In [ ]:
```
# Si nos paramos en ese término y sumamos 0.1 vemos que pasa
a = a + 0.1
p(a)
```

Out[ ]:
```
'0.40000000000000002220446049250313'
```

In [ ]:
```
# Sin embargo
p(a - 0.4)
```

Out[ ]:
```
'0.00000000000000000000000000000000'
```

In [ ]:
```
# En este caso, el último incremento no hace que el resultado llegue a caer más allá de la representación del 0.4
a - (np.nextafter(0.4, 0.5))
```

Out[ ]:
```
-5.551115123125783e-17
```

In [1]:
```
# Otro caso más general:
# Sumamos n terminos 1/n
# Probar con distintos n
n = 100
suma = 0.0
for i in range(n):
    suma = suma + 1/n
suma
```

Out[1]:
```
1.0000000000000007
```

Vamos a definir algunos algoritmos para trabajar con el problema de suma de lista de números.

In [ ]:
```
# Numba permite compilar bloques de código a lenguaje máquina 'just in time'
# Las funciones con el decorator @jit se trataran de esa forma y se utilizará
# el código ya compilado cada vez que se ejecuten en lugar de ser interpretadas
# desde el bytecode.
# De momento, se puede ignorar ya lo veremos con algo más de cuidado más adelante
from numba import jit
```

In [2]:
```
from numba import jit
@jit
def suma(lista):
    suma = 0.0
    for x in lista:
        suma = suma + x
    return suma

@jit
def kahan(lista):
    # Accumulator
    suma = 0.0
```