Lezione 03: Comprehensions

Comprehensions

Modo *intensionale* di costruire una collezione (lista, set, dizionario), ovvero descrivere un insieme in base alle loro proprietà, piuttosto che in modo *estensivo* (ovvero elencando gli elementi).

```
[e for e in range(1,11) if e % 2 == 0] # list [2, 4, 6, 8, 10]
{e * 2 for e in range(1,11) if e % 2 == 0} # set {4, 8, 12, 16, 20}
{e: e**2 for e in range(1,11) if e % 2 == 0} # dict {2: 4, 4: 16, 6: 36, 8: 64, 10: 100}
```

È possibile utilizzare più variabili nella stessa comprehension:

```
[(x,y) for x in range(3) for y in 'abc']
# [(0, 'a'), (0, 'b'), (0, 'c'), (1, 'a'), (1, 'b'), (1, 'c'), (2, 'a'), (2, 'b'), (2, 'c')]
```

Oppure nestare più comprehension:

```
[[x for x in range(3)] for y in range(2)]
# [[0, 1, 2], [0, 1, 2]]
```

Early return

In una comprehension non è possibile fare un *early return* come in un ciclo `for` normale, ovvero interrompere l'esecuzione al primo elemento che soddisfa una certa condizione. Un trucco per ottenere un comportamento simile è quello di utilizzare un'**eccezione**, non lanciandola con `raise` ma **causandola**.

Ad esempio, per verificare se un numero è primo, basta fermarsi al primo divisore. Con una comprehension "normale", vengono controllati **tutti** i numeri fino alla radice quadrata in ogni caso.

```
def is_prime(x):
    return len([i for i in range(1, int(math.sqrt(x))+1) if x % i == 0]) == 0
```

Causando una divisione per zero, possiamo interrompere l'esecuzione al primo divisore trovato:

```
def is_prime(x):
    try:
        [True if x % i == 0 else 0/0 for i in range(1, int(math.sqrt(x))+1)]
        return True
    except ZeroDivisionError:
        return False
```