LABORATORIO DI PYTHON

ITERAZIONE IN PYTHON

22 Marzo 2019

Correzione esercizi per oggi

ESERCIZIO 1 PER CASA

Scrivere una funzione che restituisce `True` se una stringa passata come parametro è palindroma (senza considerare gli spazi, la punteggiatura ma facendo distinzione tra maiuscole e miniscole: es "`Anna`" **non** è palindroma, "`ANGOLO BAR, A BOLOGNA!`" sì). Usare il ciclo `for`.

ESERCIZIO 1 PER CASA I

```
1  import string
2
3  def normalizza(stringa):
4      """Funzione che restituisce una nuova stringa, ripulita da:
       spazi e punteggiatura"""
5      stringa_pulita = ''
6      for c in stringa:
7          if c not in string.punctuation and c not in string.
       whitespace:
8              stringa_pulita = stringa_pulita+c
9      return stringa_pulita
10
11 def palindroma(s):
12     """Funzione che stabilisce se s e' palindroma"""
13     s = normalizza(s)
14     return s == s[::-1]
```

ESERCIZIO 1 PER CASA II

```
15
16  def palindroma2(s):
17      """Funzione che stabilisce se s e' palindroma (senza usare
        operatore di slicing)"""
18      s = normalizza(s)
19      if len(s)<2:
20          return True #una stringa lunga 0 o 1 e' palindroma per
        def
21      for i in range(len(s)//2): #scorro meta' stringa (vado da 0
        alla meta' arrotondata -1)
22          if s[i] != s[-1-i]: #controllo i caratteri
        corrispondenti
23              return False #mi basta un carattere non uguale per
        terminare
24      return True #se sono arrivato fin qui, e' palindroma
```

ESERCIZIO 2 PER CASA

Scrivere una funzione che presi due numeri come parametri della funzione, restituisca il Massimo Comun Divisore (MCD) tra i due numeri. Usare il ciclo `for`.

ESERCIZIO 2 PER CASA

Scrivere una funzione che presi due numeri come parametri della funzione, restituisca il Massimo Comun Divisore (MCD) tra i due numeri. Usare il ciclo `for`.

```
1  def MCD(a,b):
2      if b == 0:
3          return a
4      if a == 0:
5          return b
6      if a<b:
7          minore = a
8      else:
9          minore = b
10     #scorro tutti i numeri dal minore fino a 1
11     for i in range(minore, 0, -1):
12         if (a % i == 0) and (b % i == 0): #se entrambi sono
       divisibili per i...
13             return i #... allora i e' il loro massimo comun
       divisore
```

ESERCIZIO 3 PER CASA

Scrivere una funzione che preso come parametro un numero `n` (`n>2`) restituisce il più piccolo `c` (`c>=2`) tale che `MCD(n,c) == 1`. Usare il ciclo `for` e la funzione dell'esercizio 2.

ESERCIZIO 3 PER CASA

Scrivere una funzione che preso come parametro un numero `n` (`n>2`) restituisce il più piccolo `c` (`c>=2`) tale che `MCD(n,c) == 1`. Usare il ciclo `for` e la funzione dell'esercizio 2.

```
1  import Es2
2
3  def coprimo(n):
4      if n>2:
5          for c in range(2,n):
6              if Es2.MCD(n,c) == 1:
7                  return c
```

ESERCIZIO 4 PER CASA

Scrivere una funzione con un parametro **n**. Se **n>=7**, disegna una "stellina" a **n** punte. Si tratta di una generalizzazione della versione a 5 punte. NB! Il numero dei lati da "saltare" può essere scelto come un numero **coprimo con n** (e dunque si può usare la funzione dell'esercizio 3).

ESERCIZIO 4 PER CASA

Scrivere una funzione con un parametro **n**. Se **n>=7**, disegna una "stellina" a **n** punte. Si tratta di una generalizzazione della versione a 5 punte. NB! Il numero dei lati da "saltare" può essere scelto come un numero **coprimo con n** (e dunque si può usare la funzione dell'esercizio 3).

```
1 import turtle, Es3
2
3 def stellina(n):
4     if n>=7:
5         for i in range(n): #ho n lati
6             turtle.forward(100)
7             turtle.right((360/n)*Es3.coprimo(n))
```

Iterazione indeterminata

COSTRUTTO `while`

```
1 while condizione:
2     istruzioni interne al while
3     ...
4     istruzioni interne al while
5 istruzioni successive al while
```

- La **`condizione`** (espressione booleana) viene valutata.
- **Solo se** l'espressione booleana vale **`True`** allora si eseguono le istruzioni all'interno del costrutto di iterazione **`while`** (notare l'indentazione).
- Finite le istruzioni all'interno del while, **si torna nuovamente a testare la `condizione`.**
- Se è ancora vera, si eseguono nuovamente le istruzioni all'interno del **`while`**
- Si prosegue così finché la condizione diviene falsa.
- Se la condizione vale **`False`**, le istruzioni all'interno del **`while`** **non** sono più eseguite, e si passa alle istruzioni successive.

COSTRUTTO `while`

```
1 while condizione:
2     istruzioni interne al while
```

- Possiamo pensarlo come “`ripeti finché la condizione rimane vera`”

COSTRUTTO `while`

```
1 while condizione:
2     istruzioni interne al while
```

- Possiamo pensarlo come "`ripeti finché la condizione rimane vera`"
- Attenzione: bisogna verificare che l'espressione presente nella condizione venga modificata nelle istruzioni interne al while.

COSTRUTTO `while`

```
1 while condizione:
2     istruzioni interne al while
```

- Possiamo pensarlo come "`ripeti finché la condizione rimane vera`"
- Attenzione: bisogna verificare che l'espressione presente nella condizione venga modificata nelle istruzioni interne al while.
- Attenzione: anche se la forma è visivamente simile, il costrutto `while` è molto diverso semanticamente dal costrutto `if`.

COSTRUTTO `while`

```
1 while condizione:
2     istruzioni interne al while
```

- Possiamo pensarlo come "`ripeti finché la condizione rimane vera`"
- Attenzione: bisogna verificare che l'espressione presente nella condizione venga modificata nelle istruzioni interne al while.
- Attenzione: anche se la forma è visivamente simile, il costrutto `while` è molto diverso semanticamente dal costrutto `if`.
- Cosa accade?

```
1 a = 3
2 while a>0:
3     print("Numero positivo")
```

INDOVINA IL NUMERO PENSATO DAL CALCOLATORE

Un programma Python genera un numero random compreso tra 1 e 100 (ricorda il modulo `random` e la funzione `randint(a,b)`). Poi chiede all'utente di indovinarlo, e risponde "`troppo grande`" o "`troppo piccolo`". Prosegue così finché l'utente non indovina.

SOLUZIONE

```
1  import random
2  numero = random.randint(1,100)
3  tentativo = int(input("Indovina il numero tra 1 e 100: "))
4  while tentativo != numero:
5      if tentativo < numero:
6          print("Troppo piccolo! Prova con un numero piu' grande."
       )
7      elif tentativo > numero:
8          print("Troppo grande! Prova con un numero piu' piccolo."
       )
9      tentativo = int(input("Nuovo tentativo: "))
10 print("Bravo! Era proprio", numero)
```

ESERCIZIO (CONGETTURA DI COLLATZ)

La Congettura di Collatz si basa sul seguente algoritmo

1. Si prende un intero positivo n.
2. Se $n = 1$ l'algoritmo termina.
3. Se n è pari, lo si divide per due (divisione intera); se dispari, lo si moltiplica per 3 e si aggiunge 1.
4. Si riparte da capo con il nuovo n così trovato.

La Congettura, che non è ancora stata dimostrata, sostiene che l'algoritmo termini sempre.

Scrivere una funzione `Collatz(n)` che stampi (opzionale: sulla stessa riga) la sequenza di Collatz da n a 1. Per esempio `Collatz(19)` stampa:

```
19 58 29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
```

ESERCIZIO (CONGETTURA DI COLLATZ)

```
1  def Collatz(n):
2      if n<1:
3          print("n deve essere positivo")
4      else:
5          while n != 1:
6              print(n, end=' ') #senza andare a capo
7              if n % 2 == 0: #pari
8                  n = n//2
9              else: #dispari
10                 n = 3*n + 1
11         print(n) #stampo l'1.
```

COSA CALCOLA E RESTITUISCE QUESTA FUZNIONE?

```
1  def divisori(x):
2      div=()
3      for i in range(1,x+1):
4          if x%i==0:
5              div=div+(i,)
6      return div
```

DIVISORI CON IL WHILE

Riscrivere la funzione

```
1  def divisori(x):
2      div=()
3      for i in range(1,x+1):
4          if x%i==0:
5              div=div+(i,)
6      return div
```

usando però il ciclo while.

SOLUZIONE

```
1  def divisori(x):
2      div = ()
3      i = 1
4      while i <= x:
5          if x % i == 0:
6              div = div + (i,)
7          i += 1
8      return div
```

ESERCIZIO

Si scriva una funzione `somma(t)` che:

- Se t non è una tupla, ritorna `None`
- verifica che t che contenga solo interi (**usando il ciclo `while`**). Se non è così, ritorna `False`
- restituisce `True` sse la tupla di interi t gode della seguente proprietà: ogni elemento (tranne il primo) è maggiore della somma degli elementi che lo precedono. **Usare il ciclo `while`**.

SOLUZIONE

```
1  def somma(t):
2      if type(t) != tuple: #Verifico t tupla
3          return None
4      i = 0 #verifico t tutti interi
5      while i<len(t):
6          if type(t[i]) !=int:
7              return False
8          i += 1
9      #Se sono qui, tutti interi
10     #verifico se soddisfa proprieta'
11     if len(t) == 0: #caso banale
12         return True
13     i = 1
14     acc = t[0]
15     while i<len(t):
16         if t[i]<=acc:
17             return False
18         acc += t[i]
19         i += 1
20     return True
```

SOLUZIONE PIÙ SINTETICA

```
1  def somma2(t):
2      if type(t) != tuple:
3          return None
4      i = 0
5      while i<len(t) and type(t[i]) == int:
6          i += 1
7      #se non l'ho scorsa tutta, c'e' almeno un non intero
8      if i != len(t):
9          return False
10     if len(t) == 0:
11         return True
12     i = 1
13     acc = t[0]
14     # piu' sintetico:
15     while i<len(t) and t[i]>acc:
16         acc += t[i]
17         i += 1
18     #se scorsa tutta, gode di proprieta', altr. no
19     return i == len(t)
```

SOLUZIONE CHE USA UN SOLO WHILE

```
1  def somma(t):
2  ^^Iif type(t) == tuple: #else it returns None
3  ^^I^^Ii = 0
4  ^^I^^Isomma = 0
5  ^^I^^Iwhile i < len(t):
6  ^^I^^I^^Iif (type(t[i]) != int) or (i>0 and     t[i] <= somma):
7  ^^I^^I^^I^^Ireturn False
8  ^^I^^I^^Isomma += t[i]
9  ^^I^^I^^Ii += 1
10 ^^I^^Ireturn True
```

ESERCIZIO VELOCISSIMO

Scrivere una funzione che prende come parametro una stringa e stampa tutte le vocali contenute nella stringa, usando il ciclo while e il metodo `lower()`.

SOLUZIONE

```
1  def vocali(s):
2      i = 0
3      while i<len(s):
4          if s[i].lower() in "aeiou":
5              print(s[i])
6          i+=1
```

Esercizi per casa

Esercizi per casa (per il 27/03/2018 ore 8:59 (mail: lab06-.....)

1. Scrivere un programma che presenta un menu di scelte all'utente (inserire un nuovo voto in una tupla, visualizzare tutti i voti, stampare la media dei voti, uscire dal programma), esegue le azioni richieste, ripetutamente, e termina solo quando l'utente inserisce 0. (Esempio nella pagina successiva)
2. Scrivere una funzione che preso un numero come parametro restituisca `True` se è primo, e `False` altrimenti. Usare il ciclo `while` ma *non* il ciclo `for`.
3. Scrivere una funzione senza parametri che conta la punteggiatura in una stringa chiesta in input all'utente, stampa ``` ``Il numero di segni di punteggiatura è...'' ``` e restituisce anche tale numero, **usando il ciclo `while` ma *non* il ciclo `for`**.
4. Una sequenza di interi A è ciclica modulo k se per ogni indice i, l'elemento $A[i+1]$ è il successore modulo k di $A[i]$ (cioè `A[i+1] == (A[i]+1)%k`), e inoltre $A[0]$ è il successore modulo k

ESEMPIO OUTPUT/INTERAZIONE UTENTE ES. 1

```
***Gestione voti studente***
- Inserire un voto (1)
- Visualizzare tutti i voti (2)
- Stampare la media dei voti (3)
- Terminare il programma (0)
Scelta: 4
Scelta non valida
Scelta: 3
Nessun voto su cui calcolare la media
Scelta: 2
()
Scelta: 1
Inserisci un voto: 30
Scelta: 1
Inserisci un voto: 28
Scelta: 2
(30, 28)
```