Laboratorio di Informatica

Seconda lezione a Python

Dottore Paolo Parisen Toldin - parisent@cs.unibo.it
Dottoressa Sara Zuppiroli - sara.zuppiroli@unibo.it

L'importanza di capire

- Perché non dobbiamo dichiarare le variabili in Python?
- Su quali concetti si fondano i linguaggi procedurali?
- Quali sono le strutture dati in Python?

Obiettivi

- Problem solving
- Funzioni
- Iterazione

Problem solving

- È una attività del pensiero che si mette in atto per raggiungere un determinato risultato.
- Può essere svolta in maniera intuitiva o metodologica
- Fasi intuitive
  - Individuazione del problema
  - Suddivisione in sotto-problemi
  - Formulazione e verifica dell'ipotesi
  - Valutazione delle soluzioni
  - Implementazione della soluzione migliore
  - Verifica dei risultati

Problem solving e produzione del software

- Individuazione del problema, suddivisione in sotto-problemi
  - Fase di analisi
- Formulazione e verifica dell'ipotesi, valutazione delle soluzioni
  - Fase di progettazione
- Implementazione della soluzione migliore
  - Fase implementativa
- Verifica dei risultati
  - Fase di test

Strumenti per la suddivisione in sotto-problemi

- Analizzando il problema ci si può accorgere che questo sia suddivisibile in problemi più semplici.
- Lo strumento messo a disposizione dal linguaggio di programmazione è la *definizione di funzione*
- Cosa è una funzione?

Funzione matematica

- Funzione è una relazione che associa ad ogni $x \in \mathbf{X}$ elemento uno ed un solo elemento y tale che $f(x) = y$

- Una funzione parziale se esiste al più un $y \in \mathbf{Y}$ tale che $f(x) = y$

- Ad esempio:
  - La somma è una funzione. Prende due operandi e restituisce la somma degli operandi
    $f : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$

    $f(a, b) = a + b$

Funzione in informatica

- Una funzione permette di raggruppare una sequenza di istruzioni volte alla soluzione di un problema.

- Esempio: risolvere equazioni di secondo grado della forma $ax^2 + bx + c = 0$

  ha al suo interno la risoluzione dell'equazione di primo grado nel caso in cui $a = 0$
- Quindi è possibile invocare la funzione che risolva l'equazione di primo grado.

Funzione

- La funzione prende degli argomenti/parametri in input e restituisce un valore.
    - Nb: anche la funzione costante è una funzione!

- Possiamo schematizzare come:

mia_funzione($par_1, par_2, \ldots, par_n$) = valore_ritornato

Funzioni in Python

- Sintassi:

```
def nome_funzione({parametri}*):
    #la mia funzione calcola qualcosa
    <codice della funzione indentato>
    return risultato
```

Nota il due punti

Il codice all'interno della funzione DEVE ESSERE INDENTATO!!!

return è un nome riservato

Analogie e differenze

- Cosa c'è di simile tra la funzione matematica e la funzione informatica?

- Quali sono le differenze?

Funzioni in Python

- Proviamo a definire una funzione. Ad esempio: la somma!

```
def miasomma(a,b):
    return a+b
```

```
>> c = 3
>> d= 4
>>miasomma(c,d)
7
```

- Possiamo definire anche altre funzioni interessanti. Ad esempio: la funzione che effettua la divisione tra interi e restituisce sia il valore che il resto.

```
def miadivisione(a,b):
    resto=a%b
    valore=a/b
    return [valore,resto]
```

Provate a controllare se è corretto.
Il risultato è sempre quello voluto?

Cosa succede se passo delle stringhe?
La funzione da errore.
vedremo più avanti come gestire gli errori.

nb: il valore restituito è UN valore. È UNA lista di due valori!

Esercizio

- Risolvere l'equazione di secondo grado richiamando la funzione per la risoluzione dell'equazione di primo grado nel caso a=0.

Esercizio

- Definite miafun come una funzione che prenda in input una stringa e un numero e stampi la stringa un numero di volte pari al numero inserito.

```
>>miafun(“pippo”,2)
pippo
pippo
```

- Definire una funzione $f : (\mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}) \to \mathbf{Z}$

$$f(a,b,c) = \begin{cases} b * c & \text{se } a \bmod 2 = 1 \\ b + c & \text{altrimenti} \end{cases}$$

```
>>f(1,2,3)
>>6
>>f(2,4,6)
>>10
```

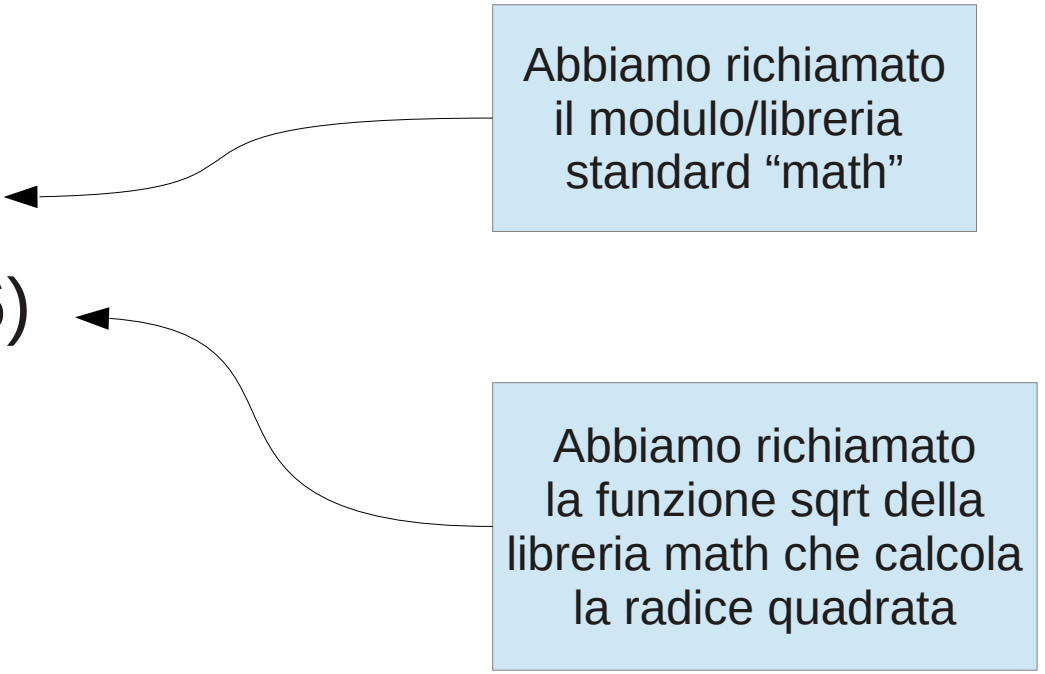
Moduli

- Definizione:
  - Si chiama modulo un file in cui sono definite delle funzioni.
  - Il nome del file è il nome del modulo e deve essere seguito da “.py”.
- Le definizioni presenti in un modulo possono essere importate in altri moduli o nel programma corrente.

Funzioni matematiche

- Ricordate l'esercizio per calcolare la radice quadrata?

```
>> import math
>> math.sqrt(16)
4.0
```



Input da tastiera

- Studieremo due modi per prendere input da tastiera.
  - Il primo se l'input desiderato è di tipo numerico

    >> mio_input = input(“Scrivi un numero: ”)
  - Il secondo se l'input è una stringa

    >> mio_input = raw_input(“Inserisci qualcosa: ”)

Testate le due istruzioni.
Che valore possiede la variabile “mio_input” dopo l'esecuzione?

NB: quello tra parentesi è quello che poi comparirà al momento della richiesta di input da tastiera

Esercizio

- Prendere in input una stringa e un numero. Scrivere una funzione che restituisca la stringa concatenata tante volte tante quante previste dal numero.

Iterazione

- Quali sono i comandi in python?
  - comando for per l'iterazione sugli elementi di una sequenza
  - comando while per i cicli a condizione iniziale

Iterazione

- Si usa il costrutto for <variabile> in <lista>:

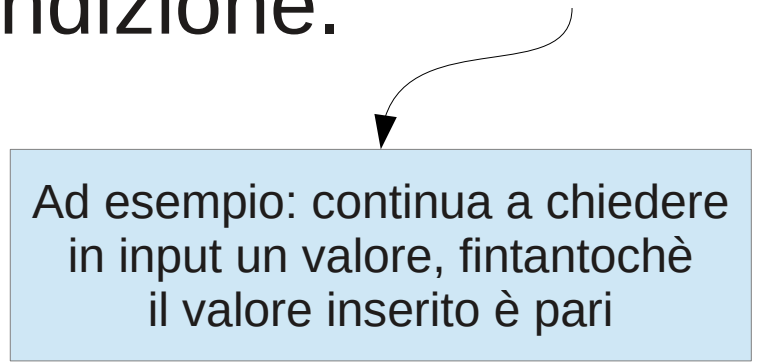
```
a=[“a”,”b”,”c”,”d”]
for mio_elemento in a:
    print mio_elemento
```

Indentazione tramite tasto TAB

  - Il programma restituirà in sequenza tutti i valori della lista.

While – iterazione condizionata

- Il costrutto while permette di definire un ciclo utilizzando una condizione.



while - sintassi

- Sintassi del comando

```
while <condizione_booleana>:
    <codice_indentato>
```

- Esempio:

```
mio_input = 0
while (mio_input%2)!=0:
    mio_input = input(“Inserisci un valore: ”)
```

Cosa fa questo Programma? Testatelo!

Esercizi

- Scrivere un programma che chiede in input un numero. Il programma continua a chiedere in input un valore fino a che il valore inserito non è un numero dispari.
- Scrivere un programma che implementi una calcolatrice.
  - Premendo 1 il programma deve chiedere due numeri ed effettuare la somma
  - Premendo 2 il programma deve chiedere due numeri ed effettuare la moltiplicazione
  - Premendo 3 il programma deve chiedere due numeri ed effettuare la sottrazione del primo con il secondo
  - Premendo 4 il programma deve chiedere un numero ed effettuare l'estrazione di radice quadrata
  - Premendo 5 il programma deve richiedere tre numeri e restituire il valore massimo
  - Premendo 6 il programma deve richiedere tre numeri e risolvere l'equazione di secondo grado associata $ax^2 + bx + c = 0$
  - Premendo 7 il programma deve terminare.

Esempio di menu:

1) somma
2) moltiplicazione
3) sottrazione
….
Scelta:

Ricorsione

- La ricorsione primitiva è una operazione definita sulle funzioni in questo modo:

$$h(x_1, \ldots, x_n, y) = \begin{cases} f(x_1, \ldots, x_n) & \text{se } y = 0 \\ g(x_1, \ldots, x_n, h(x_1, \ldots, x_n, y-1)) & \text{altrimenti} \end{cases}$$

- Quindi, “ricorsione” vuol dire che la funzione richiama sé stessa ed è definita a partire da uno o più “casi base” e poi da uno o più casi che richiamano la funzione con parametri più “semplici”.

- La definizione della funzione risulterà più leggibile e più semplice da trattare!

Ricorsione

- La somma, ad esempio, può essere definita in modo “ricorsivo”, per a$\geq$0 come

$$somma(a,b) = \begin{cases} b & \text{se } a \text{ è } 0 \\ somma(a-1, b+1) & \text{altrimenti} \end{cases}$$

- Il nostro codice python sarà dunque

```
def somma(a,b):
    if a==0 :
        return b
    else :
        return somma(a-1,b+1)
```

Provatelo, funziona? (ovviamente con numeri positivi)

Ricorsione

- L'elevamento a potenza può essere definito in modo ricorsivo come a^b

$$exp(a,b) = \begin{cases} a & \text{se } b \text{ è } 1 \\ a \cdot exp(a, b-1) & \text{altrimenti} \end{cases}$$

- Il nostro codice python sarà dunque

```
def esponenziale(a,b):
    if b==1 :
        return a
    else :
        return a*exponenziale(a,b-1)
```

Provatelo, funziona? (ovviamente con numeri positivi)

Iterazione vs Ricorsione

```
def somma(a,b):
    risultato = b
    for i in range(0,a):
        risultato = risultato + 1
    return risultato
```

```
def somma(a,b):
    if a==0 :
        return b
    else :
        return somma(a-1,b+1)
```

- A sinistra la somma iterata, a destra la somma ricorsiva
- A sinistra abbiamo una istruzione (risultato=risultato+1) che viene eseguita un numero di volte pari al valore di a.
- A destra abbiamo una funzione (somma(a,b)) che viene richiamata più volte, passando, via via, parametri diversi
- Con la ricorsione calcoliamo il risultato in modo “top-down”, cioè generiamo tutte le chiamate ricorsive di funzione che servono per calcolare il risultato
- Con l'iterazione, partiamo dal valore base e costruiamo passo dopo passo il valore finale. Visione “bottom-up”.

Esercizio

- Proviamo ad implementare la successione di fibonacci! Sappiamo che la successione di fibonacci è definita nel seguente modo:

$$fib(n) = \begin{cases} 1 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ fib(n-1) + fib(n-2) & \text{altrimenti} \end{cases}$$

- Quindi:

$fib(2) = fib(1) + fib(0) = 1 + 1 = 2$
$fib(3) = fib(2) + fib(1) = 3$
$fib(4) = 5$
$\ldots$

Esercizio

- Implementare un programma che, richiesto un numero intero positivo in input da tastiera n, stampa in output il valore fib(n).
  - Implementare il programma senza utilizzare la ricorsione ma usando il costrutto while

    suggerimento: ad ogni ciclo tenete traccia dei valori precedentemente calcolati ed aggiornateli ad ogni iterazione.

Numero aureo

- Il rapporto tra due numeri di fibonacci consecutivi tende al numero aureo $\phi = 1,61803398875$

$$\phi = \lim_{n \to \text{inf}} \frac{fib(n)}{fib(n-1)} = \lim_{n \to \text{inf}} \frac{fib(n-1) + fib(n-2)}{fib(n-1)} =$$

$$1 + \lim_{n \to \text{inf}} \frac{fib(n-2)}{fib(n-1)} \Rightarrow \phi = 1 + \frac{1}{\phi} \Rightarrow \phi^2 - \phi - 1 = 0$$

Ed otteniamo il numero aureo!

$$\phi = \frac{1 + \sqrt{5}}{2}$$

Esercizio

- Implementare la successione di fibonacci utilizzando la formula di Binet

$$fib(n) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

E' una soluzione ricorsiva, iterativa o nessuna delle due?

Esercizio

- Scrivere un programma che stampi un menù con quattro voci; ogni voce da la possibilità di calcolare la successione di fibonacci in modo diverso:
  - Ricorsione
  - Iterazione
  - Formula di Binet

(chiaramente, il menù deve anche avere la possibilità di chiudere il programma)

Esercizio

- Basandosi sull'esercizio precedente, scrivere un programma che chiede in input un valore numerico n > 0. Dopodichè, stampa tutti i valori da fib(0) a fib(n).
  - Implementare la soluzione richiamando le tre funzioni appena viste
  - Confrontare i tempi di risposta delle tre soluzioni per $n \in \{1, 10, 20, 30, 40, 50, 80, 100\}$