Bucle `for_in`

Sintaxis:

```
for <<variable>> in <<secuencia>>:  <<cuerpo del bucle>>
```

Bucles `for_in` con listas

In [1]:
```
def assessment(grade_list):
    """
    Computes the average of a list of grades

    Parameters
    ----------
    grades : [float]
        List of grades

    Returns
    -------
    float
        Average of grades

    Example
    -------
    >>> assessment([5,6,9,10])
    7.5
    """
    average = 0.0
    for grade in grade_list:
        average += grade
    return average / len(grade_list)
```

In [5]:
```
assessment([4.5, 6, 5]), assessment([4, 6, 5, 7, 5, 6, 8]), assessment([5,6,9,10])
```

Out [5]:
```
(5.166666666666667, 5.857142857142857, 7.5)
```

In [8]:
```
def short_names(name_list, n):
    """
    From a list of names, the function chooses the names with length below of equal n.
    All the names that satisfy the condition are returned in a list.

    Parameters
    ----------
    name_list : [string]
        List of names
    n : int
        Maximum length

    Returns
    -------
    [string]
        List of names in name_list with length <= n

    Example
    -------
    >>> short_names(l, ['Ana', 'Marta', 'Patricia', 'Alba', 'Silvia', 'Gloria', 'Lara'
    ['Ana']
    """
    short = []
    for name in name_list:
        if len(name) <= n:
            short.append(name)
    return short
```

In [9]:
```
l = ['Ana', 'Marta', 'Patricia', 'Alba', 'Silvia', 'Gloria', 'Lara']
short_names(l, 5), short_names(l, 3)
```

Out [9]:
```
(['Ana', 'Marta', 'Alba', 'Lara'], ['Ana'])
```

La función `range()`

La función `range()` genera listas de forma muy versatil. Una manera muy frecuente de hacer bucles `for_in` es generando la lista que hace de secuencia con la función `range()`. Veamos algunos detalles sobre esta función `range()`.

In [10]:
```
range(10)
```

Out [10]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [11]:
```
range(3, 12)
```

Out [11]:
```
[3, 4, 5, 6, 7, 8, 9, 10, 11]
```

In [12]:
```
range(5, 60, 5)
```

Out [12]:
```
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55]
```

In [13]:
```
range(10, 2)
```

Out [13]:
```
[]
```

In [14]:
```
range(10, 2, -1)
```

Out [14]:
```
[10, 9, 8, 7, 6, 5, 4, 3]
```

In [15]:
```
a=6
b=10
range(a-1, (b*2)-3)
```

Out [15]:
```
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
```

Bucles `for_in` para listas generadas con `range()`

In [19]:
```
def random_list(n):
    """
    Returns a list of n random integer between 0 and 100.

    Parameters
    ----------
    n : int
        Lenght of the resulting list of random integers

    Returns
    -------
    [int]
        List of random integers between 0 and 100 with length n

    Example
    -------
    >>> random_list(3)
    [1, 88, 31]
    """
    import random
    result = []
    for x in range(n):
        result.append(random.randint(0, 100))
    return result
```

In [20]:
```
random_list(3), random_list(5)
```

```
([3, 21, 93], [4, 25, 43, 4, 42])
```

Out [20]:

In [24]:
```
def random_list(n, minimum, maximum):
    """
    Returns a list of n random integers between minimum and maximum.

    Parameters
    ----------
    n : int
        Number of random integers
    minimum : int
        Minimum value of the generated random numbers
    maximum : int
        Maximum value of the generated random numbers

    Returns
    -------
    [int]
        List of n random numbers between minimum and maximum

    Example
    -------
    >>> random_list(3,1,5)
    [2, 4, 4]
    """
    import random
    result = []
    for x in range(n):
        result.append(random.randint(minimum, maximum))
    return result
```

In [25]:
```
random_list(3,1,5), random_list(5,0,1000)
```

Out [25]:
```
([3, 5, 2], [69, 983, 30, 26, 111])
```

In [28]:
```
def multiple_7_and_5(n):
    """
    Returns the list of positive numbers below n that are, at the same time,
    multiple of 7 and 5.

    Parameters
    ----------
    n : int
        Right limit

    Returns
    -------
    [int]
        List of numbers below n that are multiple of 7 and 5

    Example
    -------
    >>> multiple_7_and_5(100)
    [0, 35, 70]
    """
    result = []
    for x in range(n):
        if (x % 5 == 0) and (x % 7 == 0):
            result.append(x)
    return result
```

In [29]:
```
multiple_7_and_5(100)
```

Out [29]:
```
[0, 35, 70]
```

Si no nos gusta que aparezca el 0, podemos hacer que el rango comience en 1.

In [30]:
```
def multiple_7_and_5(n):
    """
    Returns the list of numbers in [1..n) that are, at the same time,
    multiple of 7 and 5.

    Parameters
    ----------
    n : int
        Right limit

    Returns
    -------
    [int]
        List of numbers in [1..n) that are multiple of 7 and 5

    Example
    -------
    >>> multiple_7_and_5(100)
    [35, 70]
    """
    result = []
    for x in range(1, n):
        if (x % 5 == 0) and (x % 7 == 0):
            result.append(x)
    return result
```

In [31]:
```
multiple_7_and_5(100)
```

Out [31]:
```
[35, 70]
```

Pero… no es una forma muy eficaz, hay muchos números de los que podemos fácilmente *'librarnos'*

In [32]:
```
def multiple_7_and_5(n):
    """
    Returns the list of numbers in [1..n) that are, at the same time,
    multiple of 7 and 5.

    Parameters
    ----------
    n : int
        Right limit

    Returns
    -------
    [int]
        List of numbers in [1..n) that are multiple of 7 and 5

    Example
    -------
    >>> multiple_7_and_5(100)
    [35, 70]
    """
    result = []
    for x in range(7, n, 7):
        if x % 5 == 0:
            result.append(x)
    return result
```

In [33]:
```
multiple_7_and_5(100)
```

Out [33]:
```
[35, 70]
```

In [36]:
```
def reverse(initial_list):
    """
    Returns a list with the elements of initial_list reversed, that is, the first elem
    initial_list would be the last element, the second element of initial_list woulb b
    second to last...

    Parameters
    ----------
```

```
    initial_list : list
        Original list

    Returns
    -------
    list
        Reversed list

    Example
    -------
    >>> reverse([1,2,3,4])
    [4, 3, 2, 1]
    """
    result = []
    start = len(initial_list)-1
    end = -1
    for i in range(start, end, -1):
        result.append(initial_list[i])
    return result
```

In [37]:
```
reverse([1,2,3,4]), reverse(["hola","buenas","tardes"])
```

Out [37]:
```
([4, 3, 2, 1], ['tardes', 'buenas', 'hola'])
```

Bucles `for_in` en strings

In [39]:
```
for c in 'hola':
    print c,
```

```
h o l a
```

In [40]:
```
for c in 'buenas tardes':
    print c.lower(), ord(c.lower()), c.upper(), ord(c.upper())
```

```
b 98 B 66
u 117 U 85
e 101 E 69
n 110 N 78
a 97 A 65
s 115 S 83
  32   32
t 116 T 84
a 97 A 65
r 114 R 82
d 100 D 68
e 101 E 69
s 115 S 83
```

In [44]:
```
def letter_count(letter, word):
    """
    Counts the occurrences of letter in word.

    Parameters
    ----------
    letter : string
        Letter to count the occurrences
    word : string
        Word

    Result
    ------
    int
        Number of ocurrences of letter in word
```

```
    Example
    -------
    >>> letter_count('o', 'pelirrojo')
    2
    """
    cont = 0
    for char in word:
        if char == letter:
            cont = cont + 1
    return cont
```

```
letter_count('o', 'pelirrojo')
```

In [45]: 2

Out [45]:

```
letter_count('j', 'pelirrojo')
```

In [46]: 1

Out [46]:

```
letter_count('a', 'pelirrojo')
```

In [47]: 0

Out [47]:

```
letter_count('J', 'pelirrojo')
```

In [48]: 0

Out [48]:

In [49]:

```
def letter_count(letter, word):
    """
    Counts the occurrences of letter in word. This function is not case sensitive, tha
    is letter_count('A', 'Ana') = 2 and letter_count('a', 'ANA') = 2.

    Parameters
    ----------
    letter : string
        Letter to count the occurrences
    word : string
        Word

    Result
    ------
    int
        Number of ocurrences of letter in word, ignoring case

    Example
    -------
    >>> letter_count('L', 'pelirrojo')
    1
    """
    cont = 0
    for char in word:
        if char.upper() == letter.upper():
            cont = cont + 1
    return cont
```

```
letter_count('j', 'Pelirrojo')
```

In [50]: 1

Out [50]:

```
letter_count('J', 'Pelirrojo')
```

In [51]: 1

Out [51]:

```
letter_count('p', 'Pelirrojo')
```

In [52]: 1

Out [52]:

In [20]: