While - Esercizi

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

Università di Milano Bicocca

Multipli

- Scrivete uno script che richiede all'utente di inserire un numero da tastiera maggiore di 2. Non occorre verificare che l'utente abbia inserito un numero corretto.
- Il numero inserito dall'utente sarà utilizzato come limite superiore. Lo script dovra' visualizzare a video tutti i multipli del numero 7 oppure del numero 9 compresi tra 2 e il numero inserito dall'utente.

Multipli - Soluzione

```
## Multipli

ls = int(input('Limite superiore? '))
i=2
while i<=ls:
    if i%7==0 or i%9==0:
        print(i)
    i=i+1
```

Divisori in comune

- Scrivete uno script che accetta in ingresso due numeri e identifica tutti i divisori in comune.

Divisori in comune - Soluzione

```
1 n1 = int(input('Inserisci il 1. numero '))
2 n2 = int(input('Inserisci il 2. numero '))
3
4 i=1
5 while i<=n1 and i<=n2:
6     if n1%i==0 and n2%i==0:
7         print(i)
8     i=i+1 # i+=1
```

Somma sequenza

- Calcolate la somma di tutti i numeri compresi tra 1 e un limite superiore fornito dall'utente a tastiera.
- Nell'eseguire il calcolo, NON utilizzate la formula di Gauss, ma utilizzate contatori e cicli.
- Se volete potete poi utilizzare la formula di Gauss per verificare il risultato.

Somma sequenza - Soluzione

```
1 ls = int(input('Inserisci limite superiore '))
2 somma=0
3 i=1
4 while i<=ls:
5     somma=somma+i
6     i=i+1
7 print(somma)
8 print('Verifica con la formula di Gauss')
9 print( ls*(1+ls)/2.0 )
```

- Potete trovare maggiori informazioni sulla formula di Gauss qua.
- Formula di Gauss: $\sum_{i=1}^{n} i = n\frac{(1+n)}{2}$

Libreria per generare numeri casuali

In questo esercizio utilizzeremo una funzione della libreria python per generare numeri casuali. Una libreria è una collezione di codice scritto da altri che voi potete riutilizzare. Questi argomenti saranno ripresi con maggiore dettaglio nelle lezioni successive.

```
1  # Il comando seguente importa la funzione randint
2  # dalla libreria random, e' necessaria per poter
3  # generare dei numeri casuali
4  from random import randint
5
6  # Per 3 volte genero un numero casuale compreso
7  # tra 1 e 3 e lo stampo a video.
8  # Provate ad eseguire lo script per vedere cosa succede
9  print(randint(1,3))
10 print(randint(1,3))
11 print(randint(1,3))
```

```
3
1
2
```

Test generazione numeri casuali

- Le funzioni per generare dei numeri casuali dovrebbero produrre numeri con una distribuzione uniforme.
- Vi chiediamo di generare numeri casuali compresi tra 1 e 3 (estremi compresi) per un numero elevato di volte e di contare quante volte vengono generati ognuno dei 3 numeri.
- Visualizzate a video le numerosità dei numeri, la media e gli scostamenti in percentuale rispetto alla media.
- Provate ad eseguire lo script diverse volte, ogni volta aumentando la quantità di numeri generati casualmente, e osservate cosa succede.

Test generazione numeri casuali - Soluzione parte 1

```
n1=0
n2=0
n3=0
i=0
nprove=10000
while i<nprove:
    ncasuale=randint(1,3) # genero un numero casuale
    if ncasuale==1:
        n1=n1+1
    elif ncasuale==2:
        n2+=1
    elif ncasuale==3:
        n3=n3+1
    else:
        print('Numero fuori range') # non dovrebbe mai accadere
    i=i+1
print('Numerosita n. 1',n1)
print('Numerosita n. 2',n2)
print('Numerosita n. 3',n3)
```

. . . Continua nella slide successiva

Test generazione numeri casuali - Soluzione parte 2

```
...
20 print('Ora calcoliamo la media')
21
22 me=(n1+n2+n3)/3.0
23 print('Media '+str(me))
24
25 print('Scostamenti')
26 print('Variazione n1: %f percento' % ( (n1-me)/me*100) )
27 print('Variazione n2: %f percento' % ( (n2-me)/me*100) )
28 print('Variazione n3: %f percento' % ( (n3-me)/me*100) )
```

Triangolo di Floyd

Scrivete uno script python che visualizza a video le prime n righe del triangolo di Floyd. Il triangolo di Floyd ha la forma seguente:

```
1
2  3
4  5  6
7  8  9  10
11 12 13 14 15
```

Fate attenzione che il numero di valori presenti in una riga è pari all'indice della riga, per esempio: c'è un numero nella prima riga; due numeri nella seconda riga; ...
Suggerimento: usare una stringa come se fosse un contatore, es:

```
1 st='a'
2 st=st+'b'
3 print(st) # viene visualizzato 'ab'
```

Floyd - Soluzione

```
nr=4
riga=1
i=1
while riga<=nr:
    s='' # creo una stringa vuota
    k=0
    while k<riga: # il valore di riga ci dice anche quante colonne servono
        s=s+str(i)+'   ' # in alternativa, si puo' usare il comando seguente
        #s=s+ '%3d  ' % (i)
        i=i+1
        k=k+1
    print(s)
    riga=riga+1
```

%3d e' equivalente a %d, ma i numeri vengono sempre visualizzati con 3 caratteri, riempiendo eventualmente con spazi bianchi (se il numero non ha cifre a sufficienza). Al posto del 3 posso inserire un qualsiasi numero intero.

Floyd - Output

```
1
2    3
4    5    6
7    8    9    10
```

Massimo Comun Divisore (MCD) di due numeri

```
1  n1 = int(input('Primo numero? '))
2  n2 = int(input('Secondo numero? '))
3
4  if n1>n2:
5      nmag=n1
6      nmin=n2
7  else:
8      nmag=n2
9      nmin=n1
10
11 mcd=-1 #massimo comun divisore
12 i=nmin
13 while i>0 and mcd==-1: # potrebbe bastare mcd==-1
14     if n1%i==0 and n2%i==0:
15         mcd=i
16     i=i-1
17 print("L'MCD e': %d" % (mcd))
```

Minimo Comune Multiplo (mcm) di due numeri

```
1  n1 = int(input('Primo numero? '))
2  n2 = int(input('Secondo numero? '))
3
4  if n1>n2:
5      nmag=n1
6      nmin=n2
7  else:
8      nmag=n2
9      nmin=n1
10
11 i=nmag
12 mcm=-1
13 while i<=n1*n2 and mcm==-1: # nella peggiore delle ipotesi l'mcm e' n1*n2
14                            # potrei non mettere la condizione i<=n1*n2
15     if i%n1==0 and i%n2==0:
16         mcm=i
17     i=i+1
18 print("Il minimo comune multiplo e' "+str(mcm))
```