DIPARTIMENTO DI INGEGNERIA E SCIENZE DELL'INFORMAZIONE E MATEMATICA – UNIVAQ

CORSO DI FONDAMENTI DI INFORMATICA

ESERCITAZIONE 5

PROF. GABRIELE DI STEFANO - DR. FRANCESCO GALLO

Blocco 0 - III Piano - francesco.gallo@univaq.it

Web: http://people.disim.univaq.it/~francesco.gallo/fondamenti.html

Come vengono passati i parametri in Python

```
def Esempio1(x):
    x = x + 1
    return x

x = 3
print("Stampa x = x + 1: {}".format(Esempio1(x)))
print("Stampa x: {}".format(x))
```

Qual e' il valore di **x**?

Come vengono passati i parametri in Python

```
def Esempio1(x):
    x = x + 1
    return x

x = 3
print("Stampa x = x + 1: {}".format(Esempio1(x)))
print("Stampa x: {}".format(x))
```

Qual e' il valore di **x**?

```
# Stampa x = x + 1: 4
```

Come vengono passati i parametri in Python

```
def Esempio1(x):
    x = x + 1
    return x

x = 3
print("Stampa x = x + 1: {}".format(Esempio1(x)))
print("Stampa x: {}".format(x))
```

Qual e' il valore di **x**?

```
# Stampa x = x + 1: 4
# Stampa x: 3
```

Come vengono passati i parametri in Python

```
def Esempio1(x):
    x = x + 1
    return x
```

Parametri Formali

```
x = 3
print("Stampa x = x + 1: {}".format(Esempio1(x)))
print("Stampa x: {}".format(x))
```

Parametri Attuali

Qual e' il valore di **x**?

```
# Stampa x = x + 1: 4
# Stampa x: 3
```

<u>Passaggio per copia (o per valore)</u>:
Implica che i parametri attuali vengono copiati nei parametri formali
e quindi la funzione lavora su una copia dei valori.

Come vengono passati i parametri in Python

```
def Esempio2(x, y):
    x.append(y)
    return x
```

Parametri Formali

```
x = [3, 2]
print("Stampa x.append(y): {}".format(Esempio2(x, 1)))
print("Stampa x: {}".format(x))
```

Parametri Attuali

Qual e' il valore di **x**?

Come vengono passati i parametri in Python

```
def Esempio2(x, y):
    x.append(y)
    return x
```

Parametri Formali

```
x = [3, 2]
print("Stampa x.append(y): {}".format(Esempio2(x, 1)))
print("Stampa x: {}".format(x))
```

Parametri Attuali

Qual e' il valore di **x**?

```
# Stampa x.append(y): [3, 2, 1]
```

Come vengono passati i parametri in Python

```
def Esempio2(x, y):
    x.append(y)
    return x
```

Parametri Formali

```
x = [3, 2]
print("Stampa x.append(y): {}".format(Esempio2(x, 1)))
print("Stampa x: {}".format(x))
```

Parametri Attuali

Qual e' il valore di **x**?

```
# Stampa x.append(y): [3, 2, 1]
# Stampa x: [3, 2, 1]
```

Come vengono passati i parametri in Python

```
def Esempio2(x, y):
    x.append(y)
    return x
```

Parametri Formali

```
x = [3, 2]
print("Stampa x.append(y): {}".format(Esempio2(x, 1)))
print("Stampa x: {}".format(x))
```

Parametri Attuali

Qual e' il valore di **x**?

```
# Stampa x.append(y): [3, 2, 1]
# Stampa x: [3, 2, 1]
```

Passaggio per riferimento:

Implica che le variabili locali contengono un riferimento al parametro attuale. Questo può accadere solo quando vengono passati a una funzione oggetti mutabili e quando la funzione modifica l'oggetto che e' stato passato.

Matrici come liste di liste

Come possiamo rappresentare una matrice $A^{r \times c}$ in Python

$$A^{3x2} = \begin{matrix} 0 & 1 \\ 3 & 2 \\ 5 & 6 \end{matrix}$$

può essere rappresentata come una lista di liste.

A = [[0, 1], [3, 2], [5, 6]] **len(A) = 3**, **len(A[0]) = 2**

Matrici come liste di liste

Come possiamo rappresentare una matrice $A^{r \times c}$ in Python

$$A^{3x2} = \begin{matrix} 0 & 1 \\ 3 & 2 \\ 5 & 6 \end{matrix}$$

può essere rappresentata come una lista di liste.

A = [[0, 1], [3, 2], [5, 6]] **len(A) = 3**, **len(A[0]) = 2**

ATTENZIONE!!!

- [1, 2, 3] e' una lista di numeri, **MA NON UNA MATRICE**
- [[1, 2, 3]] e' un **vettore riga**, **CIOE' UNA MATRICE**
- [[1], [2], [3]] e' un **vettore colonna**, **CIOE' UNA MATRICE**

Esercizi

1. Scrivere un metodo che, dati una lista di interi **a** ed un intero **n**, restituisce la posizione della prima occorrenza di **n** in **a**, e **-1** se **n** non compare in **a**.

2. Scrivere un metodo che, date due liste **a** e **b** di interi, restituisce **true** se tutti gli elementi della lista **b** compaiono nella lista **a** nello stesso ordine in cui compaiono in **b**, altrimenti il metodo restituisce **false**.
a = {-5, 4, 7, -1, 10, 21, 9, -7} e b = {4, -1, 9, -7} restituisce **true**

3. Scrivere un metodo che prende come parametro una matrice e restituisce il numero delle sue righe

4. Scrivere un metodo che prende come parametro una matrice e restituisce il numero delle sue colonne

5. Scrivere un metodo che calcola e restituisce la trasposta di una matrice presa come parametro.

6. Scrivere una funzione che, presi come parametri due liste moltiplicabili, calcola e restituisce il prodotto scalare, oppure `None`.

7. Definire un metodo che restituisce la matrice prodotto tra due matrici prese come parametri (se non sono moltiplicabili, stampa un messaggio di errore e restituisce `None`)

Esercizi

1. Scrivere un metodo che, dati una lista di interi **a** ed un intero **n**, restituisce la posizione della prima occorrenza di **n** in **a**, e **-1** se **n** non compare in **a**.

Esercizi

1. Scrivere un metodo che, dati una lista di interi **a** ed un intero **n**, restituisce la posizione della prima occorrenza di **n** in **a**, e **-1** se **n** non compare in **a**.

```
def posizione(a, n):
    i = 0
    trovato = False

    while i < len(a) and not trovato:
        if a[i] == n:
            trovato = True
        else:
            i = i + 1
    if trovato:
        return i
    else:
        return -1

a = [3, 2, 7, 9, 10]
n = 10

print(posizione(a, n))
```

Esercizi

2. Scrivere un metodo che, date due liste **a** e **b** di interi, restituisce T**rue** se tutti gli elementi della lista **b** compaiono nella lista **a** nello stesso ordine in cui compaiono in **b**, altrimenti il metodo restituisce F**alse**.
a = [-5, 4, 7, -1, 10, 21, 9, -7] e b = [4, -1, 9, -7] restituisce **true**

Esercizi

2. Scrivere un metodo che, date due liste **a** e **b** di interi, restituisce **true** se tutti gli elementi della lista **b** compaiono nella lista **a** nello stesso ordine in cui compaiono in **b**, altrimenti il metodo restituisce **false**.
a = {-5, 4, 7, -1, 10, 21, 9, -7} e b = {4, -1, 9, -7} restituisce **true**

```
def ordineArray(a, b):
    i = 0
    j = 0

    while j < len(b) and i < len(a):
        if b[j] == a[i]:
            i = i + 1
            j = j + 1
        else:
            i = i + 1

    if j == len(b):
        return True
    else:
        return False

a = [5,4,7,-1,10,21,9,-7]
b = [4,-1,9,-7]

print(ordineArray(a, b))
```

Esercizi

3. Scrivere un metodo che prende come parametro una matrice e restituisce il numero delle sue righe

```
def righe(M):
    return len(M)

M = [[1,2,3],[4,5,6]]

print(righe(M))
```

Esercizi

4. Scrivere un metodo che prende come parametro una matrice e restituisce il numero delle sue colonne

Esercizi

4. Scrivere un metodo che prende come parametro una matrice e restituisce il numero delle sue colonne

```
def colonne(M):
    return len(M[0])

M = [[1,2,3],[4,5,6],[7,8,9]]

print(colonne(M))
```

Esercizi

5. Scrivere un metodo che calcola e restituisce la trasposta di una matrice presa come parametro.

Esercizi

5. Scrivere un metodo che calcola e restituisce la trasposta di una matrice presa come parametro.

```
from Esercizio3 import righe
from Esercizio4 import colonne

def trasposta(M):
    MT = []
    r = righe(M)
    if r == 0:
        return MT
    else:
        c = colonne(M)
        for i in range(c):
            col = []
            for j in range(r):
                col.append(M[j][i])
            MT.append(col)
        return MT

M = [[1,2,3],[4,5,6],[7,8,9]]

print(trasposta(M))
```

Esercizi

6. Scrivere una funzione che, presi come parametri due liste moltiplicabili, calcola e restituisce il prodotto scalare, oppure `None`.

Esercizi

6. Scrivere una funzione che, presi come parametri due liste moltiplicabili, calcola e restituisce il prodotto scalare, oppure `None`.

```python
def molt_liste(U, V):
    if len(U) == len(V):
        pr_sc = 0
        for i in range(len(V)):
            pr_sc = pr_sc + (U[i]*V[i])
        return pr_sc
    else:
        return None
```

Esercizi

7. Definire un metodo che restituisce la matrice prodotto tra due matrici prese come parametri (se non sono moltiplicabili, stampa un messaggio di errore e restituisce `None`)

Esercizi

7. Definire un metodo che restituisce la matrice prodotto tra due matrici prese come parametri (se non sono moltiplicabili, stampa un messaggio di errore e restituisce `None`)

```python
from Esercizio3 import righe
from Esercizio4 import colonne
from Esercizio5 import trasposta
from Esercizio6 import molt_liste

def molt_mat(A, B):
    if colonne(A) != righe(B):
        print("Matrici non moltiplicabili")
        return None
    else:
        AB = []
        BT = trasposta(B)
        for i in range(righe(A)):
            riga = []
            for j in range(righe(BT)):
                ij = molt_liste(A[i], BT[j])
                riga.append(ij)
            AB.append(riga)
        return AB

M = [[1,2,3],[4,5,6]]
N = [[7, 8], [9, 10]]

print(molt_mat(M, N))
```