Funzioni come Valori

Un Esempio di Motivazione: Ordinamento in Python

In diversi problemi pratici è utile poter ordinare una collezione

A questo scopo, Python fornisce la funzione `sorted`

In [1]:
```
l = [3, 1, 4, 2, 5]
print(sorted(l))
```

```
[1, 2, 3, 4, 5]
```

- Data una collezione, `sorted` restituisce una lista ordinata

Funziona con diversi tipi di dato

...Purché i tipi considerati supportino l'operatore " **<=** "

In [2]:
```
print('Beatrice' <= 'Antonio', 'Antonio' <= 'Beatrice')
l = ['Beatrice', 'Carlo', 'Antonio']
print(sorted(l))
```

```
False True
['Antonio', 'Beatrice', 'Carlo']
```


Un Esempio di Motivazione: Ordinamento in Python

Si può invertire l'ordinamento con il parametro `reverse`

In [3]:
```
l = [3, 1, 4, 2, 5]
print(sorted(l, reverse=True))
```

```
[5, 4, 3, 2, 1]
```

Funziona con qualunque collezione o generatore

In [4]:
```
print(sorted({6, 2, 1, 9}))
print(sorted(range(1, 10), reverse=True))
print(sorted({'Walter': 30, 'Luisa': 28, 'Paolo': 27}))
```

```
[1, 2, 6, 9]
[9, 8, 7, 6, 5, 4, 3, 2, 1]
['Luisa', 'Paolo', 'Walter']
```

- Nel caso dei dizionari, si considerano solo le chiavi
- ...Che vengono ordinate secondo le normale relazione d'ordine

Un Esempio di Motivazione: Ordinamento in Python

E se volessimo ordinare un dizionario per valore?

Per esempio, potremmo volere ordinare questo dizionario per valore (voto):

```
In [5]: voti = {'Walter': 30, 'Luisa': 28, 'Paolo': 27}
```

Un possibile approccio:

- Durante l'ordinamento, al momento di confrontare due chiavi
- ...Recuperiamo il valore (voto) e lo usiamo per fare il confronto

Più in dettaglio:

- Invece di fare il confronto $k_i \leq k_j$
- ...Facciamo il confronto $d[k_i] \leq d[k_j]$

Un Esempio di Motivazione: Ordinamento in Python

Potremmo rendere le cose un po' più generali:

- Dati due elementi di una collezione x_i e x_j
- Invece di valutare $x_i \leq x_j$ valutiamo invece $f(x_i) \leq f(x_j)$

Dove la funzione f è da specificare

In questo modo

- Il metodo funziona per qualunque collezione (non solo i dizionari)
- Possiamo cambiare il criterio di ordinamento semplicemente cambindo f

La difficoltà è che $f(x)$ va valutata durante l'ordinamento

- ...Mentre quando chiamiamo una funzione (come `sort`)
- ...I parametri sono valutati prima della chiamata

Funzioni Come Dati

Per poter procedere, al momento di chiamare `sorted`

- ...Dovremmo poter passare la funzione f senza eseguirla
- I.e., dovremmo passare la funzione stessa come se fosse un dato

Ad effettuare l'esecuzione sarò poi il codice di `sorted`

La cosa interessante è che si può fare

In [6]:
```
voti = {'Walter': 30, 'Luisa': 28, 'Paolo': 27}

def get_voto(k):
    return voti[k]

print(sorted(voti, key=get_voto))
```

```
['Paolo', 'Luisa', 'Walter']
```


Funzioni Come Dati

Consideriamo di nuovo il codice appena visto

In [7]:
```
voti = {'Walter': 30, 'Luisa': 28, 'Paolo': 27}

def get_voto(k):
    return voti[k]

get_voto
```

Out[7]:
```
<function __main__.get_voto(k)>
```

Quando in Python definiamo una funzione

- Stiamo in realtà introducento una variabile
- ...Che contiene uno speciale tipo di dato (appunto, una funzione)
- Usando `get_voto`, facciamo riferimento al dato contenuto nella variabile
- ...Esattamente come con tutte le variabili

Funzioni Come Dati

Consideriamo di nuovo il codice appena visto

In [8]:
```
voti = {'Walter': 30, 'Luisa': 28, 'Paolo': 27}

def get_voto(k):
    return voti[k]

get_voto('Luisa')
```

Out[8]: 28

- Solo aggiungendo le parentesi tonde
- ...La funzione viene effettivamente eseguita

La notazione `()` si chiama anche operatore di chiamata a funzione


Funzioni Come Dati

Consideriamo di nuovo il codice appena visto

In [9]:
```
voti = {'Walter': 30, 'Luisa': 28, 'Paolo': 27}

def get_voto(k):
    return voti[k]

f=get_voto
f('Paolo')
```

Out[9]: 27

In effetti, si può anche assegnare una funzione ad una variabile

- In questo caso, nella variabile `f` inseriamo `get_voto`
- ...Così che chiamando `f` chiamiamo `get_voto`


Funzioni Come Dati

Consideriamo di nuovo il codice appena visto

In [10]:
```
voti = {'Walter': 30, 'Luisa': 28, 'Paolo': 27}

def get_voto(k):
    return voti[k]

print(sorted(voti, key=get_voto))
```

```
['Paolo', 'Luisa', 'Walter']
```

- Il parametro `key` di `sorted` è pensato per ricevere in ingresso una funzione
- ...Che viene chiamata in fase di ordinamento per effettuare i confronti

Questo significa che possiamo passare `get_voto` come valore per `key`!


Funzioni Come Dati

Vediamo come usare una funzione passata come dato

In [11]:
```
l = [2, 4, 5]

def doppio(v):
    return 2*v

def stampa(l, f):
    for v in l:
        print(f(v))

stampa(l, f=doppio)
```

```
4
8
10
```

- Durante la chiamata nella variable `f`
- Viene inserita la funzione `doppio`
- Nel codice di `stampa`, chiamando `f` chiamiamo in effetti `doppio`