Autovalutazione - Esercizi

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

Università di Milano Bicocca

1. Stampa numeri dispari

- Scrivete uno script python che stampi a video i numeri dispari nell'intervallo [0, 2000].
- Dopo aver risolto il problema con un algoritmo, provate a risolvere di nuovo il problema con altri 2 algoritmi, diversi da quello/quelli che avete usato il precedenza.

Numeri dispari in [0, 2000] - Possibili soluzioni

- Soluzione 1

```
1  i=1
2  while i<2000:
3      print(i)
4      i=i+2
```

- Soluzione 2

```
10 i=1999
11 while i>0:
12     print(i)
13     i=i-2
```

- Soluzione 3

```
20 i=0
21 while i<=2000:
22     if i%2==1:
23         print(i)
24     i=i+1
```

2. Fattoriale

- Scrivere un programma che chieda all'utente un numero e ne calcoli il fattoriale
- Soluzione

```
1 n=int(input('Inserisci un numero '))
2 prod=1
3 i=n
4 while i>1:
5     prod=prod*i
6     i=i-1
7 print("Il fattoriale di %d e' %d" % (n,prod))
```

3. Verifica date

- Scrivere un programma che verifichi ripetutamente se una data inserita è plausibile. Il programma deve chiedere ogni volta all'utente se vuole procedere alla verifica di una nuova data, ed in caso positivo, deve richiedere di inserire (in formato numerico) giorno, mese e anno. Dopodiché, il programma deve verificare che le informazioni inserite siano plausibili (si trascurino gli anni bisestili). Per esempio 31/13/2016 non e' una data plausibile (non esiste il 13mo mese), oppure 31/11/2016 non è possibile (novembre ha solo 30 giorni). In caso di verifica positiva il programma deve costruire la stringa corrispondente alla data nel formato 'gg/mm/aaaa' e poi stampare a video questa stringa.

```
continua = True
while continua == True:
    gg=int(input('Giorno?'))
    mm=int(input('Mese? '))
    aa=int(input('Anno? '))
    controllo=True
    if mm<1 or mm>12:
        controllo=False
    elif gg<1 or gg>31:
        controllo=False
    elif (mm==4 or mm==6 or
          mm==9 or mm==11)
          and gg>30:
        controllo=False
    elif mm==2 and gg>28:
        controllo=False
    # Gia' che ci siamo
    elif aa<0:
        controllo=False
    if controllo==True:
    # ... continua ...
```

Nota: la colonna successiva riparte da riga 20

```
    #if controllo==True:#ripet.
        if gg<10:
            st='0'
        else:
            st=''
        st=st+str(gg)+'/'
        if mm<10:
            st=st+'0'
        st=st+str(mm)+'/'
        if aa<10:
            st=st+'000'
        elif aa<100:
            st=st+'00'
        elif aa<1000:
            st=st+'0'
        st=st+str(aa)
        print(st)
    else:
        print('Non plausib.')
    print('Continui (1: si)? ')
    n=int(input())
    if n!=1:
        continua = False
```

4. Stampa numeri in verticale

- Scrivete un programma che richiede in ingresso un numero intero e lo visualizza su piu' righe, una cifra per riga, a partire dalla cifra meno significativa fino alla cifra più significativa.
- Per esempio,

```
Numero? 1532
Output
2
3
5
1
```

- Soluzione

```
1 n=int(input('Inserisci un n. '))
2 print('Output')
3 while n>0:
4     r=n%10
5     print(r)
6     n = n // 10 # Risultato intero della divisione
```

4. Stampa numeri in verticale - soluzione alternativa

- Scrivete . . .
  dalla cifra meno significativa fino alla cifra più significativa.
- Per esempio,

```
Numero? 1532
Output
2
3
5
1
```

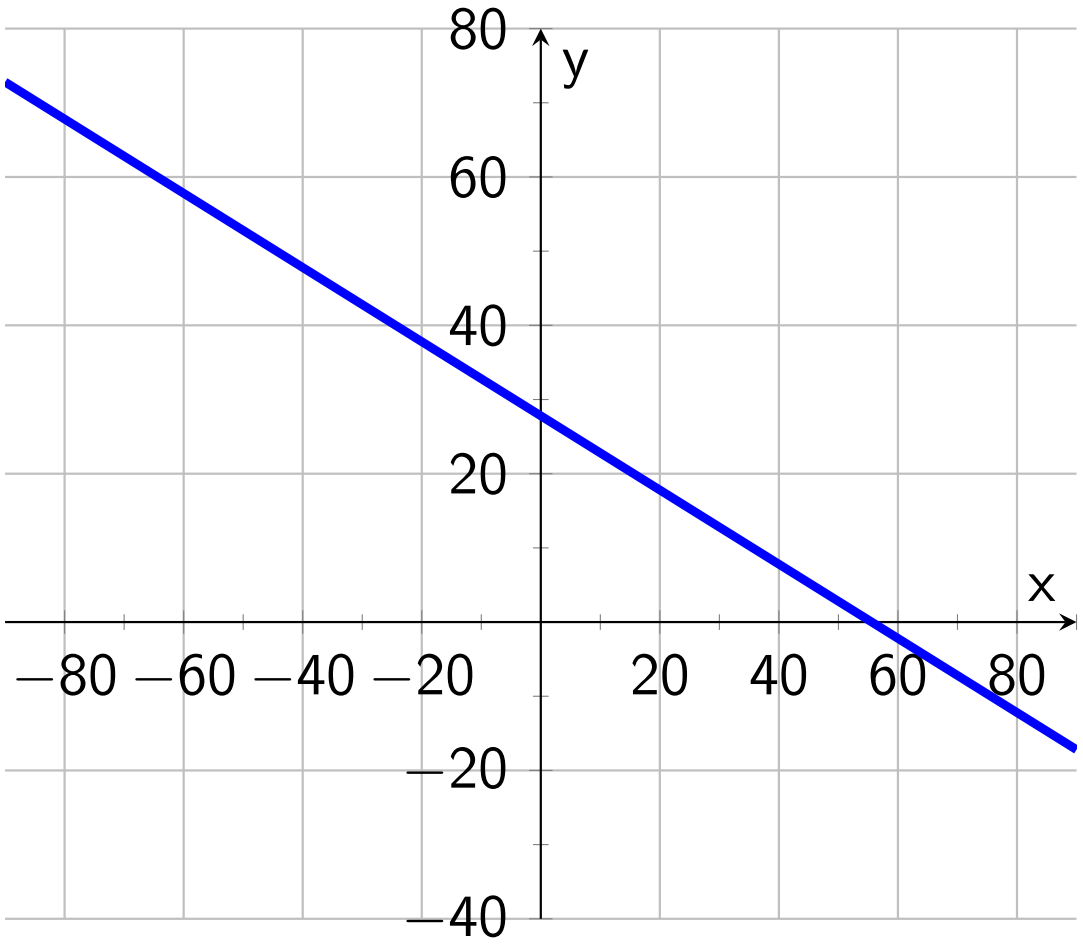
- Soluzione (fa uso delle strutture dati complesse)

```
1 st=str(input('Inserisci un n. '))
2 i = len(st)-1
3 print('Output')
4 while i>=0:
5     print(st[i])
6     i=i-1
```

Intersezione

- Scrivete un programma che individui l'intersezione (x_0, y_0) tra una retta data $y = -\frac{1}{2}x + 27.8$ e l'asse delle ascisse.
- Vi suggeriamo di utilizzare un algoritmo che sfrutti il metodo della bisezione. Tale metodo sfrutta la proprietà che tale intersezione si trova tra un punto avente ordinata positiva ed uno di ordinata negativa. Si consideri come intervallo di partenza l'intervallo $[0, 100]$ e come approssimazione accettabile il valore $\epsilon = 10^{-4}$ i.e., se $y_0 \in [-\epsilon, +\epsilon]$ allora x_0 è da considerarsi l'ascissa dell'intersezione.
- Non dovete necessariamente usare la bisezione, potete utilizzare anche un altro procedimento. Tuttavia vi chiediamo di trovare l'intersezione (x_0, y_0) utilizzando un algoritmo e non trovando matematicamente le soluzioni dell'equazione $-\frac{1}{2}\ x + 27.8 = 0$

Grafico



- $y = -\frac{1}{2}\, x + 27.8$

Intersezione

Scrivete un programma che individui l'intersezione (x_0, y_0) tra una retta data $y = -\frac{1}{2}x + 27.8$ e l'asse delle ascisse.

```
1  xsx=0
2  xdx=100
3  epsilon=10.0**-4 # =0.0001
4  med = (xsx+xdx)/2.0
5  while -0.5*med+27.8 < -epsilon or -0.5*med+27.8 > epsilon:
6      if -0.5*med+27.8 >0:
7          xsx=med
8      else:
9          xdx=med
10     med = (xsx+xdx)/2.0
11 print('Risultato',med) # ('Risultato', 55.5999755859375)
12 # -0.5*x+27.8=0; x=27.8/0.5
13 print('Verifica ',27.8/0.5) # ('Verifica ', 55.6)
```

Intersezione

La condizione del ciclo while

```
1 −0.5∗med+27.8 < −epsilon or −0.5∗med+27.8 > epsilon:
```

potrebbe essere riscritta nei modi seguenti:

- Sfruttando le proprietà di De Morgan
  - Reminder
  - not (A and B) = (not A) or (not B)
  - not (A or B) = (not A) and (not B)
  - (not $x > 5$) = $x \leq 5$

```
2 not (−epsilon <  −0.5∗med+27.8
3                  and −0.5∗med+27.8 < epsilon)
```

- Il linguaggio python permette di esprimere una condizione come la precedente usando un formalismo simile a quello usato in ambito matematico

```
4 not −epsilon <  −0.5∗med+27.8 < epsilon:
```

```
1  xsx=0
2  xdx=100
3  epsilon=10.0**-4
4  med = (xsx+xdx)/2.0
5  while not -epsilon <
6   -0.5*med+27.8 < epsilon:
7       print(xsx,xdx,med,...)
8       if -0.5*med+27.8 >0:
9           xsx=med
0       else:
1           xdx=med
2       med = (xsx+xdx)/2.0
3  print('Risultato',med)
4  print('Verifica ',
5              27.8/0.5)
```

xsx	xdx	med	f(med)
0.00000	100.00000	50.00000	2.80000
50.00000	100.00000	75.00000	-9.70000
50.00000	75.00000	62.50000	-3.45000
50.00000	62.50000	56.25000	-0.32500
50.00000	56.25000	53.12500	1.23750
53.12500	56.25000	54.68750	0.45625
54.68750	56.25000	55.46875	0.06563
55.46875	56.25000	55.85938	-0.12969
55.46875	55.85938	55.66406	-0.03203
55.46875	55.66406	55.56641	0.01680
55.56641	55.66406	55.61523	-0.00762
55.56641	55.61523	55.59082	0.00459
55.59082	55.61523	55.60303	-0.00151
55.59082	55.60303	55.59692	0.00154

```
('Risultato', 55.599975...
('Verifica ', 55.6)
```

Perché *Risultato* e l'ultimo valore della colonna *med* non coincidono? A causa dell'uscita dal ciclo . . . il nuovo valore di *med* è stampato solo a riga 13, la print a riga 7 non viene eseguita con l'ultimo valore.