Appunti di informatica

Lezione 8

anno accademico 2016-2017

Mario Verdicchio

Il ciclo FOR

- Molto spesso in un programma bisogna ripetere un'operazione per un numero prefissato di volte
- Si tratta di un'iterazione (ripetizione) che dipende da un conteggio
- Il conteggio è dato dalla modifica del valore di una variabile (chiamata contatore) che passa da un valore iniziale a un valore finale, ogni volta aumentando di 1
- Ogni volta che il contatore assume un nuovo valore, l'operazione viene ripetuta

Range

- In inglese, “range” vuol dire intervallo di valori
- In Python scriviamo

  range(0,10)

  per intendere l’intervallo dei valori interi che vanno da 0 (incluso) a 10 (escluso)
- Quindi range(0,10) indica i valori:

  0, 1, 2, 3, 4, 5, 6, 7, 8, e 9

Esempio di ciclo FOR

```
for x in range(0,10):
    print(“ciao”)
```

- Con questo codice diciamo al computer che deve eseguire print(“ciao”) per 10 volte, ossia per tutti i valori di x che vanno da 0 (incluso) a 10 (escluso)
- L’incremento di x viene eseguito automaticamente perché così funziona il ciclo FOR

Esercizi

Scrivere il codice Python che sia come output su schermo quanto segue:

1. sequenza orizzontale di dieci “0”
2. sequenza orizzontale di dieci elementi (“0” e “1” alternati)
3. sequenza verticale di dieci elementi (i primi cinque “ciao”, gli ultimi cinque “bye”)

Risposte

1.
```
for x in range(0,10):
    print(“0”, end=“ ”)
```

2.
```
for x in range(0,5):
    print(“0 1”, end=“ ”)
```

oppure

```
for x in range(0,10):
    if (x%2==0):
        print(“0”, end=“ ”)
    else:
        print(“1”, end=“ ”)
```

Risposte

3.
```
for x in range(0,10):
    if (x<5):
        print(“ciao”)
    else:
        print(“bye”)
```

Osservazioni

- È chiaro che, se l'operazione ripetuta è un'istruzione condizionale che dipende da una condizione basata sul valore del contatore, il ciclo FOR è più flessibile di quanto potesse sembrare all'inizio
- Il ciclo FOR non offre solo la ripetizione di una stessa operazione per un certo numero di volte, ma anche la ripetizione di un'operazione condizionale, il cui risultato può dipendere dal valore del contatore nel momento in cui tale operazione sta per essere eseguita

Scalabilità

- Attenzione, una soluzione può essere corretta ma non scalabile.
- Per stampare dieci elementi (“0” e “1” alternati), avremmo potuto anche usare un codice senza ciclo FOR:
  print(“0 1 0 1 0 1 0 1 0 1”)
- Questa soluzione è priva di scalabilità, ovvero non può essere facilmente modificata se le dimensioni dei dati trattati cambiano.
- Immaginate che, anziché stampare dieci elementi, vi chiedano di stampare diecimila elementi così.

Scalabilità

- La modifica di una soluzione non scalabile è tutt'altro che banale, dovremmo scrivere all'interno della print diecimila "0" o "1".
- Invece, il codice con il ciclo FOR è una soluzione scalabile, perché basta modificare il secondo valore del range per adattare la soluzione al nuovo problema:

```
for x in range(0,10):
    if (x%2==0):
        print("0", end=" ")
    else:
        print("1", end=" ")
```

```
for x in range(0,10000):
    if (x%2==0):
        print("0", end=" ")
    else:
        print("1", end=" ")
```

Esercizio

- Scrivere il codice Python per stampare su schermo una matrice 5x5 di “0”:

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

Iterazione

- La stampa di una matrice come questa può essere vista come la ripetizione della stampa di una riga:

1➔ 0 0 0 0 0
2➔ 0 0 0 0 0
3➔ 0 0 0 0 0
4➔ 0 0 0 0 0
5➔ 0 0 0 0 0

Iterazione

- Se, come spesso accade in informatica, iniziamo a contare da 0:

0➔ 0 0 0 0 0
1➔ 0 0 0 0 0
2➔ 0 0 0 0 0
3➔ 0 0 0 0 0
4➔ 0 0 0 0 0

Pseudocodice

- Chiamiamo “pseudocodice” un linguaggio in cui mischiamo un linguaggio di programmazione (nel nostro caso: Python) e un linguaggio naturale (nel nostro caso: l’italiano)
- La soluzione per stampare la matrice in pseudocodice:

```
for x in range(0,5):
    stampa una riga di cinque “0”
```

Da pseudocodice a codice

- L'operazione *stampa una riga di cinque "0"* può essere facilmente scritta in Python:

```
for y in range(0,5):
    print("0", end=" ")
```

- A questo punto, possiamo sostituire l'italiano nello pseudocodice con questo codice Python

Soluzione in codice Python

```
for x in range(0,5):
    for y in range(0,5):
        print(“0”, end=“ ”)
```

- In questo modo, una iterazione guidata dal contatore y fa eseguire 5 stampe di “0”, ed essendo tale iterazione a sua volta all’interno di un’iterazione, le 5 stampe vengono ripetute 5 volte, per un totale di 5 x 5 = 25 stampe di “0”

Soluzione corretta

- Attenzione: non c'è scritto da nessuna parte che bisogna andare a capo
- Eseguendo questo codice finiamo con l'avere 25 "0" tutti su una stessa riga
- Dobbiamo eseguire una print(" ") per far andare il cursore a capo
- Quando dobbiamo andare a capo?
- Ogni 5 stampe, ovvero al termine di ogni iterazione interna (quella comandata dal contatore y)

Soluzione corretta in Python

```
for x in range(0,5):
    for y in range(0,5):
        print(“0”, end=“ ”)
    print(“ ”)
```

- per ogni valore del contatore x, il contatore y completa un giro
- quando ci sono cicli innestati come questi, il contatore esterno varia lentamente, quello interno velocemente

Esercizio

- Scrivere il codice Python per stampare su schermo una matrice 5x5 fatta così:

0 1 0 1 0
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
0 1 0 1 0

Soluzione non scalabile

```
for x in range(0,5):
    if (x%2==0):
        print(“0 1 0 1 0”)
    else:
        print(“1 0 1 0 1”)
```

Per una soluzione scalabile

- Immaginiamo di disegnare la matrice con due cicli innestati riga per riga

y

x		0	1	2	3	4
		↓	↓	↓	↓	↓
	0 →	0	1	0	1	0
	1 →	1	0	1	0	1
	2 →	0	1	0	1	0
	3 →	1	0	1	0	1
	4 →	0	1	0	1	0

Per una soluzione scalabile

- Notate come gli zeri occupino posizioni in cui x e y hanno la stessa caratteristica:

		y				
		0	1	2	3	4
		↓	↓	↓	↓	↓
	0→	0	1	0	1	0
	1→	1	0	1	0	1
x	2→	0	1	0	1	0
	3→	1	0	1	0	1
	4→	0	1	0	1	0

sono entrambi pari, oppure entrambi dispari

Per una soluzione scalabile

- Se due numeri sono entrambi pari oppure entrambi dispari, la loro somma sarà comunque sempre pari
- Perciò, gli zeri vanno stampati quando (x+y)%2 == 0
- Negli altri casi (else), va stampato un “1”

Soluzione scalabile in Python

```
for x in range(0,5):
    for y in range(0,5):
        if (x+y)%2==0:
            print("0", end=" ")
        else:
            print("1", end=" ")
    print(" ")
```

non scordatevi che bisogna andare a capo ogni volta che si completa una riga, ossia ogni volta che il ciclo interno termina.

Strutture dati

- Finora abbiamo costruito strutture di dati solo sullo schermo, con sequenze di print opportunamente organizzate
- Ora vediamo come creare strutture di dati anche all'interno del computer, nella sua memoria
- In Python si chiama "list" una sequenza omogenea di dati, trattata come unitaria

```
v = [1, 3, 34, 56, 8, 78]
```

v = [1, 3, 34, 56, 8, 78]

- Con questa scrittura creiamo in memoria un cosiddetto “vettore” o “array” o “lista” (anche se il termine “lista” è usato solo in Python, mentre in altri linguaggi indica un altro tipo di struttura dati)
- Il vettore appena creato ha dimensione 6 e i valori che contiene sono quelli specificati tra parentesi quadre
- Per fare riferimento a ciascuno di tali valori scriveremo il nome del vettore, seguito dalla posizione numerica tra parentesi quadre (la prima posizione è la 0).
- Ad esempio, v[0] == 1 è vera, v[3] == 55 no.

Assegnamenti

- I riferimenti alle diverse posizioni di un vettore possono essere usati come destinazioni di assegnamenti
- Ad esempio, dopo che è stata eseguita l'operazione

  v[2] = 40

  il vettore sarà così in memoria

  [1, 3, 40, 56, 8, 78]

Porzioni di vettori

v[2:5]

- Questa scrittura fa riferimento alla porzione del vettore che va dalla posizione 2 (inclusa) alla posizione 5 (esclusa)
- I riferimenti alle porzioni possono essere usati come destinazioni di assegnamenti
- Ad esempio

  v[0:2] = [100, 101]

  vuol dire assegnare i numeri 100 e 101 alle posizioni 0 e 1 del vettore, rispettivamente

  [100, 101, 40, 56, 8, 78]

Lunghezza dei vettori

- Esiste la funzione len che dice quanti elementi un vettore contiene
- Ad esempio

  len(v) == 6
  è una condizione vera

Domanda

- Che cosa fa il seguente codice?

```
m = 0
for i in range (0,len(v)):
    if m < v[i]:
        m = v[i]
print(m)
```

Risposta

- Stampa su schermo il valore massimo presente nel vettore

Funzioni di manipolazione dei vettori

- **append**: per inserire un elemento in coda al vettore
  - si invoca scrivendo il nome del vettore, seguito da un punto (“dot notation”: notazione col punto) e dal comando “append” con un parametro che indica l’elemento da inserire (anche un altro vettore, volendo)
  - ad es. se v è un vettore [1, 2, 3], l’esecuzione di

    v.append(7)

    fa sì che v diventi [1, 2, 3, 7]
  - se il vettore è vuoto, inserire in coda vuol dire semplicemente inserire il primo elemento del vettore

Funzioni di manipolazione dei vettori

- **insert**: per inserire un elemento in un vettore in una posizione specifica
  - si invoca scrivendo il nome del vettore, seguito da un punto e dal comando “insert” con due parametri: l’indice della posizione di inserimento e il valore da inserire
  - ad es. se v è un vettore [1, 2, 3], l’esecuzione di

    v.insert(0,5)

    fa sì che v diventi [5, 1, 2, 3]
  - se un vettore è vuoto, l’unica posizione disponibile è quella con indice 0