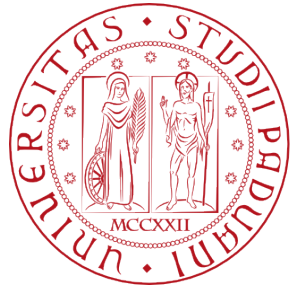
Laboratorio Programmazione
Anno 2012 - 2013
Lezione 3



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Scripting

- Fino ad ora, per fare esercizi, abbiamo utilizzato l'ambiente interattivo di python.
- È possibile creare dei file contenenti (lunghe) sequenze di istruzioni che possono essere eseguite senza doverle ridigitare ogni volta
- Per creare i file potete utilizzare
  - notepad++ se siete sotto Windows
  - gedit se siete sotto linux
- Per utilizzare gedit:
  - cercate il programma tra le applicazioni
  - oppure aprite una finestra di terminale e digitate: **gedit &**

Il mio primo script

- apriamo l'editor gedit
- ditigiamo una serie di comandi come li digiteremmo nell'ambiente interattivo
- salviamo il file con un nome significativo, ad es. primo.py
- digitare da terminale **python primo.py** corrisponde a:
  - aprire l'ambiente python (ovvero digitare **python** da terminale)
  - digitare le istruzioni nel file in sequenza
  - uscire dall'ambiente python
- più o meno. . . mentre nell'ambiente interattivo la valutazione di un'espressione è stampata a video automaticamente, in uno script è necessario utilizzare il comando **print**

Input dall'utente

- Un modo per poter modificare il valore di variabili all'interno dello script è quello di richiedere tali valori all'utente:

```
x=raw_input(''messaggio'');
```

- raw_input stampa la stringa "messaggio" e salva i caratteri digitati dall'utente nella **stringa** x

Modifica del flusso di programma: IF

- I programmi che abbiamo scritto finora consistono di un'unica sequenza di istruzioni che vengono eseguite in ordine
- È possibile eseguire una (sotto)sequenza di istruzioni solo se una condizione risulta vera:

```
if cond:
    comando1
    comando2
comando3
comando4
```

- le istruzioni **comando1** e **comando2** vengono eseguite solamente se **cond** è vera
- le istruzioni **if**, **comando3** e **comando4** vengono eseguite indipendentemente dal valore di **cond**.

Modifica del flusso di programma: IF


```
if cond:
    comando1
    comando2
else:
    comando3
comando4
```

- **comando1-4** è una qualsiasi istruzione (anche un altro **if**)
- Tutti i comandi appartenenti alla stessa sequenza di istruzioni inziano dalla stessa colonna (ma non è vero il contrario)
- il carattere : introduce una nuova sequenza di istruzioni, che termina quando un'istruzione inizia da una colonna precedente
- nell'esempio abbiamo tre distinte sequenze di istruzioni (**comando1**,**comando2**), (**comando3**) ed infine (**if**, **comando4**).

Esempio di istruzione condizionale

Assumiamo che x sia un intero definito prima delle seguenti istruzioni

```
if x<0:
   print("x e' negativa")
elif x==0:
   print("x vale 0")
else:
   print("x e' positiva")
x=x+1;
```

- Al termine di esse troviamo stampato "x e' negativa", "x vale 0", "x e' positiva" se, rispettivamente, $x < 0$, $x = 0$, $x > 0$.

Esempio di istruzione condizionale

Le linee cancellate sono quelle che non vengono eseguite

```
x=-1;
if x<0:
    print("x e' negativa")
elif x==0:
    print("x vale 0")
else:
    print("x e' positiva")
x=x+1;
```

- Output a video: "x e' negativa".
- valore di x alla fine: $x = 0$;

Esempio di istruzione condizionale

Le linee cancellate sono quelle che non vengono eseguite

```
x=0;
if x<0:
    print("x e' negativa")
elif x==0:
    print("x vale 0")
else:
    print("x e' positiva")
x=x+1;
```

- Output a video: "x vale 0".
- valore di x alla fine: $x = 1$;

Esempio di istruzione condizionale

Le linee cancellate sono quelle che non vengono eseguite

```
x=3;
if x<0:
    print("x e' negativa")
elif x==0:
    print("x vale 0")
else:
  print("x e' positiva")
x=x+1;
```

- Output a video: "x e' positiva".
- valore di x alla fine: $x = 4$;

Condizioni

- Vi ricordo che per i confronti potete utilizzare gli operatori $<$, $>$, $<=$, $>=$, $==$ (uguale), $!=$ (diverso)
- Per creare condizioni più complesse, potete collegare le condizioni sopra tramite gli operatori **and**, **or**, **not**
- Ad esempio

```
x=4; print(x>2 and x<6)
```

- restituisce True
- Ricordate che, per **x and y**, se x e' falso y non viene valutato. Allo stesso modo, per **x or y**, se x e' vero y non viene valutato

Esempio di Script

- creare uno script “guess.py” che
  - chiede all’utente di inserire un carattere tramite tastiera stampando a video il messaggio “Inserisci un carattere: ”
  - stampare un messaggio di congratulazione se il carattere e’ *a*

Esempio di Script: Soluzione

- creare uno script "guess.py" che
  - chiede all'utente di inserire un carattere tramite tastiera stampando a video il messaggio "Inserisci un carattere: "
  - stampare un messaggio di congratulazione se il carattere e' *a*

```
x=raw_input(''Inserisci un carattere: '');
if x == 'a':
  print(''Bravo, mi hai letto nel pensiero");
```

Esercizi

1. creare lo script "inputAnumber.py" che
   - inizializza una variabile *n* al valore 4
   - chiede all'utente di inserire un numero naturale maggiore o uguale a *n* tramite tastiera
   - stampa un messaggio di rimprovero nel caso che il numero sia minore di *n* oppure di felicitazioni in caso contrario
2. creare lo script "simulaAnd.py" che, senza utilizzare l'operatore and
   - inizializza due variabili, a e b, all'interno dello script ad un valore a scelta tra False,True
   - stampa "vero" se sia a che b valgono True, "falso" altrimenti
3. creare lo script "simulaOr.py" che sia l'equivalente di simulaAnd.py per l'operatore or
   - le due variabili a e b, devono essere numeri letti da standard input. Utilizzate la corrispondenza 0=False, 1=True

Esercizi

1. a partire dallo script “inputAnumber.py”, creare lo script “inputAnumber2.py” che
   - chiede all'utente di inserire un numero naturale minore di 10 tramite tastiera
   - stampa un messaggio di errore nel caso in cui il numero sia maggiore o uguale a 10
   - in ogni caso stampa se il numero è uguale o diverso da 4, e una delle seguenti possibilità: se è minore di 3, se è compreso tra 3 e 6 (inclusi) e se è maggiore di 6. Le stampe a video devono provenire ciascuna da un solo comando print (il che non significa che deve esserci un solo comando print in tutto lo script). Esempi:
     per 1 stampa “diverso da 4 e minore di 3”
     per 4 stampa “uguale a 4 e compreso tra 3 e 6”
     per 5 stampa “diverso da 4 e compreso tra 3 e 6”
     per 8 stampa “diverso da 4 e maggiore di 6”

Moduli

- È possibile utilizzare funzionalità scritte da altri nei proprio programmi tramite i moduli.
- Ad esempio, Il modulo random mette a disposizione comandi per generare numeri casuali

```
import random
n = random.randint(1,10)
```

- il comando **random.randint** permette di generare un numero intero casuale $x.1 \leq x \leq 10$

Iterazioni: While

- Se abbiamo la necessità di ripetere più volte una serie di istruzioni, possiamo utilizzare i comandi **while** e **for**

```
while cond:
  comandi
```

- ripete **comandi** fino a che cond è vera. Esempio:

```
while x>0:
  x=x-1;
  print(x);
print("fine");
```

Iterazioni: While


```
while x>0:
  x=x-1;
  print(x);
print("fine");
```

- può essere pensato mentalmente come segue (se assumete l'esistenza di un magico comando vai_all'istruzione_x)

```
1 if x>0:
2   x=x-1;
3   print(x);
4   vai_all'istruzione_1
5 print("fine");
```

Iterazioni: While


```
while cond:
  comandi
```

- ripete **comandi** fino a che cond è vera: da 0 volte se cond è inizialmente falsa a infinite se rimane sempre vera

```
while True:
  print("redrum");
```

  - Utilizzate la combinazione di tasti Ctrl C per uscire da uno script che non termina

Iterazioni: For


```
for x in ITERABILE:
  comandi
```

- ripete **comandi** per ogni elemento di ITERABILE: ad ogni iterazione x assume il valore dell'elemento corrente di ITERABILE. Esempi:

```
for x in [1,4,5]: print(x+1);
```

equivale a
x=1;print(x+1);x=4;print(x+1);x=5;print(x+1);

```
somma=0;
for x in range(0,5): somma+=x;
print(somma)
```

- il secondo esempio calcola la somma dei primi 4 numeri

Esempio

Creare uno script "max.py" che

- legge n numeri naturali dall'utente (con n una variabile inizializzata a piacere all'interno dello script)
- restituisce il maggiore dei numeri inseriti

Esempio: Soluzione

Creare uno script “max.py” che

- legge n numeri naturali dall’utente (con n una variabile inizializzata a piacere all’interno dello script)
- restituisce il maggiore dei numeri inseriti

```
massimo=?
n=3
for i in range(0,n):
    x=int(raw_input("Inserisci un numero: "));
    if x > massimo:
            massimo = x;
```

Esempio: Soluzione

Creare uno script "max.py" che

- legge n numeri naturali dall'utente (n una variabile inizializzata a piacere all'interno dello script)
- restituisce il maggiore dei numeri inseriti

```
massimo=-1
n=3
for i in range(0,n):
    x=int(raw_input("Inserisci un numero: "));
    if x > massimo:
            massimo = x;
```

Esempio

Creare uno script "ricercaCerta.py" che

- definisce la lista valori=[9,5,3,4,1,2,6,8,7];
- chiede un numero positivo minore di 10 all'utente
- restituisce l'indice del numero nella lista (notate che siamo certi di trovare il numero nella lista)
- vogliamo che il programma esamini gli elementi della lista da sinistra a destra e che non esamini più elementi del dovuto

Esempio: Soluzione

Creare uno script “ricercaCerta.py” che

- chiede un numero positivo minore di 10 all'utente
- restituisce l'indice del numero nella lista [9,5,3,4,1,2,6,8,7]

```
n=3
lista=[9,5,3,4,1,2,6,8,7];
x=int(raw_input("Inserisci x. 0<x<10: "));
trovato=False
i=0;
while trovato==False:
    if lista[i]==x:
        trovato=True;
        print("il numero e' in posizione %d" % i);
    i+=1;
```

Esercizi

1. Creare uno script "factorial.py" che
   - chiede all'utente di inserire un numero naturale
   - controlla che il numero inserito sia maggiore di 0
   - stampa il fattoriale del numero, es. 4!=4*3*2=24
2. creare uno script "tabelline.py" che calcola le tabelline dei primi 10 numeri. L'output deve essere della forma x*y=xy (con $1 \leq x \leq 10$, $1 \leq y \leq 10$): es. 4*6=24
3. Creare uno script "verifyRand.py" che
   - simula n volte il lancio di un dado e stampa le frequenze con cui ciascuna faccia del dado è uscita
   - verificate a occhio se, per n sufficientemente grande, il nostro dado virtuale è truccato
   - esempio di output: "Dopo 10000 tiri, ecco il numero di volte che ogni faccia del dado e' uscita: [1691, 1620, 1703, 1664, 1726, 1596]"

Esercizi

1. Creare uno script "orderList.py" che
   - chiede all'utente di inserire n numeri (n fissato all'interno dello script)
   - restituisce la lista, ordinata numericamente, dei numeri inseriti