Laboratorio 06

Programmazione - CdS Matematica

Michele Donini
03 dicembre 2013

Esercizio I

Esercizio

Scrivere una funzione `crypt` che cripta una stringa, passando un carattere per volta alla funzione (parametro) `f`.

Esempio di invocazione

```
def f(c, k = 3) :
    return " " if c == " " else chr(ord("a") + ((ord(c) -
        ord("a") + k) % 26))

def g(c, k = -3):
    return f(c, k)

s = "mia stringa"
crypt(s, f), crypt(s, g)
```

Soluzione I

```
def f(c, k = 3) :
    return # ...

def g(c, k = -3):
    return f(c, k)

def crypt(s, f):
    enc = ""
    for c in s:
        enc += f(c)
    return enc
```

```
>>> crypt("test crittazione", f)
'whvw fulwwdclrqh'
>>> crypt("whvw fulwwdclrqh", g)
'test crittazione'
```

Funzioni ricorsive

Funzioni che dipendono dal risultato della funzione stessa su valori "più semplici".

Funzioni ricorsive

Funzioni che dipendono dal risultato della funzione stessa su valori "più semplici".

Il fattoriale di un numero:

$$n! = \begin{cases} 1 & \text{se } n \leq 1 \rightarrow \text{ detto } \textit{caso base} \\ n(n-1)! & \text{se } n > 1 \end{cases}$$

Funzioni ricorsive

Funzioni che dipendono dal risultato della funzione stessa su valori "più semplici".

Il fattoriale di un numero:

$$n! = \begin{cases} 1 & \text{se } n \leq 1 \rightarrow \text{ detto } \textit{caso base} \\ n(n-1)! & \text{se } n > 1 \end{cases}$$

```
def fatt(n):
    if n <= 1 :
        return 1
    else:
        return n * fatt(n-1)
```

Funzioni ricorsive

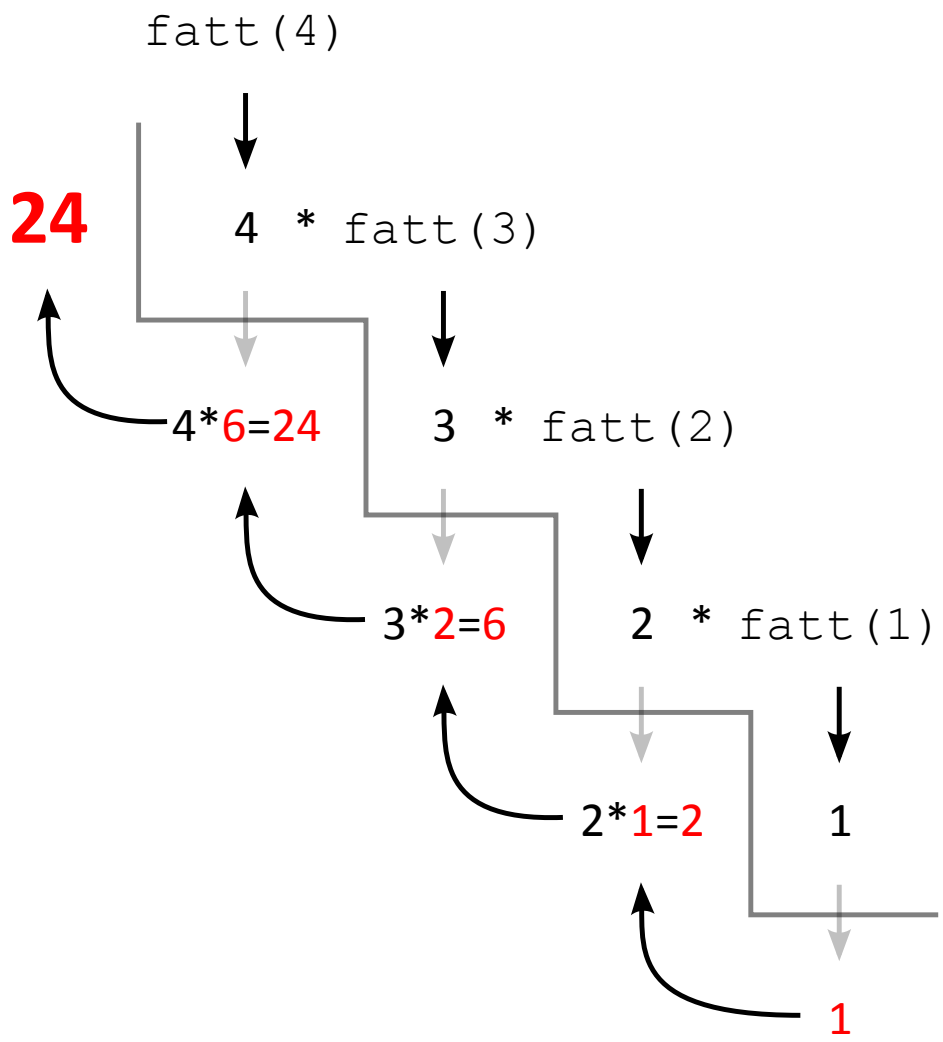
```
def fatt(n):
  if n <= 1 :
    return 1
  else:
    return n*fatt(n-1)
```

```
>>> fatt(4)
24
```

Funzioni ricorsive

```
def fatt(n):
  if n <= 1 :
    return 1
  else:
    return n*fatt(n-1)
```

```
>>> fatt(4)
24
```



Esercizio facile

Esercizio

Scrivere una funzione ricorsiva per il calcolo dell'n-esimo numero nella serie di Fibonacci. Dove $F_n = F_{n-1} + F_{n-2}$.
Si assume $F_1 = 0$ e $F_2 = 1$ (a volte anche $F_1 = 1$ e $F_2 = 1$).
La serie è definita come: $0, 1, 1, 2, 3, 5, 8, \ldots$

Esercizio facile

Esercizio

Scrivere una funzione ricorsiva per il calcolo dell'n-esimo numero nella serie di Fibonacci. Dove $F_n = F_{n-1} + F_{n-2}$.
Si assume $F_1 = 0$ e $F_2 = 1$ (a volte anche $F_1 = 1$ e $F_2 = 1$).
La serie è definita come: $0, 1, 1, 2, 3, 5, 8, \ldots$

```
def fib(n):
    if n == 1 :
        return 0
    elif n == 2 :
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

Esercizio

Esercizio

Riscrivere la funzione `chiedi_positivo()` senza usare **`while`**, ma sfruttando la ricorsione.

Esercizio

Esercizio

Riscrivere la funzione `chiedi_positivo()` senza usare **`while`**, ma sfruttando la ricorsione.

```
def chiedi_positivo():
    n = int(raw_input("Inserisci un intero positivo: "))
    if n < 0:
        return chiedi_positivo()
    return n
```

```
>>> chiedi_positivo()
Inserisci un intero positivo: -5
Inserisci un intero positivo: 10
10
```

Esercizio

Esercizio

Definire una funzione ricorsiva che, data una stringa, ritorna `True` se questa è palindroma, `False` altrimenti.
Non si può usare alcuna funzione delle stringhe, ad eccezione di `len` e dello *slicing* (non si può usare *striding*).

Esercizio

Esercizio

Definire una funzione ricorsiva che, data una stringa, ritorna `True` se questa è palindroma, `False` altrimenti.
Non si può usare alcuna funzione delle stringhe, ad eccezione di `len` e dello *slicing* (non si può usare *striding*).

```
def palindroma(s):
    if not s:
        return True
    elif s[0] == s[len(s)-1]:
        return palindroma(s[1:len(s)-1])
    else:
        return False
```

Esercizio

Estensione esercizio

Estendere l'esercizio precedente al trattamento di **frasi** palindrome (saltando spazi e segni di punteggiatura e la distinzione maiuscole/minuscole).

Esempi di frasi palindrome:

- O mordo tua nuora o aro un autodromo
- I topi non avevano nipoti
- Avida diva
- Amo Roma
- Ettore evitava le madame lavative e rotte
- Eran i mesi di seminare
- Occorre pepe per Rocco
- Etna gigante
- Ave, Eva!
- Alla bisogna tango si balla
- I tropici, mamma. Mi ci porti?
- Alle carte t'alleni nella tetra cella

Soluzione

Possibile soluzione

```
def palindroma(s):
    if not s:
        return True
    if s[0].isalpha() == False:        # nuovo caso
        return palindroma(s[1:len(s)])
    if s[-1].isalpha() == False:       # nuovo caso
        return palindroma(s[0:len(s)-1])
    elif s[0].lower() == s[len(s)-1].lower():
        return palindroma(s[1:len(s)-1])
    else:
        return False
```

Esercizio

Esercizio

Scrivere un generatore di frasi casuali (non di senso compiuto).

- Ciascuna `frase()` genera un `soggetto()` un `verbo()` e, se il verbo è transitivo, un `complemento()`
- Più frasi si possono concatenare con una `congiunzione()`
- Non si possono concatenare più di 4 frasi assieme
- Esecuzione di `frase()` devono produrre risultati differenti (spesso)

Soluzione

```
import random
```

Soluzione

```
import random

def soggetto():
    l = ["Pippo", "Pluto", "Paperino", "un robot"]
    return l[random.randint(0, len(l)-1)]
```

Soluzione

```
import random

def soggetto():
    l = ["Pippo", "Pluto", "Paperino", "un robot"]
    return l[random.randint(0, len(l)-1)]

def verbo():
    l = [("corre", False), ("dorme", False),
         ("mangia", True), ("lancia", True)]
    verbo = l[random.randint(0, len(l)-1)]
    if (verbo[1]):
        return verbo[0] + " " + complemento()
    return verbo[0]
```

Soluzione

```
import random

def soggetto():
    l = ["Pippo", "Pluto", "Paperino", "un robot"]
    return l[random.randint(0, len(l)-1)]

def verbo():
    l = [("corre", False), ("dorme", False),
         ("mangia", True), ("lancia", True)]
    verbo = l[random.randint(0, len(l)-1)]
    if (verbo[1]):
        return verbo[0] + " " + complemento()
    return verbo[0]

def complemento():
    l = ["il libro", "la porta", "un panino"]
    return l[random.randint(0, len(l)-1)]
```

Soluzione

```
def congiunzione():
    l = ["e", "ma", "mentre"]
    return l[random.randint(0, len(l)-1)]
```

Soluzione

```
def congiunzione():
    l = ["e", "ma", "mentre"]
    return l[random.randint(0, len(l)-1)]

def frase(n = 2):
    if (n == 0 or random.randint(0,9) > 6):
        return soggetto() + " " + verbo()
    else:
        return frase(n-1) + " " + congiunzione() + " " +
            frase(n-1)
```