Esercizi di Fondamenti di Informatica

Andrea Gussoni
andrea1.gussoni at polimi.it

Politecnico di Milano

November 14, 2019

Table of Contents

1 Ripasso Esercitazione Precedente

2 Esercizi Vari

3 Funzioni

Section 1

Ripasso Esercitazione Precedente

Ripasso

Problemi con esercizi della precedente esercitazione?

- Costrutti ciclico (while/for)
- Esercizi con stringhe
- Algoritmo di ordinamento

Ripasso

Problemi con esercizi della precedente esercitazione?

- Costrutti ciclico (while/for)
- Esercizi con stringhe
- Algoritmo di ordinamento

Ripasso

Problemi con esercizi della precedente esercitazione?

- Costrutti ciclico (while/for)
- Esercizi con stringhe
- Algoritmo di ordinamento

Section 2

Esercizi Vari

Pattern a Scacchiera

Ideare un programma che accetti come input un intero n, e stampi come output una matrice che presenti un pattern *a scacchiera*, ovvero che abbia su ogni riga e ogni colonna un'alternanza di 0 e 1.

Pattern a Scacchiera I

Possibile soluzione:

```
n = int(input("Inserisci grandezza griglia"))

for riga in range(0, n):
    for col in range(0, n):

        # Usiamo una flag booleana per capire se
        # siamo in una riga pari o dispari
        riga_pari = False
        if riga % 2 == 0:
            riga_pari = True

        # Alterniamo uno 0 e 1 zero a seconda della
        # colonna
        if riga_pari:
```

Pattern a Scacchiera II

```
            if col % 2 == 0:
                print("0", end="")
            else:
                print("1", end="")
        else:
            if col % 2 == 0:
                print("1", end="")
            else:
                print("0", end="")

    # Alla fine di ogni riga, andiamo a capo
    print("")
```

Pattern a Scacchiera

Possibile soluzione:

```
n = int(input("Inserisci grandezza griglia"))

for riga in range(0, n):
    for col in range(0, n):

        # Alterniamo un 1 e 1 zero a seconda della
        # somma degli indici di riga e colonna
        if (riga + col) % 2 == 0:
            print("0", end="")
        else:
            print("1", end="")

    # Alla fine di ogni riga, andiamo a capo
    print()
```

Quiz

Scrivere un programma che effettui il seguente quiz:

- Una prima domanda, a risposta secca, che chieda se 7 è un numero primo.
- Una seconda domanda, che chieda di inserire un numero pari. Questa domanda verrà ripetuta finchè l'utente non soddisferà il requisito.
- Una terza domanda che chieda di inserire il valore di pi greco, fino alla 5 cifra decimale. In questo caso l'utente avrà a disposizione 5 tentativi, ed ad ogni tentativo il punteggio assegnato alla risposta diminuirà (partendo da 5 e arrivando a 0).
- Una coppia di domande interdipendenti, nelle quali all'utente è richiesto prima di esprimere una preferenza tra quadrato e triangolo, e che in base alla scelta effettuata chiederà di inserire i valori di lato o base e altezza, con la relativa area associata, e verificherà la correttezza della risposta.
- Se non specificato altrimenti, per ogni risposta corretta verrà assegnato un punto.
- Alla fine del quiz sarà mostrato all'utente il punteggio ottenuto.

Quiz I

Possibile soluzione:

```
print("Rispondi alle seguenti domande:")

risp = input("7 e' un numero primo? s/n")
if risp == "y":
    voto1 = 1
else:
    voto1 = 0

voto2 = 0
while voto2 != 1:
    risp = int(input("Inserisci un numero pari?"))
    if risp % 2 == 0:
        voto2 = 1
```

Quiz II

```
voto3 = 0
for i in range (5, 0, -1):
    risp = float(input("Inserisci il valori di pigreco fino
                        alla 5 cifra decimale"))
    if risp == 3.14159:
        voto3 = i
        break

voto4 = 0
risp = input("Preferisci il quadrato o il triangolo? q/t")
if risp == "q":
    lato = int(input("Inserisci la lunghezza del lato"))
    area = int(input("Inserisci il valore dell'area"))
    if area == lato ** 2:
        voto4 = 1
elif risp == "t":
```

Quiz III

```
    base = int(input("Inserisci la lunghezza della base"))
    altezza = int(input("Inserisci la lunghezza dell'altezza"))
    area = int(input("Inserisci il valore dell'area"))
    if area == base * altezza / 2:
        voto4 = 1

print("Il risultato ottenuto e'", voto1 + voto2 + voto3 + voto4)
```

Section 3

Funzioni

Funzioni

- Una funzione è una sequenza di istruzioni ben specifiche etichettate con un nome.
- Il vantaggio delle funzioni è che permettono di riutilizzare una porzione di codice in molti punti senza dover duplicare il codice stesso.
- Il modo più comune di visualizzare una funzione è pensare ad essa come ad una *black box*:

Funzioni

- Una funzione può accettare uno o più parametri, che possiamo pensare come delle variabili che saranno poi disponibili all'interno del corpo della funzione stessa.
- Una funzione inoltre ha generalmente un valore di ritorno. Questo valore di ritorno è in sostanza un valore che rappresenta l'esito della computazione della funzione.
- Abbiamo già usato diverse funzioni, e.g., la funzione `len` per le stringhe.
- In quel caso, il parametro era la stringa stessa, e il valore di ritorno la lunghezza della stringa.
- Attenzione alla nomenclatura: differenza tra parametro (argomento formale) e argomento (parametro attuale).

Funzioni

- Dobbiamo fare attenzione al fatto che i parametri e le variabili usate o definite all'interno delle funzioni non sono disponibili al loro esterno.
- In particolare, per ora ci basta tenere a mente che quando una funzione viene invocata, viene effettuata una copia dei parametri attuali in quelli formali.
- In questo modo, la funzione opererà su una nuova copia di tutti i parametri.

Funzioni e Parametri

```
def somma(totale, addendo):
    totale = totale + addendo
    return totale

totale = 0
addendo = 0
while addendo != -1:
    somma(totale, addendo)
    addendo = int(input("Inserisci addendo"))

print(totale)
```

Cosa stampa il programma se diamo in input i valori 1, 2, 5 e -1?

Funzioni e Parametri

```
def somma(totale, addendo):
    totale = totale + addendo
    return totale

totale = 0
addendo = 0
while addendo != -1:
    somma(totale, addendo)
    addendo = int(input("Inserisci addendo"))

print(totale)
```

Cosa stampa il programma se diamo in input i valori 1, 2, 5 e -1?
0

Funzioni e Parametri

```
def somma(totale, addendo):
    totale = totale + addendo
    return totale

totale = 0
addendo = 0
while addendo != -1:
    totale = somma(totale, addendo)
    addendo = int(input("Inserisci addendo"))

print(totale)
```

Cosa stampa il programma se diamo in input i valori 1, 2, 5 e -1?

Funzioni e Parametri

```
def somma(totale, addendo):
    totale = totale + addendo
    return totale

totale = 0
addendo = 0
while addendo != -1:
    totale = somma(totale, addendo)
    addendo = int(input("Inserisci addendo"))

print(totale)
```

Cosa stampa il programma se diamo in input i valori 1, 2, 5 e -1?
8

Funzioni e Valori Restituiti

- Un errore comune è anche dimenticarsi dei valori di ritorno per alcuni percorsi di una funzione.

Funzioni e Valori Restituiti

```
def area_triangolo(base,altezza):
    if base >= 0 and altezza >= 0:
        return base*altezza/2

base = int(input("Inserisci base"))
altezza = int(input("Inserisci altezza"))

print(area_triangolo(base, altezza))
```

Cosa succede se passiamo alla funzione i valori -1 e -2?

Funzioni e Valori Restituiti

```
def area_triangolo(base,altezza):
    if base >= 0 and altezza >= 0:
        return base*altezza/2

base = int(input("Inserisci base"))
altezza = int(input("Inserisci altezza"))

print(area_triangolo(base, altezza))
```

Cosa succede se passiamo alla funzione i valori -1 e -2?
None

Funzioni e Valori Restituiti

- Un errore comune è anche dimenticarsi dei valori di ritorno per alcuni percorsi di una funzione.
- Se in una funzione esiste un percorso di esecuzione che non è terminato da uno statement di `return` ben definito la funzione restituirà automaticamente il valore speciale None.

Funzioni e Valori Restituiti

```
def area_triangolo(base,altezza):
    if base >= 0 and altezza >= 0:
        return base*altezza/2
    else:
        return 0

base = int(input("Inserisci base"))
altezza = int(input("Inserisci altezza"))

print(area_triangolo(base, altezza))
```

Cosa succede se passiamo alla funzione i valori -1 e -2?

Funzioni e Valori Restituiti

```
def area_triangolo(base,altezza):
    if base >= 0 and altezza >= 0:
        return base*altezza/2
    else:
        return 0

base = int(input("Inserisci base"))
altezza = int(input("Inserisci altezza"))

print(area_triangolo(base, altezza))
```

Cosa succede se passiamo alla funzione i valori -1 e -2?
0

Funzioni e Valori Restituiti

- Un errore comune è anche dimenticarsi dei valori di ritorno per alcuni percorsi di una funzione.
- Se in una funzione esiste un percorso di esecuzione che non è terminato da uno statement di `return` ben definito la funzione restituirà automaticamente il valore speciale None.
- Per evitare di avere molteplici istruzioni di ritorno in una funzione, si può cercare di ristrutturare il codice in modo che un solo `return` sia presente, solitamente tramite l'uso di variabili aggiuntive.

Funzioni e Valori Restituiti

```
def area_triangolo(base,altezza):
    if base >= 0 and altezza >= 0:
        area = base*altezza/2
    else:
        area = 0
    return area

base = int(input("Inserisci base"))
altezza = int(input("Inserisci altezza"))

print(area_triangolo(base, altezza))
```

Funzione `main`

- Fino ad ora abbiamo scritto i nostri programmi semplicemente all'interno del nostro editor.
- Una buona norma è incapsulare il codice principale (quello che vogliamo sia eseguito quando l'esecuzione del nostro programma parte) a sua volta dentro una funzione speciale.
- Questa funzione è convenzionalmente chiamata funzione `main`.
- Dobbiamo ricordarci di invocare la funzione main per fare partire l'esecuzione.

Funzione main

Il codice:

```
for i in range (0,10):
  print(i)
```

Diventerà:

```
def main:
  for i in range (0,10):
    print(i)

main()
```

Prime Funzioni

Progettare un programma che accetti come input un orario, sotto la forma di ora e minuti. Il programma accetterà solo valori validi per ora e minuti. Per farlo usare le funzioni in modo da riutilizzare le parti in comune del codice.

Prime Funzioni I

```
def main():
    print("Inserire un orario, prima l'ora e poi i minuti")
    ora = numero_tra(0,24)
    minuto = numero_tra(0,59)
    print("L'ora inserita e' " + str(ora) + ":" + str(minuto))

def numero_tra(min, max):
    valore = int(input("Inserire un valore tra " + str(min)
                       + " e " + str(max)))
    while valore < min or valore > max:
        print("Il valore immesso e' fuori dall'intervallo consentito")
        valore = int(input("Inserire un valore tra " + str(min)
                           + " e " + str(max)))
    return valore
```

Prime Funzioni II

```
main()
```

Password Casuale

Ideare un programma che si occupi della generazione di una password casuale. La password dovrà avere i seguenti requisiti:

- Chiedere all'utente da quanti caratteri dovrà essere composta.
- Avere una cifra numerica al suo interno.
- Avere un carattere speciale ($+ - */?!@\#\%\&$) al suo interno.

Password Casuale I

Il codice:

```
from random import randint

def main():
    l = int(input("Lunghezza password"))
    password = genera(l)
    print(password)

def genera(n):
    password = ""
    for i in range (0, n - 2):
        password = password + casuale("abcdefghijklmnopqrstuvwxyz")

    numero = casuale("0123456789")
    password = inserisci(password, numero)
```

Password Casuale II

```
    carattere_speciale = casuale("+-*/?!@#$%&")
    password = inserisci(password, carattere_speciale)

    return password

def casuale(input):
    n = len(input)
    r = randint(0, n - 1)
    return input[r]

def inserisci(stringa, char):
    n = len(stringa)
    r = randint(0, n)
    nuova = ""
```

Password Casuale III

```
    for i in range(r):
        nuova = nuova + stringa[i]

    nuova = nuova + char

    for i in range(r, n):
        nuova = nuova + stringa[i]

    return nuova

main()
```