Esercizi di Fondamenti di Informatica

Andrea Gussoni
andrea1.gussoni at polimi.it

Politecnico di Milano

December 12, 2019

Table of Contents

Section 1

Ripasso Esercitazione Precedente

Ripasso

Problemi con esercizi della precedente esercitazione?

- Game Of Life

Copia File

Ideare un programma che effettui la copia di un file nell'altro, elaborando una riga per volta.

Copia File I

Una possibile soluzione

```
def copia_file(sorgente, destinazione):
    file_sorgente = open(sorgente, 'r')
    file_destinazione = open(destinazione, 'w')

    riga = file_sorgente.readline()
    while len(riga) != 0:
        file_destinazione.write(riga)
        riga = file_sorgente.readline()

    file_sorgente.close()
    file_destinazione.close()
```

Copia File II

```
def main():
    nome_file_sorgente = 'sorgente.txt'
    nome_file_destinazione = 'destinazione.txt'

    copia_file(nome_file_sorgente, nome_file_destinazione)

main()
```

Estrazioni

Scrivere un programma che legge un file contenente i nomi delle ruote del gioco del lotto e ne crei un altro contenente l'estrazione di 5 numeri casuali per ogni ruota del lotto.

Estrazioni I

Una possibile soluzione

```
from random import randint

def estrazione_ruota():
    numeri = []
    for i in range(0,5):
        numero = randint(1, 90)
        numeri.append(numero)

    stringa_numeri = ''
    for numero in numeri:
        stringa_numeri = stringa_numeri + ' ' + str(numero)

    return stringa_numeri
```

Estrazioni II

```
def main():
    file_ruote = open('ruote.txt', 'r')
    file_estrazioni = open('estrazioni.txt', 'w')
    for riga in file_ruote:
        nuova_riga = riga.rstrip('\n') + estrazione_ruota() + '\n'
        print(nuova_riga)
        file_estrazioni.write(nuova_riga)

    file_ruote.close()
    file_estrazioni.close()

main()
```

Estrazioni I

Una possibile soluzione

```
from random import randint

def estrazione_ruota():
    numeri = []
    while len(numeri) < 5:
        numero = randint(1, 90)
        if numero not in numeri:
            numeri.append(numero)

    stringa_numeri = ''
    for numero in numeri:
        stringa_numeri = stringa_numeri + ' ' + str(numero)

    return stringa_numeri
```

Estrazioni II

```
def main():
    file_ruote = open('ruote.txt', 'r')
    file_estrazioni = open('estrazioni.txt', 'w')
    for riga in file_ruote:
        nuova_riga = riga.rstrip('\n') + estrazione_ruota() + '\n'
        print(nuova_riga)
        file_estrazioni.write(nuova_riga)

    file_ruote.close()
    file_estrazioni.close()

main()
```

Estrazioni

Usando il file delle estrazioni precedentemente creato, calcolare i numeri non estratti su nessuna ruota, e il numero di ripetizioni dei numeri estratti.

Estrazioni I

Una possibile soluzione

```
def conta_occorenze(lista, numero):
    occorenze = 0
    for elem in lista:
        if elem == numero:
            occorenze = occorenze + 1
    return occorenze

def leggi_tutti_i_numeri(nome_file):
    file = open(nome_file, 'r')
    lista_numeri = []
    for riga in file:
        lista_riga = riga.split()
        lista_numeri += lista_riga[1:]
    file.close()
```

Estrazioni II

```
    return lista_numeri

def main():
    lista_numeri = leggi_tutti_i_numeri('estrazioni.txt')
    for i in range(1, 91):
        if str(i) not in lista_numeri:
            print(format(i, '2d'), ' non e' stato estratto')

    set_numeri = set(lista_numeri)
    for numero in set_numeri:
        r = conta_occorenze(lista_numeri, numero)
        print(format(int(numero), '2d'), ' estratto: ',
              format(r, '2d'), ' volte.')

main()
```

Esplora Cartella

Ideare un programma che utilizzando le funzioni messe a disposizione della libreria os di Python listi tutte le GIF presenti nella cartella dove il programma viene lanciato, stampando anche il contenuto di tutte le sue eventuali sottocartelle.

Esplora Cartella I

Una possibile soluzione

```
import os

print("File Immagini:")

# Otteniamo tutto il contenuto della cartella corrente.
cartella_corrente = os.getcwd()
contenuto = os.listdir()
for nome in contenuto :

    # Se stiamo ispezionando una cartella, esploriamo
    # ricorsivamente il suo contenuto.
    if os.path.isdir(nome) :
        for nome_2 in os.listdir(nome) :
            path_file = os.path.join(nome, nome_2)
```

Esplora Cartella II

```
        # Se il nome del file finisce per .gif, allora stampalo.
        if os.path.isfile(path_file) and nome_2.endswith(".gif") :
            print(os.path.join(cartella_corrente, path_file))

# Se, altrimenti, siamo in presenza diretta di un file,
# stampiamo semplicemente il suo percorso.
elif nome.endswith(".gif") :
    print(os.path.join(cartella_corrente, nome))
```

Analisi Dataset

Ideare un programma che analizzi un set di files di tipo .csv contente i record di registrazioni della qualità dell'aria relativi a più anni, e che stampi in output i valori minimi e massimi ottenuti dalle rilevazione per ciascun anno. I dataset sono disponibili qui [1]

[1] https://www.dati.lombardia.it/stories/s/auv9-c2sj

Analisi Dataset I

Una possibile soluzione

```
from csv import reader, writer
import os

def calcola_qualita_aria(percorso_file):
    dataset = open(percorso_file)
    csv_input = reader(dataset)

    # Saltiamo la prima riga, che e' l'intestazione
    next(csv_input)

    minimo = 100
    massimo = 0

    for riga in csv_input:
```

Analisi Dataset II

```
            id = riga[0]
            data = riga[1]
            valore = float(riga[2])
            stato = riga[3]
            operatore = riga[4]
            if stato == "VA":
                if valore < minimo:
                    minimo = valore
                if valore > massimo:
                    massimo = valore

        dati = [minimo, massimo]
        return dati

    def scansiona_cartelle():
        contenuto_cartella = os.listdir()
```

Analisi Dataset III

```
    for elemento in contenuto_cartella:
        if os.path.isdir(elemento):
            print("Analizziamo anno:", elemento)
            for file in os.listdir(elemento):
                valori = calcola_qualita_aria(os.path.join(elemento,
                                                           file))
                print("Il massimo e':", valori[0])
                print("Il minimo e':", valori[1])

def main():
    # Invochiamo la funzione che scansiona tutte le cartelle
    scansiona_cartelle()

main()
```

Analisi Dataset

Ideare un programma che analizzi un file di tipo `.csv` contente i record di registrazioni della qualità dell'aria relativi ad un anno, e che accorpi tutte le rilevazioni effettuate da un certo sensore, serializzandone i valori in un file separato per sensore.

Analisi Dataset I

Una possibile soluzione

```
import os
from csv import reader, writer

def serializza(percorso_file):
    sensori_dict = {}
    dataset = open(percorso_file)
    csv_input = reader(dataset)

    # Saltiamo la prima linea di header
    next(csv_input)

    # Creiamo una nuova cartella dove
    # salvare i risultati, file per file
    os.mkdir("risultati")
```

Analisi Dataset II

```
    os.chdir("risultati")

    for riga in csv_input:
        id = int(riga[0])
        data = riga[1]
        valore = float(riga[2])
        id_operatore = int(riga[3])

        file_dest = open(str(id), 'a')
        csvwriter = writer(file_dest)
        csvwriter.writerow(riga)
        file_dest.close()

def main():
    serializza("dati_stime_comunali_2019.csv")
```

Analisi Dataset III

```
main()
```

External Sort

Implementare in un programma l'algoritmo di outer sort. L'algoritmo di outer sort è utilizzato per ordinare grandi quantità di dati. In particolare, l'idea è di spezzare i dati da ordinare in chunk più piccoli e trattabili, ordinare ciascun chunk indipendentemente, e successivamente riunire tutti i risultati in un solo output. Nel nostro caso, l'obiettivo sarà quello di ordinare in ordine alfabetico un file contenente 100 parole, processando un file della grandezza di 10 parole alla volta.

External Sort I

Una possibile soluzione

```
### Funzione per effettuare l'ordinamento
def bubblesort(lista):
    n = len(lista)
    modifica = True

    while modifica:
        modifica = False

        # Attraversiamo tutta la lista
        for i in range(0, n - 1):
            if lista[i] > lista[i + 1]:
                tmp = lista[i]
                lista[i] = lista[i + 1]
                lista[i + 1] = tmp
```

External Sort II

```
                modifica = True

    return lista

def main():
    input_file = open('100_parole.txt')

    # Calcola lunghezza file scorrendo un
    # elemento alla volta
    lunghezza = 0
    for linea in input_file:
        lunghezza = lunghezza + 1
    print(lunghezza)

    # Torna all'inizio del file
    input_file.seek(0)
```

External Sort III

```
# Calcola in quante parti dividere
# il file
chunk = 10
grandezza_chunk = int(lunghezza/chunk)

# Per ogni porzione crea un file temporaneo
# contenente un chunk di parole
for i in range(0, chunk):
    file_temporaneo = open('temp' + str(i), 'w')
    for j in range(0, grandezza_chunk):
        linea = input_file.readline()
        file_temporaneo.write(linea)
    file_temporaneo.close()

# Ordina singolarmente ogni temporaneo
```

External Sort IV

```
for i in range(0, chunk):
    file_temporaneo = open('temp' + str(i), 'r')
    da_ordinare = file_temporaneo.readlines()
    file_temporaneo.close()
    ordinate = bubblesort(da_ordinare)
    file_temporaneo = open('temp' + str(i), 'w')
    for parola in ordinate:
        file_temporaneo.write(parola)
    file_temporaneo.close()

# Apri tutti i temporanei e tieni
# gli oggetti file in una lista
files_da_unire = []
for i in range(0, chunk):
    file = open('temp' + str(i), 'r')
    files_da_unire.append(file)
```

External Sort V

```
# Scrittura nel file finale
output_file = open('output.txt', 'w')
parole_successive = []
for file in files_da_unire:
    parole_successive.append(file.readline())
while parole_successive:
    candidata = min(parole_successive)
    output_file.write(candidata)
    indice = parole_successive.index(candidata)
    candidata_succ = files_da_unire[indice].readline()
    if candidata_succ:
        parole_successive[indice] = candidata_succ
    else:
        parole_successive.pop(indice)
        files_da_unire[indice].close()
```

External Sort VI

```
            files_da_unire.pop(indice)

main()
```