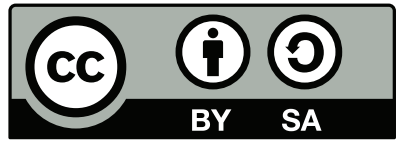
Scientific Programming

Lezione AE1 – Esercizi

Alberto Montresor

Università di Trento

2021/02/02



Esercizio – Parentsi

- Donald ha scritto varie espressioni in Python in un file, una per linea
- Sfortunatamente, l'interprete dice che in alcune espressioni c'è un errore di parentesi, senza specificare dove.
- Scrivere un programma che prenda in input un file di espressioni e per ogni espressione sbagliata, restituisca informazioni per correggerla (la posizione di una parentesi chiusa di troppo, oppure la mancanza di una o più parentesi di chiusura)
- Versione più complessa: espressioni con parentesi tonde, quadre, graffe

Esercizio – Parentesi

```
(((x+y)*7)-5                                  Mancano 1 parentesi
(3*((4+2)*8)/(4)((3*7)*14)-(3*3)              Mancano 1 parentesi
(x+4)(x+5)((x+5)*3)
(((8+2)*2)/4)((3+7)*14)-(3*3)
)(x+7)(                                       )(x+7)(
                                              ^
```

- Input: un file contenente linee di testo, contenenti espressioni
- Funzione: prende un'espressione su stringa e restituisce 0 se corretta, -x se mancano x parentesi chiuse, x se esiste una parentesi chiusa di troppo in posizione x+1
- Output: per ogni linea, stampare l'errore

Esercizio – Duckburg

Paperoga ha memorizzato i numeri di telefono dei suoi amici in un file, uno per riga.
Ora, vuole scoprire qual è il prefisso di Paperopoli - per questo motivo, cerca il più lungo prefisso comune a tutti i numeri.
Scrivere un programma che prede in input il file e restituisca il prefisso di Paperopoli.

Esercizio – Duckburg

```
12345
12395
12349
12312

123
```

```
919239144321
919239143321
919239124891
91923914431
919239144621
91923914202737

9192391
```

- Input: un file contenente diverse linee di testo.
- Funzione: prende una lista di stringhe e restituisce la più lunga stringa che è prefisso di ognuna di esse
- Output: stampa il prefisso più lugno

Esercizio – Sequenze di caratteri uguali

Sia dato un input un file contenente una tabella di $N \times M$ caratteri. Scrivere un algoritmo che restituisca la lunghezza della più lunga sequenza di caratteri uguali consecutivi, collocati lungo le righe e le colonne (opzionale, lungo e noioso: lungo le diagonali)

- Input: un file contenente diverse line di testo, di uguale lunghezza
- Funzione: una funzione che prende una sequenza e restituisca la più lunga sottostringa di caratteri uguali contenuta in essa;
- Output: stampare la lunghezza della più lunga sottostringa di caratteri uguali

Esercizio – Sequenze di caratteri uguali

uieiki
zqeitz
cheuth
keinoi

aqnrb
baqah
zanqi
oahqq
rovna
zgpao
dddee

Esercizio – Quadrato magico

Dato un file contenente n linee, ognuna delle quali contiene n interi separati da spazi, scrivere un programma che verifichi che il file contenga un quadrato magico: ovvero, contenga i numeri fra 1 e n^2 tali per cui ogni cella contiene un intero differente e la somma degli interi in ogni riga, in ogni colonna, nella diagonale principale e nella diagonale opposta sia sempre uguale.
Esempio:

```
 7    12     1    14
 2    13     8    11
16     3    10     5
 9     6    15     4
```

Spoiler alert!

Parentesi - Soluzione

```
def check(S):
    opened = 0
    for i in range(len(S)):
        if S[i] == "(":
            opened = opened+1
        elif S[i] == ")":
            opened = opened-1
            if (opened < 0):
                return i+1
    return -opened
```

Parentesi - Soluzione

```
with open("input.txt") as f:
    for line in f:
        line = line.strip()
        err = check(line)
        if (err < 0):
            print(line)
            print("Mancano", -err, "parentesi di chiusura")
        elif err>0:
            print(line)
            print(" "*(err-1) + "^")
```

Paperopoli - Soluzione

```
def prefix(numbers):
    minlen = min([len(number.strip()) for number in numbers])
    i = 0
    stop = False
    while i < minlen and not stop:
        col = [number[i] for number in numbers]
        if col.count(col[0]) != len(col):
            stop = True
        else:
            i = i+1
    return numbers[0][:i]

with open("input.txt") as f:
    numbers = [number.strip() for number in f]
    print(prefix(numbers))
```

Sequenze di caratteri uguali – Soluzione

```
# Returns the length of the longest subsequence of equal characters
def checkString(S):
    prev = ""
    count = 0
    longest = 0
    for c in S:
        if c != prev:
            count = 0
            prev = c
        count = count + 1
        longest = max(longest, count)
    return longest
```

Sequenze di caratteri uguali – Soluzione

```
with open("input.txt") as f:
    rows = [row.strip() for row in f]

nrows = len(rows)
ncols= len(rows[0])

# Compute cols
cols = [ [row[i] for row in rows]
         for i in range(ncols) ]

# Search the maximum length
maxrows = max([checkString(row) for row in rows])
maxcols = max([checkString(col) for col in cols])
print(max(maxrows, maxcols))
```

Quadrato magico – Soluzione

```
def ismagic(L):
    n = len(L)
    tot = sum(L[0])
    for i in range(n):
        val = sum(L[i])
        if val != tot:
            return False
    for j in range(n):
        val = sum([L[i][j] for i in range(n)])
        if val != tot:
            return False
    val = sum([L[i][i] for i in range(n)])
    if val != tot:
        return False
    val = sum([L[i][n-1-i] for i in range(n)])
    if val != tot:
        return False
    return True
```