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))
```

Itereazione con il while

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

Scrivere un programma Python che genera un numero random *compreso tra 1 e 100*.

N.B. Usare la funzione `random.randint(a, b)` del modulo `random` – Return a random integer N such that a <= N <= b. Alias for randrange(a, b+1).
`https://docs.python.org/3.7/library/random.html`

Chiedere all'utente di indovinarlo e rispondere "`troppo grande`" o "`troppo piccolo`".

Proseguire così finché l'utente non indovina.

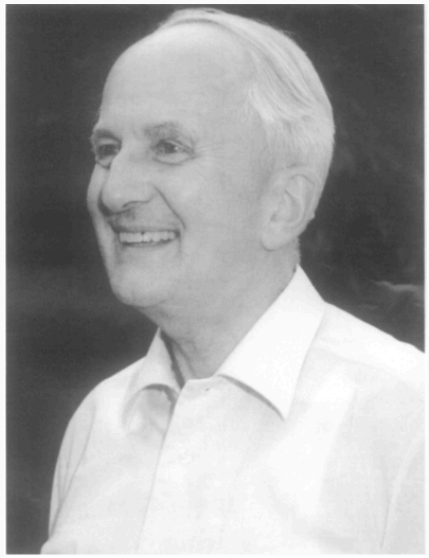
SOLUZIONE

```
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")
7      elif tentativo > numero:
8          print("troppo grande")
9      # modifico quello che confronto
10     tentativo = int(input("nuovo tentativo: "))
11 print("hai indovinato")
```

CONGETTURA DI COLLATZ

La Congettura di Collatz (nella foto, il matematico tedesco) 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.



LOTHAR COLLATZ
1910 – 1990

Info. La congettura di Collatz sostiene che l'algoritmo descritto *termini* sempre.

Esercizio (congettura di Collatz)

Scrivere una funzione `Collatz(n)` che *stampi* (possibilmente sulla stessa riga) la sequenza di Collatz da n a 1. La sequenza di Collatz viene generata a partire dall'algoritmo considerato dalla congettura.

Esempio. `Collatz(29)` stampa:

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

SOLUZIONE (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.
```

ESERCIZIO

Cosa calcola e restituisce questa funzione?

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

ESERCIZIO

Cosa calcola e restituisce questa funzione?

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

Riscrivere la funzione usando il costrutto 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, preso come parametro `t`

- se `t` non è una tupla, restituisce `None`
- verifica che `t` contenga solo interi (**usando il ciclo `while`**), altrimenti restituisce `False`.
- restituisce `True` se e solo se la tupla di interi `t` gode della seguente proprietà: ogni elemento, ad eccezione del primo, è maggiore della somma di tutti gli elementi che lo precedono.

N.B. Usare il costrutto `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 # almeno uno non e' intero
8      i += 1
9    # verifico se gli elementi di t soddisfano la proprieta'
10   if len(t) == 0: # caso banale
11     return True
12   i = 1 #parto dall'elemento con indice 1
13   somma_parziale = t[0]
14   while i < len(t):
15     if t[i] <= somma_parziale:
16       return False # almeno uno non e' maggiore
17     somma_parziale += t[i]
18     i += 1
19   return True
```

ESERCIZIO VELOCISSIMO

Scrivere una funzione che prende come parametro una stringa e stampa (anche una per riga) tutte le vocali contenute nella stringa.

Usare il ciclo while ed il metodo `lower()`

N.B. `lower()`, chiamato su una stringa (`"Ciao".lower()`) restituisce una stringa con gli stessi caratteri in minuscolo, mantenendone l'ordine (`"ciao"`).

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
```

ALTRI ESERCIZI, EVENTUALMENTE DA FINIRE A CASA.

Scrivere un programma di gestione dei voti di uno studente, che presenta il seguente elenco di scelte all'utente che sceglie inserendo il codice associato ad ogni opzione:

- inserire un nuovo voto in una tupla [1];
- stampare tutti i voti [2];
- stampare la media dei voti [3];
- terminare il programma [0].

esempio di interazione con l'utente nella pagina successiva.

ESEMPIO OUTPUT/INTERAZIONE UTENTE

```
***Gestione voti studente***
- Inserire un voto [1]
- Stampare 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)
>>> Scelta: 3
>>> 29.0
>>> Scelta: 0
```

SOLUZIONE I

```
print("***Gestione voti studente***")
print("- Inserire un voto [1]")
print("- Stampare tutti i voti [2]")
print("- Stampare la media dei voti [3]")
print("- Terminare il programma [0]")
voti = ()
scelta = int(input("Scelta: "))
while scelta != 0:
    if scelta == 1:
        voto = int(input("Inserisci un voto: "))
        voti += (voto,) # concateno una nuova tupla
    elif scelta == 2:
        print(voti)
    elif scelta == 3:
        if len(voti) > 0:
```

SOLUZIONE II

```
16              import statistics # uso un modulo esterno per la
    media
17              media = statistics.mean(voti)
18              print("La media dei voti e' %.2f" % media) # nuovo
    modo per stampare numeri
19          else:
20              print("Nessun voto su cui calcolare la media")
21      else:
22          print("Scelta non valida")
23      print()
24      scelta = int(input("Scelta: "))
```

ALTRI ESERCIZI, EVENTUALMENTE DA FINIRE A CASA.

Scrivere una funzione che preso un numero come parametro restituisca `True` se è primo, e `False` altrimenti.

Testare con 32416190071.

N.B. Usare `while` e *NON* `for`.

SOLUZIONE

Esistono molti algoritmi diversi:

https://en.wikipedia.org/wiki/Primality_test

```
1  def is_prime(n):
2      if n <= 3:
3          return n > 1
4      elif n % 2 == 0 or n % 3 == 0:
5          return False
6      else:
7          i = 5
8          while i * i <= n:
9              if n % i == 0 or n % (i + 2) == 0:
10                 return False
11             i = i + 6
12         return True
```

ALTRI ESERCIZI, EVENTUALMENTE DA FINIRE A CASA.

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, anche, restituisce tale numero.

Il modulo **`string`** ha numerose funzioni e costanti che possono tornare utili, in particolare la costante **`string.punctuation`** è la stringa contenente segni di punteggiatura.

N.B. Usare **`while`** e *NON* **`for`**.

SOLUZIONE

```
1  import string
2  def contaPunteggiatura():
3      stringa = input("Insersci una stringa: ")
4      n = 0 # accumulatore per conteggio
5      i = 0 # indice per caratteri nella stringa
6      while i < len(stringa):
7          if stringa[i] in string.punctuation:
8              n +=1
9          i +=1
10     print("Il numero di segni di punteggiatura e' %d" % n)
11     return n
```

ALTRI ESERCIZI, EVENTUALMENTE DA FINIRE A CASA.

Una sequenza di interi A è ciclica modulo k se

1. 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`;
2. $A[0]$ è il successore modulo k dell'ultimo elemento di A.

ALTRI ESERCIZI, EVENTUALMENTE DA FINIRE A CASA.

Si scriva una funzione `ciclo(A,k)` che data una tupla di interi A e un intero k, verifica se A è ciclica modulo k.

Testare con `A = (1,0,1,0)` e `k = 2`.

N.B. Usare `while` e *NON* `for`.

SOLUZIONE

```
1  def ciclo_modulo(A,k):
2      # sequenza vuota, banalmente vero
3      if len(A) == 0:
4          return True
5      # scorro tutti gli elementi, escluso l'ultimo
6      i = 0
7      while i < len(A)-1:
8          condizione = A[i+1] != (A[i]+1)%k
9          if condizione:
10             return False
11         i += 1
12     # controllo se il primo e' il successore dell'ultimo
13     check_succ = A[0] == (A[len(A)-1]+1)%k
14     return check_succ
```