lista con aquellos que son múltiplos de 5.

(6)

Crea la función `select_long_words` . Recibe dos parámetros:

1. Un número
2. Una lista de cadenas

Devuelve la lista de las palabras cuya longitud sea igual o mayor que el número.

- ¿Qué devolverías si el número es negativo o cero?
- ¿Qué devolverías si recibes una lista vacía?

(7)

Crea la función `select_starts_letter` . Recibe dos parámetros:

1. Una cadena de longitud 1 (un caracter)
2. Una lista de cadenas

Devuelve la lista de aquellas cadenas que empiezan por el carcter que has recibido. Respeta el orden.

- La primera cadena ha de ser un sólo caracter. Si no es así, la función no debería de continuar.
- Una forma de comprobar que un parámetro cumple una cierta condición es con la palabra clave `assert`

Uso de `assert`

`assert` se aplica a una *expresión booleana* y si ésta devuelve `False` , se produce un error y se pára la evaluación del programa.

```
assert 3 < 4 OK assert 4 < 3 ERROR
```

OJO, `assert` no es una función (por eso no lleva paréntesis a la hora de aplicarse).

`assert` se utiliza a veces para comprobar que un parámetro cumple con ciertos requisitos. Más adelante veremso formas algo mejores de hacer esto.

- Usa `assert` para asegurarte que la función sólo sigue adelante cuando la cadena de entrada tiene la longitud correcta
- ¿ `select_starts_letter` es total o parcial?

(8)

Crea la función la función `select_contains` . Recibe dos parámetros:

1. Una cadena de cualquier longitud
2. Una lista de cadenas

Devuelve una lista de aquellas cadenas que contienen la recibida como parámetro, en cualquier posición (principio, em medio, final, no importa)

DIVIDE & VENCERAS

Tienes un subproblema: averiguar si una cadena contiene a otra. Resuelve eso primero.

Teniendo en cuenta que la cadena A sólo puede ser una subcadena de la cadena B, si `len(A) <= len(B)` , ¿cómo usarías tú eso para evitar que `select_contains` haga operaciones innecesarias?

(9)

Crea la función `select_starts` . Recibe dos parámetros:

1. Una cadena
2. Una lista de cadenas

Devuelve una lista con aquellas cadenas que *empiezan* por la cadena que has recibido como parámetro.

¿Qué devolverías cunado recibes una lista vacía?

¿ `select_starts` es total o parcial?

(10)

1. Compara `select_starts` con `select_starts_letter`
2. ¿Crees que una se podría implementar a partir de la otra?

Pista: DRY

(11)

1. ¿En qué se parecen todas estas funciones?
2. ¿Qué tiene cada una de particular?
3. ¿Identificas algún patrón?

Repaso y Conclusiones

NO LEAS ESTO ANTES DE HACER LOS EJERCICIOS

Describe con tus palabras aquello que has visto de común que tienen todas estas funciones.

NO SIGAS ANTES DE HACER ESO

Una pista:

En 1908 Melitta Bentz, una ama de casa alemana registró una de las primeras patentes de una mujer por un invento de gran éxito comercial: Un filtro de papel para café (https://vintagenewsdaily.com/in-1908-the-coffee-filter-and-filter-paper-were-founded-by-a-german-housewife/)

Selectores

Todas estas funciones reciben una lista y devuelven otra lista. La lista de salida es una *versión filtrada* de la de entrada.

Todas ellas *seleccionan* (mediante un predicado o comparación) aquellos elementos que pasan el test.

Todas ellas van *construyendo una nueva lista* con aquellos elementos que van pasando el test.

Para ello, se usa una variable que empieza con la lista vacía y a ella se van añadiendo aquellos elementos que pasan el test.

Cuando llegas al final de la lista, todos los elementos seleccionados estarán en la variable.

El patrón general de un **selector** es:

```
def select(elements):
    selected = []
    for element in elements:
        if <predicate>(element):
            selected.append(element)
    return selected
```

Esqueleto común y partes específicas

Más arriba hemvos sito el esqueleto común a todo selector. Lo que es específico de cada uno está entre <>:

- el predicado de selección

A veces nos pasan parte de ese predicado como un parámetro: un valor contra el que se compara de alguna manera todos los elementos de la lista.

Todas estas funciones se parecen demasiado.

> DRY: Do not repeat yourself

Estamos cometiendo un grave pecado contra el principio DRY y deberíamos buscar una solución.

¿Qué solución se te ocurre a tí?

Más adelante resolveremos esto.

¿Reversible?

1. En tu opinión, ¿los selectores suelen ser funciones *reversibles*?
2. Si tuviesemos muchos casos de input (listas) y output (valor comprimido), ¿crees que algunos selectores serían *irreversibles estimables*?

Recuerda que descubrir una (posible) función *irreversible estimable*, a partir de muchos datos es una tarea que se resuelve con IA, en concreto Deep Learning. Los que vayan para Big Data & Machine Learning lo verán mucho.

Destrucción

Ninguna de estas funciones altera o destruye sus parámetros. Por ejemplo, la lista de entrada queda tal cual después de llamar a la función.

> Las funciones que alteran sus parámetros se llaman destructivas. Son más peligrosas que una piraña en un bidé

Nunca implementes funciones destructivas a no ser que sea absolutamente indispensable. Si lo haces, documéntalo muy bien.