Bucle while

Sintaxis:

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

Ejemplo muy sencillo

Sumar los números enteros en un determinado rango

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

Out[1]:
```
14
```

En forma de función

In [3]:
```
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.

    @type a: int
    @type b: int
    @rtype: int
    """
    cont = a
    result = 0
    while cont <= b:
        result = result+cont
        cont = cont+1
    return result
```

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

Out[4]:
```
(33, 22, 0)
```

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 [5]:
```
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.

    @type n: int
    @rtype: int, int
    """
    pow = 1
    while n%2 == 0:
        pow = pow*2
        n = n // 2
    return pow, n
```

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

Out[7]:
```
((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 [8]:
```
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.

    @type n: int
    @rtype: int, int
    """
    cont = 0
    while n%2 == 0:
        cont = cont+1
        n = n//2
    return cont,n
```

In [9]:
```
power2(2), power2(8), power2(7), power2(12)
```

Out[9]:
```
((1, 1), (3, 1), (0, 7), (2, 3))
```

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

In [16]:
```
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 [17]:
```
n = 1536
```

In [18]:
```
cifra1 = n%10
cifra1
```

Out[18]: 6

In [19]:
```
suma = cifra1
n = n//10
suma, n
```

Out[19]: (6, 153)

In [20]:
```
cifra2 = n%10
suma = suma+cifra2
n = n//10
suma, n
```

Out[20]: (9, 15)

In [21]:
```
cifra3 = n%10
suma = suma+cifra3
n = n/10
suma,n
```

Out[21]: (14, 1)

In [22]:
```
cifra4 = n%10
suma = suma+cifra4
n = n/10
suma,n
```

Out[22]: (15, 0)

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

In [37]:
```
def digit_sum(n):
    """
    This function computes the sum of the digits of a positive integer, n >= 0.

    @type n: int
    @rtype: int
    """
    result = 0
    while n != 0:
        digit = n%10
        result = result + digit
        n = n//10
    return result
```

In [38]:
```
n = 233435342523452345234532452435245243522
digit_sum(n)
```

Out[38]: 133L

El número 233432436598764523578 es divisible por 3 y por 9.

¿Por qué?

In [39]:
```
def divisible_by_3(n):
    """
    This function decides if a positive integer is divisible by 3. n >= 0.

    @type n: int
    @rtype : bool
    """
    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 [44]:
```
divisible_by_3(334132413413241231)
```

Out[44]: True

In [48]:
```
def divisible_by_9(n):
    """
    This function decides if a positive integer is divisible by 9. n >= 0.

    @type n: int
    @rtype : bool
    """
    copy=n
    while copy > 9:
        copy = digit_sum(copy)
    if copy==0 or copy==9:
        return True
    else:
        return False
```

In [51]:
```
divisible_by_9(18), divisible_by_9(3413413413414), divisible_by_9(19)
```

Out[51]: (True, True, False)

¿Te atreves?

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