Bucle while

Sintaxis:

`while <<condición>>:  <<instrucción>>`

Ejemplo muy sencillo

Sumar los números enteros en un determinado rango

In [26]:
```
n = 5
cont = 2
result = 0
while cont <= n:
    result = result + cont
    cont = cont + 1
result
```

Out [26]:
```
14
```

En forma de función

In [27]:
```
def sum_range(a,b):
    """
    This function calculates the sum of all integers from a to b,
    including both. If b<a this function returns 0 (there are no
    integers holding the condtions.

    Parameters
    ----------
    a : int
        Start of the range
    b : int
        Finish of the range

    Returns
    -------
    int
        Sum of the range [a..b]

    Example
    -------
    >>> sum_range(1,4)
    10
    """
    cont = a
    result = 0
    while cont <= b:
        result = result + cont
        cont = cont + 1
    return result
```

In [28]:
```
sum_range(3,8),sum_range(-3,7),sum_range(6,-3),sum_range(1,4)
```

Out [28]:
```
(33, 22, 0, 10)
```

Algo un poco más interesante

Calcular las potencias de dos de un número.

Ejemplos: `12 = 2^2 * 3, 16 = 2^4, 7 = 2^0 * 7`.Una primera aproximación.

In [29]:
```
def pow2(n):
    """
    This function extract the greatest power of two of a given number n >= 0.
    It returns a tuple, the power of two and the remaining, that is
    pow2(n) = t, k then t is a power of two and t*k = n.

    Parameters
    ----------
    n: int
        Integer to extract the greatest power of two

    Returns
    -------
    (int, int)
        Pair (t,k) where t is the power of two and t*k = n

    Example
    -------
    >>> pow2(12)
    (4,3)
    """
    pow = 1
    while (n % 2) == 0:
        pow = pow * 2
        n = n // 2
    return pow, n
```

In [30]:
```
pow2(2), pow2(8), pow2(7), pow2(12), pow2(2**100)
```

Out [30]:
```
((2, 1), (8, 1), (1, 7), (4, 3), (1267650600228229401496703205376L,
1L))
```

Ya estamos muy cerca, ahora contamos el exponente de la potencia en lugar de calcularlo.

In [31]:
```
def power2(n):
    """
    This function extract the greatest power of two of a given number n >= 0.
    It returns a tuple, the exponent of the power of two and the remaining, that is
    pow2(n) = s, k then (2**s)*k = n.

    Parameters
    ----------
    n: int
        Integer to extract the greatest power of two

    (int, int)
        Pair (s,k) where s is the exponent and (2**s)*k = n

    Example
    -------
    >>> power2(12)
    (2, 3)
    """
    cont = 0
    while (n % 2) == 0:
        cont = cont + 1
        n = n // 2
    return cont, n
```

In [32]:
```
power2(2), power2(8), power2(7), power2(12)
```

Out [32]:
```
((1, 1), (3, 1), (0, 7), (2, 3))
```

Si queremos 'ver' los resultados más bonitos...

In [33]:
```
n = 3214134134
a, b = power2(n)
print str(n) + " = 2^" + str(a) + " * " + str(b)
print n, "= 2^", a, "*", b
```

```
3214134134 = 2^1 * 1607067067
3214134134 = 2^ 1 * 1607067067
```

No es buena idea introducir estas expresiones en el `return` de la función. La función es más útil si devolvemos el par de números en lugar de un string.

Suma de las cifras de un número

Con un poco de paciencia podemos hacerlo paso a paso utilizando únicamente expresiones y asignaciones

In [34]:
```
n = 1536
```

In [35]:
```
cifra1 = n % 10
cifra1
```

Out [35]:
```
6
```

In [36]:
```
suma = cifra1
n = n // 10
suma, n
```

Out [36]:
```
(6, 153)
```

In [37]:
```
cifra2 = n % 10
suma = suma + cifra2
n = n // 10
suma, n
```

Out [37]:
```
(9, 15)
```

In [38]:
```
cifra3 = n % 10
suma = suma + cifra3
n = n / 10
suma, n
```

Out [38]:
```
(14, 1)
```

In [39]:
```
cifra4 = n % 10
suma = suma + cifra4
n = n / 10
suma, n
```

Out [39]:
```
(15, 0)
```

Con `while` la cosa es más sencilla, general y clara

In [40]:
```
def digit_sum(n):
    """
    This function computes the sum of the digits of a positive integer, n >= 0.

    Parameters
    ----------
    n : int
        Positive integer to sum the digits

    Returns
    -------
    int
        Sum of the digits of n

    Example
    -------
    >>> digit_sum(123)
    6
    """
    result = 0
    while n != 0:
        digit = n % 10
        result = result + digit
```

```
        n = n // 10
    return result
```

In [41]:
```
n = 233435342523452345234532452435245243522
print digit_sum(n)
print digit_sum(123)
```

```
133
6
```

El número `233432436598764523578` es divisible por `3` y por `9`.

¿Por qué?

In [42]:
```
def divisible_by_3(n):
    """
    This function decides if a positive integer is divisible by 3. n >= 0.

    Parameters
    ----------
    n : int
        Integer positive number

    Returns
    -------
    bool
        Whether n is divisible by 3 or not

    Example
    -------
    >>> divisible_by_3(14)
    False
    """
    copy = n
    while copy > 9:
        copy = digit_sum(copy)
    if (copy == 0) or (copy == 3) or (copy == 6) or (copy == 9):
        return True
    else:
        return False
```

In [43]:
```
print divisible_by_3(3341324134132241231)
print divisible_by_3(14)
```

```
True
False
```

In [44]:
```
def divisible_by_9(n):
    """
    This function decides if a positive integer is divisible by 9. n >= 0.

    Parameters
    ----------
    n : int
        Integer positive number

    Returns
    -------
    bool
        Whether n is divisible by 9 or not

    Example
    -------
    >>> divisible_by_9(19)
    False
    """
    copy = n
    while copy > 9:
```

```
        copy = digit_sum(copy)
    if (copy == 0) or (copy == 9):
        return True
    else:
        return False
```

In [45]:
```
divisible_by_9(18), divisible_by_9(3413413413414), divisible_by_9(19)
```

Out [45]:
```
(True, True, False)
```

¿Te atreves?

In [49]:
```
def divisible_by_11(n):
    .....
    .....
```

In []:
```
```