Ciclo for

Mirko Cesarini - Dario Pescini
nome.cognome@unimib.it

Università di Milano Bicocca

Cicli For

- Il ciclo For è utile quando
  - il numero di ripetizioni è predeterminato
  - es. quando si deve svolgere un'operazione su tutti gli elementi di una lista
- Viene anche chiamato iteratore definito

Esempio ciclo for

- Programma

```
1 elenco = ["banane", "mele", "kiwi"]
2 for frutto in elenco:
3     print ("Voglio mangiare " + frutto + "!\n")
```

- Risultato

```
Voglio mangiare banane!
Voglio mangiare mele!
Voglio mangiare kiwi!
```

Sintassi ciclo For

- Sintassi:

```
1 for elemCor in listaElem:
2     CORPO DI ISTRUZIONI    # es.: print(elemCor)
```

- Nell'esempio precedente, la variabile *elemCor*
  - funge da "riferimento" all'elemento corrente della lista *listaElem*
  - al termine di ogni iterazione del ciclo, *elemCor* cambia e fa riferimento all'elemento successivo

Confronto For - While

- Sintassi:

```
for VARIABILE in LISTA:
    CORPO DI ISTRUZIONI    # es.: print( VARIABILE )
```

- Il blocco di istruzioni di cui sopra è equivalente a:

```
i = 0
while i < len(LISTA):
    VARIABILE = LISTA[i]
    CORPO DI ISTRUZIONI    # es.: print( VARIABILE )
    i = i + 1
```

- Il for
  - ha una sintassi più sintetica del while
  - tuttavia è meno flessibile (non è possibile modificare la condizione standard di permanenza in ciclo, né quale sarà l'elemento selezionato al ciclo successivo)
- A volte si può scegliere tra while e for, . . .
- . . . in certe circostanze invece si può usare solo il while

Ciclo for e dizionari

- Il ciclo for, applicato ad un dizionario, opera sulle chiavi di un dizionario, non sui valori
- Per accedere ai valori, occorre utilizzare la chiave

```
1 diz={"Marco":10, "Francesca":5, "Paolo":6}
2 for chiave in diz:
3     print ("chiave: %s, valore: %d" % (chiave, diz[chiave]))
```

```
chiave: Francesca, valore: 5
chiave: Paolo, valore: 6
chiave: Marco, valore: 10
```

- Questo è un modo elegante con cui si possono scorrere gli elementi (chiave e valore) di un dizionario
- Non sarebbe possibile scorere gli elementi di un dizionario con un *while*

Dizionari e ordinamento

- Nell'esempio seguente

```
1 a={1:"ciao", 2:"hallo", "pizza":"margherita"}
```

- Con quale criterio si può stabilire un ordine tra 1, 2, "pizza"? (generico e che valga per tutte le chiavi)
- Non si può
- A differenza delle liste, per i dizionari non è possibile ordinare gli elementi.
  - è errato dire che gli elementi sono "disordinati"
  - semplicemente non è possibile stabilire una relazione d'ordine tra le chiavi, pertanto gli elementi sono semplicemente non ordinati
- Al contrario, le liste utilizzano gli interi come chiavi tra interi è definita una relazione d'ordine, quindi è possibile stabilire un ordine tra gli elementi di una lista

Funzioni utili per i dizionari

- Dato un dizionario

```
1 a={"Rossi":1, "Verdi":2, "Bianchi":3}
```

- Potete utilizzare i comandi già visti in precedenza
  - len()

```
1 len(a)
```

```
3
```

  - del()

```
1 print(a) # prima
2 del(a['Rossi'])
3 print(a) # dopo
```

```
{'Rossi':1, 'Verdi':2, 'Bianchi':3}  # prima
{'Bianchi': 3, 'Verdi': 2}           # dopo
```

- Qualcosa di strano sull'output dell'ultimo punto? Vedi slide precedente

Operatori utili

- Operatore in: polimorfismo (diverso comportamento)
- Esempio 1

```
1 top3= {"Juventus":22, "Roma":19, "Udinese":16}
2 for key in top3:   #in usato nel comando for ... in ...
3     print ("Squadra: %s, punteg.: %d" % (key, top3[key]))
```

- Esempio 2

```
1 top3= ... #vedi esempio precedente
2 elSquadre=['Juventus', 'Milan', ...] #completate a
3                                      #piacere
4 for sq in elSquadre:  #in usato nel comando for
5     if sq not in top3:  #in usato per testare la presenza
6         print(sq+" non e' nelle 3 migliori squadre")
```

- Esempio 3 ...