Università degli Studi di Cagliari
Corsi di Laurea in Ingegneria Chimica ed Ingegneria Meccanica

FONDAMENTI DI INFORMATICA

http://people.unica.it/gianlucamarcialis

A.A. 2018/2019

Docente: **Gian Luca Marcialis**

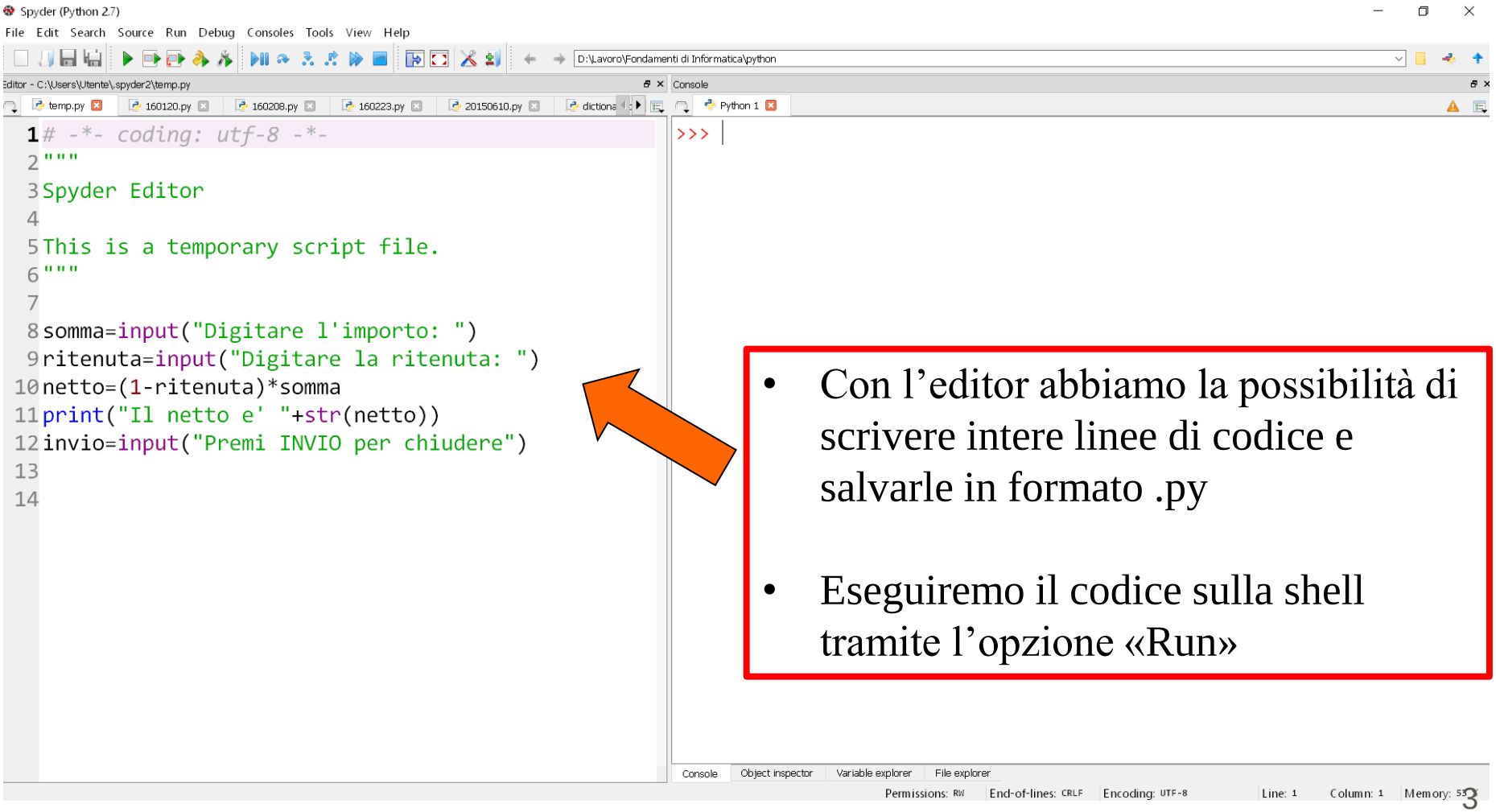
LINGUAGGIO PYTHON
Strutture di controllo

Sommario

- Introduzione
- Il costrutto **`if:`**…**`else:`**…
  - Variante **`if:`**…**`elif:`**…**`else:`**…
- Il costrutto **`while:`**
- Il costrutto **`for:`**

Avviso

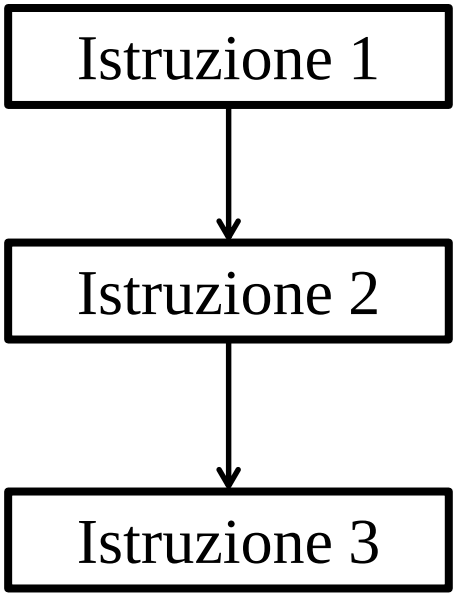
➢ In questa lezione useremo spesso l'editor:



- Con l'editor abbiamo la possibilità di scrivere intere linee di codice e salvarle in formato .py
- Eseguiremo il codice sulla shell tramite l'opzione «Run»

Introduzione

- Abbiamo visto che l'esecuzione di un programma è, in linea di principio, puramente **sequenziale**
- Nella scrittura di algoritmi c'è però la necessità, spesso, di alterare il normale flusso di esecuzione delle istruzioni, ricorrendo ad opportuni costrutti detti **strutture di controllo**
- Le strutture di controllo **alterano** il flusso di esecuzione sequenziale
- Le strutture di controllo sono di due tipi:
  - Condizionali (la maggior parte)
  - Incondizionali



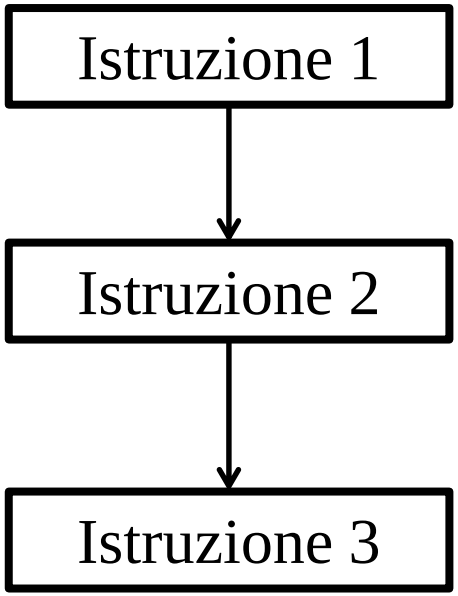
Strutture di controllo condizionali: il costrutto `if:…else:`

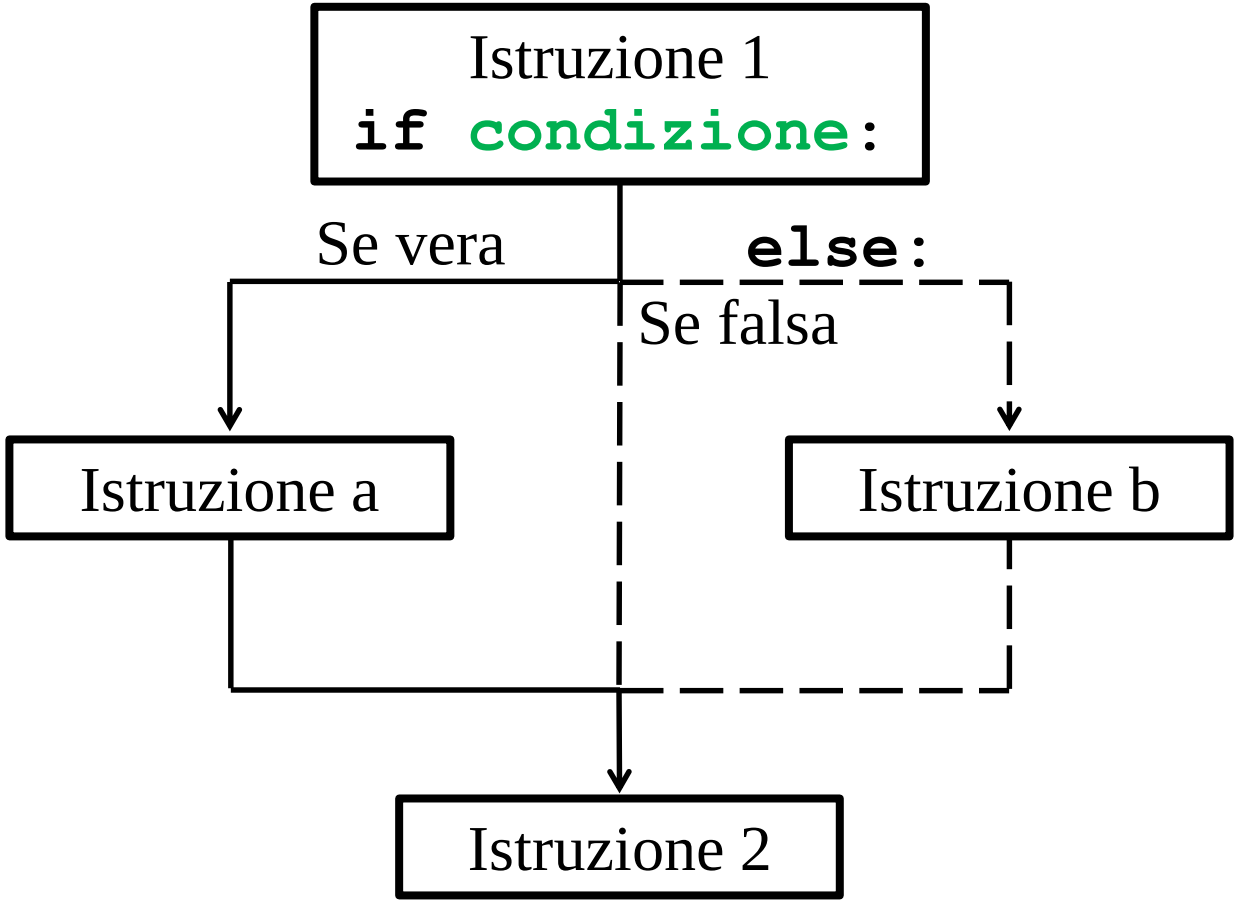
- Esse alterano il normale flusso sequenziale solo al verificarsi di una determinata condizione
- Il costrutto principale in Python è dato dalla coppia di parole-chiave **`if:…else:`**

```
if Condizione: #Se Condizione è vera
    #esegui le istruzioni indentate:
    …
[else: #altrimenti, se Condizione è falsa
    #esegui le istruzioni indentate:
    …]
```

- La parte **`else`** è **facoltativa**

Modello grafico nell'uso di `if`





Il costrutto `if:…else:…`

- Nel costrutto **`if:…else:…`** il termine indicato come `Condizione` è **un'espressione booleana** che è interpretata come **vera** se il suo valore è diverso da 0, altrimenti è **falsa**
- `Condizione` può essere quindi espressa come
  - combinazione degli operatori booleani noti (AND, OR, NOT) di variabili booleane (ovvero intere ma con significato booleano) oppure
  - espressioni di confronto determinate, ad esempio, dal valore corrente assunto da variabili numeriche o caratteri, oppure
  - altre forme più complesse (che vedremo più avanti)

Il primo esempio

- Scrivere un programma Python che, ricevendo da tastiera due valori interi, stampi su video qual è il maggiore fra essi.

```
#Programma che stampa il maggiore tra due valori interi

x=input("Dammi un numero:")
y=input("Dammi un altro numero:")

differenza = x - y

print("Il maggiore dei due valori è: ")

if differenza > 0:
   maggiore=x      #attenti all'indentazione!!!
else:
   maggiore=y      #attenti all'indentazione!!!

print maggiore
```

L’indentazione

```
if differenza > 0:
→print(x)
else:
→print(y)
```

- Tutte le istruzioni subordinate ad un dato evento od interne vanno tabulate == indentazione del codice
- L’indentazione in Python è fondamentale perché non solo rende leggibile il codice ma lo rende anche eseguibile!!!

Esercizio

- Scrivere un programma Python che, ricevendo in ingresso un valore intero non negativo, compreso tra 0 e 33, stampi a video:
  - “Valore non corretto” se il valore inserito è al di là del range consentito;
  - “Hai passato l’esame e puoi registrarlo!” se il valore inserito è compreso fra 21 e 33;
  - “Dovrai sostenere l’orale se vuoi passare”, se il valore è compreso fra 16 e 20;
  - “Mi dispiace.. sarà per un’altra volta…”, se il valore è minore od uguale a 15.

Soluzione

```
#Programma per conoscere l'esito dell'esame di Fondamenti di Informatica

voto=input("Inserire il voto.\n")

if((voto<0) or (voto>33)):
   print("Valore non corretto.\n")
else:
   if((voto>20) and (voto<=33)):
        print("Hai passato l\'esame e puoi registrarlo!\n")
   else:
        if((voto>=16) and (voto<21)):
                print("Dovrai sostenere l\'orale se vuoi passare.\n")
        else:
                print("Mi dispiace… sarà per un\'altra volta…\n")
```

Soluzione alternativa: uso di `elif`:

```
#Programma per conoscere l'esito dell'esame di Fondamenti di
   Informatica

voto=input("Inserire il voto.\n")

if((voto<0) or (voto>33)):
   print("Valore non corretto.\n")
elif ((voto>20) and (voto<=33)):
   print("Hai passato l\'esame e puoi registrarlo!\n")
elif ((voto>=16) and (voto<21)):
   print("Dovrai sostenere l\'orale se vuoi passare.\n")
else:
   print("Mi dispiace… sarà per un\'altra volta…\n")
```

Esercizio

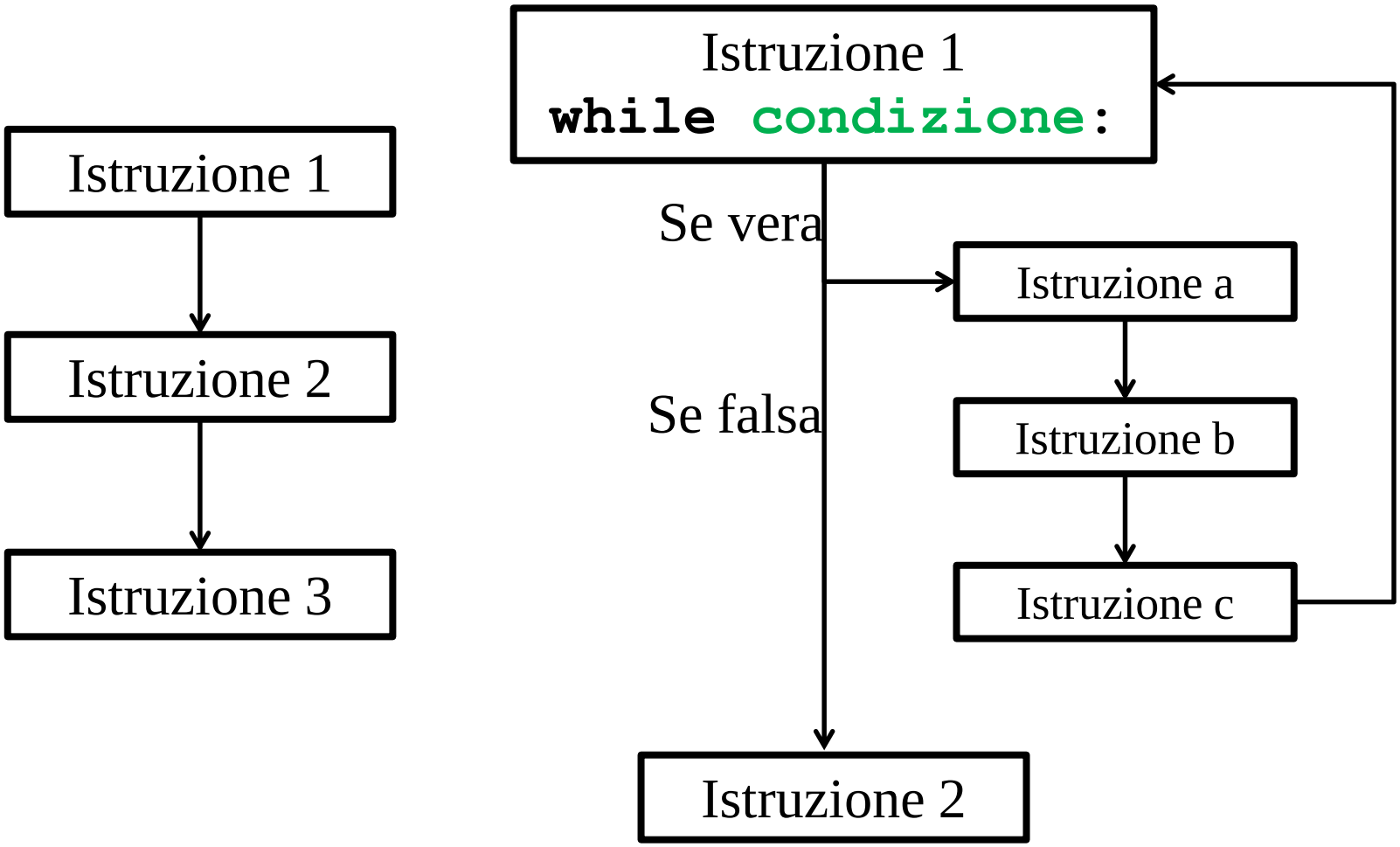
- Scrivere un programma Python che, ricevendo in ingresso una stringa che rappresenta il nome di una persona, scriva a video «Mi sei simpatico» se il nome finisce per «a», «Mi senti antipatico» se finisce per «e» o per «u», «Mi sei indifferente» in tutti gli altri casi.

Cicli (o “loop”): il costrutto `while`

- Laddove è richiesta la ripetizione iterativa di un certo gruppo di istruzioni (tipo Ripeti-Finché), in Python si adotta usualmente il costrutto **`while`**:

```
while (Condizione): #Finché Condizione è vera…
   #...esegui le istruzioni indentate:
   …
```

Modello grafico nell'uso di `while`



Esercizio

- Scrivere un programma Python che, letto da tastiera un valore intero n, stampi su video, su righe diverse, i valori da 0 a n.

```
n=input('Inserisci un numero.\n')
i=0
while i<=n:
    print i
    i=i+1     #rimuovendo questa istruzione…
              #...ciclo infinito!
```

Esercizio

- Scrivere un programma Python che legga una sequenza di numeri interi da tastiera e fermi la lettura al primo zero

```
#Programma che cerca il primo zero

valore=int(raw_input("Dammi un valore: "))

while valore!=0:      #ciclo
   valore=int(raw_input("Dammi un valore: "))

print("Ho trovato uno zero!\n")
```

Un esempio più interessante

- Scrivere un programma Python che calcoli la media aritmetica di una sequenza di valori reali non nulli progressivamente letti da tastiera, finché non viene inserito uno zero

Soluzione

```
#Programma per la media aritmetica

valore=float(raw_input("Dammi un valore: "))

n=0
media=0.0

while (valore!=0.0): #l'uso delle tonde è opzionale
        n=n+1
        media=media + valore
        valore=float(raw_input("Dammi un valore: "))

media = media / float(n)
print "La media dei valori inseriti è pari a " + str(media)

#analisi di correttezza: manca qualcosa in questo codice?
```

Soluzione corretta

```
#Programma per la media aritmetica

valore=float(raw_input("Dammi un valore: "))

n=0
media=0.0

while (valore!=0.0): #l'uso delle tonde è opzionale
        n=n+1
        media=media + valore
        valore=float(raw_input("Dammi un valore: "))
if n>0: #ORA VA MEGLIO
        media = media / float(n)
        print "La media dei valori inseriti è pari a " + str(media)
else:
        print "Nessun valore inserito."
```

Imparare a contare

- Scrivere un programma Python che calcoli la media aritmetica di una sequenza di **venti** valori reali non nulli progressivamente letti da tastiera.

Soluzione

```
#Programma per la media aritmetica su venti valori


n=20

i=0
media=0.0
while i<n: #l'uso delle tonde è opzionale
        valore = float(raw_input("Dammi un valore: "))
        media = media + valore
        i = i + 1

media = media / float(n)

print "La media dei valori inseriti è pari a " + str(media)
```

Esercitiamoci

- Scrivere un programma Python che legga **dieci** valori **interi** da tastiera e li immetta in una lista
- Se il valore inserito non è intero, non va inserito
- Stampi poi la lista andando a capo per ogni valore immesso a partire dall'ultimo.

Algoritmi in Python: un esempio completo

- Scrivere un programma Python che, ricevendo da tastiera un numero intero decimale, lo converta in un valore binario utilizzando **l'algoritmo delle divisioni successive** e lo stampi a video

L’algoritmo delle divisioni successive in “pseudo-codice”

- Ingresso (Input): una sequenza di cifre decimali X (il numero da convertire)
- Uscita (Output): una sequenza di n cifre binarie (il numero convertito) $\{b_{n-1}\ldots b_0\}$
- i = 0; (operatore “=“: assegnazione)
- **Ripeti**
  - Q = quoziente di X/2;
  - R = resto di X/2;
  - b_i = R;
  - X = Q;
  - i = i + 1;
- **Finché** Q <> 0
  - Nota “<>” significa “diverso da”

Analisi delle “strutture dati” utili

- Per implementare l’algoritmo con le conoscenze attuali:
  - Ci serviamo di una variabile (intera non negativa) che conservi il valore decimale, chiamiamola `X`
  - Un insieme di variabili intere che contenga i bit del numero convertito
    - Per questo scopo ci serviamo di un vettore di interi, `b`, di dimensione massima `N` predefinita
  - Per il calcolo del resto di `X/2`, che verrà memorizzato in un’altra variabile `R`, ci serviamo dell’operatore `%`
  - Terremo anche conto della lunghezza effettiva della stringa di bit memorizzandola nella variabile `M` (<=`N`)

Soluzione: inizializzazione di N e lettura di X da tastiera

```
#Programma per la conversione da decimale a binario*/
N=100 #il numero max di bit è 100=>max intero 2^N-1

X=int(raw_input("Inserire il valore da convertire.\n"))
```

Inizializzazione della variabile b

```
"""
Generazione ed inizializzazione a zero della
  stringa di bit.
E' necessaria perché all'atto della dichiarazione
  di b, i valori assegnati a ciascuna componente
  sono casuali
"""

b = [0] * N
```

Corpo dell'implementazione

```
i=0

while ((i<N) and (X!=0)):
  Q=X/2
  R=X%2  #due istruzioni rimpiazzabili
  b[i]=R #con b[i]=X%2
  X=Q
  i=i+1

M=i #M è il numero effettivo di bit <= N
```

Controllo della dimensione e stampa del risultato

```
#Bisogna verificare che N sia stato sufficiente a convertire il
   valore inserito

if(X!=0):
   print "Il valore inserito non può essere rappresentato con " +
   str(N) + " bit.\n"
else: #altrimenti si può stampare il valore convertito
   print "Il valore ottenuto richiede " + str(M) + "bit ed è:\n"
```

```
#attenzione all'indentazione!!!
i=M-1 #stampa in ordine inverso
stringa=""
while i>=0:
     stringa=stringa+str(b[i])
     i=i-1

print stringa+".\n"
```

```
print b[-1:-M-1:-1]
```

Esercizio

- Scrivere un programma Python che, letta una stringa `s` da tastiera, calcoli il numero di occorrenze di una sottostringa `ss` in `s`
- Nell'implementazione si ricordi che:
  - Una stringa si legge da tastiera con `raw_input()`
    - V 3.6 con `input()`
  - La lunghezza di una stringa si ottiene con `len()`
- E' proibito usare il metodo `count()`.

Soluzione

```
#Programma che calcola il numero di occorrenze di una sottostringa in una
   stringa

s=raw_input("Inserire una stringa:\n")
ss=raw_input("Inserire la sottostringa da cercare in \'"+s+ "\'.\n")

ls=len(s)        #lunghezza di s
lss=len(ss)      #lunghezza di ss

i=0
n=0              #variabile che conterrà il numero di occorrenze
while i<=ls-lss:         #avanzo fino a che i<=ls-lss
   if s[i:i+lss]==ss:    #lettura di un sottoinsieme di s
        n=n+1
   i=i+1

print "Il numero di occorrenze di " + ss + " in " + s + " è "+ str(n) +
   ".\n"
```

Il costrutto `for`

- Equivalente al costrutto **`while`**, viene utilizzato per iterare su strutture, come liste e dizionari, e qualunque variabile indicizzata, costruendo cicli ad iterazione «definita», ovvero con termine fisso non appena gli elementi del dato sono stati esaminati tutti:

```
for <iterazione su un dato iterabile>:
   #esegui le istruzioni indentate:
   …
```

Esempi

```
#iterazione su una lista con while
i=0
while i<len(lista):
    print lista[i]
    i=i+1

#iterazione su una lista con for
for x in lista:
    print x
```

Liste-indice

Per costruire una lista-indice, da usare con **`for`** in una modalità simile a **`while`**:

```
>>range(5)
[0, 1, 2, 3, 4]
>>range(1,5)
[1, 2, 3, 4]
>>range(1,6,2)
[1, 3, 5]
>>somma=0
>>for c in range(2, 11, 2):
        somma += c
>>>somma
30
```

```
indici=range(len(lista))
for i in indici:
    print lista[i]
```

Esercizio

- Scrivere un programma Python che, letta da tastiera una lista di caratteri `l` e una stringa `c`, generi una nuova stringa `s` data dalla concatenazione delle singole stringhe di `l`, utilizzando `c` come stringa separatrice.
- E' l'equivalente dell'assegnazione `s=c.join(l)`
  - Solo che dovete scrivere VOI il codice che lo fa.
- Esempio
  - Input: `l=["a","b","c"], c=" "`
  - Output: `s="a b c"`

Soluzione

```
l=input("Dammi una lista di stringhe:\n")
c=raw_input("Dammi una stringa separatrice:")

n=len(l)
s=""
for elemento in l[0:n-1]:
  s=s+elemento+c

s=s+l[-1]

print "Stringa ottenuta:\n" + s
```

Esercizio

- Scrivere un programma Python che, leggendo da tastiera **una stringa di caratteri** rappresentanti ciascuno una cifra binaria, lo converta nell'equivalente valore decimale e lo stampi a video
- Nota:
  - Il carattere meno significativo sia il primo della stringa. Esempio: la stringa immessa è '1100' ➔ valore decimale 3.

Ricordiamo l'algoritmo

- i=0;
- X = 0;
- Ripeti
  - X = X + $b_i * 2^i$
  - $i = i + 1$
- Finché i < N

Soluzione

```
#Programma per la conversione da binario a decimale

B=raw_input(“Inserire una numero binario a partire dal bit meno
   significativo:\n”)

i=0
valore=0
for b in B:
   valore = valore + int(b) * 2 ** i
   i=i+1

print(“\nIl valore binario ” + B + “ convertito risulta essere” +
   str(valore)+“.\n”)
```

- Implementare lo stesso codice usando:
  - `while`
  - `for` con lista-indice
- Re-implementare l’algoritmo considerando il primo bit inserito come quello più significativo

Esercizi con `for/while`

- Scrivere un programma Python che implementi l'algoritmo delle sottrazioni successive per ottenere quoziente e resto della divisione di un numero intero non negativo X con un altro intero non negativo Y e li stampi a video
  - **Senza** usare gli operatori / e %

- Scrivere un programma Python che, dati due valori interi non negativi X e Y, calcoli il valore X^Y e lo stampi a video
  - **Senza** usare l'operatore `**`

Esercizi

- Una certa applicazione rappresenta la parte frazionaria di un valore binario con una stringa tale che gl'indici negativi corrispondono alla posizione del relativo bit.
  - Per esempio, la stringa binaria '10' indica il valore decimale 0.25.

- Scrivere un programma che, letto da tastiera un valore frazionario binario secondo la descrizione di cui sopra, lo converta nel corrispondente decimale e lo stampi a video.

- Scrivere un programma che, letto da tastiera un valore frazionario decimale, lo converta nel corrispondente binario secondo la rappresentazione di cui sopra e lo stampi a video.

Esercizio con cicli e dizionari

- Scrivere un programma Python che, date due liste, una di chiavi, chiamata `K`, l'altra di valori, chiamata `V`, di uguale lunghezza, generi un dizionario `D` associando ogni elemento di `K` al corrispondente elemento di `V`
  - Esempio: `K=["a","b","c"], V=[1,2,3]` ➔ `D={"a":1,"b":2,"c":3}`
- Scrivere un programma Python che, dato un dizionario non vuoto `d`, ed una lista di chiavi `l`, stampi la sequenza di valori associati alle chiavi in `l` se presenti

«Intelligenza artificiale» e Python

- Scrivere un programma Python che chieda **iterativamente** all'interlocutore alla tastiera **l'inserimento di un nome** e stampi a video «E' un nome maschile» o «E' un nome femminile» a seconda del nome inserito.
- L'elenco dei nomi raggruppati per genere sono memorizzati **in un dizionario** le cui due chiavi sono appunto «maschile» e «femminile». A ciascuna chiave è associata **una lista** inizialmente vuota, destinata a contenere i nomi via via inseriti.
- Quando si inserisce un nome **non presente** nel dizionario, il sistema risponda con «Non conosco questo nome, dimmi se è un nome maschile o femminile». Dopo che l'interlocutore ha risposto da tastiera, lo aggiunga alla lista associata al relativo genere.
- Il programma termina quando si scrive la parola «fine» invece di un nome.

Per saperne di più

- K.A. Lambert, *Programmazione in Python*, Cap. 3, Apogeo