Programmazione in Python

Esercizi

Dario Pescini - Mirko Cesarini [1]

Università degli Studi di Milano-Bicocca

Dipartimento di Statistica e Metodi Quantitativi

dario.pescini@unimib.it

Indovina la parola

Si chiede di implementare un programma che, scelta una parola a caso tra quelle presenti nel file `listaParoleIta.txt`, chieda all'utente di indovinarla.
Il programma dovrà, inoltre, fornire come aiuto il conteggio, per ciascuno dei caratteri presenti nella parola proposta dall'utente, delle sue occorrenze nella parola da indovinare.

```
ossequiosi

la parola è composta da  12 caratteri
che parola è? (-1 per terminare) computer
aiuto:
c e' presente  0 volte
e e' presente  1 volte
m e' presente  0 volte
o e' presente  2 volte
p e' presente  0 volte
r e' presente  0 volte
u e' presente  1 volte
t e' presente  0 volte
```

file listaParoleIta.txt

```
a
abate
abati
abbagli
abbaglia
abbagliai
abbagliamo
abbagliano
abbagliare
abbagliata
abbagliate
abbagliati

....

zuffoli
zuffoliamo
zuffolino
zuffolo
zuppa
zuppe
zuppiera
zuzzurellone
```

Soluzione

```
import random
fileName = 'listaParoleIta.txt'

# caricamento lista parole possibili target
fileParole = open(fileName, 'r')
listaParole = fileParole.readlines()
fileParole.close()
lunghezzaLista = len(listaParole)

# sorteggio della parola segreta
iParola = random.randint(0, lunghezzaLista - 1)
parolaDaIndovinare = listaParole[iParola]
# print parolaDaIndovinare
print 'la parola è composta da ', len(parolaDaIndovinare),
    'caratteri'
```

```
# inserimento tentativi e controllo soluzione
termina = False
indovinato = False
while termina == False and indovinato == False:
    parolaTentativo = raw_input('che parola è? (-1 per
    terminare) ')
    if parolaTentativo == '-1':
        termina = True
    else:
        if parolaTentativo == parolaDaIndovinare:
            print 'hai indovinato'
            indovinato = True
        else:
            conteggio = {}
            for carattere in parolaTentativo:
                if carattere not in conteggio.keys():
                    conteggio[carattere] =
    parolaDaIndovinare.count(carattere)
            print "aiuto:"
            for el in conteggio.keys():
                print el, "e' presente ", conteggio[el],
    'volte'
```

Codice Fiscale

Si chiede di implementare in python l'algoritmo usato per la generazione del codice fiscale.
I primi quindici caratteri sono indicativi dei dati anagrafici di ciascun soggetto secondo l'ordine seguente:

- tre caratteri alfabetici per il cognome;
- tre caratteri alfabetici per il nome;
- due caratteri numerici per l'anno di nascita;
- un carattere alfabetico per il mese di nascita;
- due caratteri numerici per il giorno di nascita ed il sesso;
- quattro caratteri, di cui uno alfabetico e tre numerici per il comune italiano o per lo Stato estero di nascita.

Il sedicesimo carattere, alfabetico, ha funzione di controllo.

Generatore codice fiscale: caratteri indicativi del cognome

- I cognomi che risultano composti da più parti o comunque separati od interrotti, vengono considerati come se fossero scritti secondo un'unica ed ininterrotta successione di caratteri.
- Per i soggetti coniugati di sesso femminile si prende in considerazione soltanto il cognome da nubile.
- Se il cognome contiene tre o più consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la seconda e la terza consonante.
- Se il cognome contiene due consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima e la seconda consonante e la prima vocale.
- Se il cognome contiene una consonante e due vocali, si rilevano, nell'ordine, quella consonante e quindi la prima e la seconda vocale.
- Se il cognome contiene una consonante e una vocale, si rilevano la consonante e la vocale, nell'ordine, e si assume come terzo carattere la lettera x (ics).
- Se il cognome e' costituito da due sole vocali, esse si rilevano, nell'ordine, e si assume come terzo carattere la lettera x (ics).

Generatore codice fiscale: caratteri indicativi del nome

- I nomi doppi, multipli o comunque composti, vengono considerati come scritti per esteso in ogni loro parte e secondo un'unica ed ininterrotta successione di caratteri.
- Se il nome contiene quattro o più consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la terza e la quarta consonante.
- Se il nome contiene tre consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la seconda e la terza consonante.
- Se il nome contiene due consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima e la seconda consonante e la prima vocale.
- Se il nome contiene una consonante e due vocali, i tre caratteri da rilevare sono, nell'ordine, quella consonante e quindi la prima e la seconda vocale.
- Se il nome contiene una consonante e una vocale, si rilevano la consonante e la vocale, nell'ordine, e si assume come terzo carattere la lettera x (ics).
- Se il nome e' costituito da due sole vocali, esse si rilevano nell'ordine, e si assume come terzo carattere la lettera x (ics).

Generatore codice fiscale: Data, sesso e luogo di nascita

- I due caratteri numerici indicativi dell'anno di nascita sono, nell'ordine, la cifra delle decine e la cifra delle unità dell'anno stesso.
- Il carattere alfabetico corrispondente al mese di nascita e' quello stabilito per ciascun mese nella seguente tabella:

Gennaio = A	Maggio = E	Settembre = P
Febbraio = B	Giugno = H	Ottobre = R
Marzo = C	Luglio = L	Novembre = S
Aprile = D	Agosto = M	Dicembre = T

Generatore codice fiscale: Data, sesso e luogo di nascita

- I due caratteri numerici indicativi del giorno di nascita e del sesso vengono determinati nel modo seguente:
  - Per i soggetti maschili il giorno di nascita figura invariato, con i numeri da uno a trentuno, facendo precedere dalla cifra zero i giorni del mese dall'uno al nove.
  - Per i soggetti femminili il giorno di nascita viene aumentato di quaranta unità, per cui esso figura con i numeri da quarantuno a settantuno.

- I quatto caratteri alfanumerici indicativi del comune italiano o dello Stato estero di nascita, sono costituiti da un carattere alfabetico seguito da tre caratteri numerici, secondo la codifica stabilita dall'Agenzia del Territorio e contenuta nel file `CodiceComuniItalia.csv`

Generatore codice fiscale: Carattere alfabetico di controllo

Il sedicesimo carattere ha funzione di controllo dell'esatta trascrizione dei primi quindici caratteri e viene determinato in questo modo:

- ciascuno degli anzidetti quindici caratteri, a seconda che occupi posizione di ordine pari o posizione di ordine dispari, viene convertito in un valore numerico, in base alle tabelle di corrispondenza successivamente riportate...

  Per i sette caratteri con posizione di ordine

  - pari: file `CFTabellaPari.tsv`
  - dispari: file `CFTabellaDispari.tsv`

Generatore codice fiscale: Carattere alfabetico di controllo

- I valori numerici così determinati vengono addizionati e la somma si divide per il numero 26.
- Il carattere di controllo si ottiene convertendo il resto di tale divisione nel carattere alfabetico ad esso corrispondente nel file `CFTabellaResto.tsv`

Cavia

```
nome = 'alessandro'
cognome = 'manzoni'
dataNascita = '07031785'
genere = 'M'
comune = 'milano'

MNZLSN85C07F205D
```

Cognome

- I cognomi che risultano composti da più parti o comunque separati od interrotti, vengono considerati come se fossero scritti secondo un'unica ed ininterrotta successione di caratteri.
- Per i soggetti coniugati di sesso femminile si prende in considerazione soltanto il cognome da nubile.
- Se il cognome contiene tre o più consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la seconda e la terza consonante.
- Se il cognome contiene due consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima e la seconda consonante e la prima vocale.
- Se il cognome contiene una consonante e due vocali, si rilevano, nell'ordine, quella consonante e quindi la prima e la seconda vocale.
- Se il cognome contiene una consonante e una vocale, si rilevano la consonante e la vocale, nell'ordine, e si assume come terzo carattere la lettera x (ics).
- Se il cognome e' costituito da due sole vocali, esse si rilevano, nell'ordine, e si assume come terzo carattere la lettera x (ics).

Cognome

- I cognomi che risultano composti da più parti o comunque separati od interrotti, vengono considerati come se fossero scritti secondo un'unica ed ininterrotta successione di caratteri.
- Per i soggetti coniugati di sesso femminile si prende in considerazione soltanto il cognome da nubile.
- Se il cognome contiene tre o più consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la seconda e la terza consonante.
- Se il cognome contiene due consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima e la seconda consonante e la prima vocale.
- Se il cognome contiene una consonante e due vocali, si rilevano, nell'ordine, quella consonante e quindi la prima e la seconda vocale.
- Se il cognome contiene una consonante e una vocale, si rilevano la consonante e la vocale, nell'ordine, e si assume come terzo carattere la lettera x (ics).
- Se il cognome e' costituito da due sole vocali, esse si rilevano, nell'ordine, e si assume come terzo carattere la lettera x (ics).

```
consonanti + vocali + x
```

Cognome

- I cognomi che risultano composti da più parti o comunque separati od interrotti, vengono considerati come se fossero scritti secondo un'unica ed ininterrotta successione di caratteri.
- Per i soggetti coniugati di sesso femminile si prende in considerazione soltanto il cognome da nubile.
- Se il cognome contiene tre o più consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la seconda e la terza consonante.
- Se il cognome contiene due consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima e la seconda consonante e la prima vocale.
- Se il cognome contiene una consonante e due vocali, si rilevano, nell'ordine, quella consonante e quindi la prima e la seconda vocale.
- Se il cognome contiene una consonante e una vocale, si rilevano la consonante e la vocale, nell'ordine, e si assume come terzo carattere la lettera x (ics).
- Se il cognome e' costituito da due sole vocali, esse si rilevano, nell'ordine, e si assume come terzo carattere la lettera x (ics).

```
consonanti + vocali + x
stringa[:3]
```

Cognome

```
def estraiCarCognome(parola):
    parola = parola.replace(' ', '')
    parola = parola.upper()
    frammentoConsonanti = ''
    frammentoVocali = ''
    for carattere in parola:
        if carattere not in ['A', 'E', 'I', 'O', 'U']:
            frammentoConsonanti += carattere
        else:
            frammentoVocali += carattere
    parolaRiordinata = frammentoConsonanti + frammentoVocali +
    'XXX'
    # print parolaRiordinata
    return parolaRiordinata[:3]
```

Nome

- I nomi doppi, multipli o comunque composti, vengono considerati come scritti per esteso in ogni loro parte e secondo un'unica ed ininterrotta successione di caratteri.
- Se il nome contiene quattro o più consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la terza e la quarta consonante.
- Se il nome contiene tre consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la seconda e la terza consonante.
- Se il nome contiene due consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima e la seconda consonante e la prima vocale.
- Se il nome contiene una consonante e due vocali, i tre caratteri da rilevare sono, nell'ordine, quella consonante e quindi la prima e la seconda vocale.
- Se il nome contiene una consonante e una vocale, si rilevano la consonante e la vocale, nell'ordine, e si assume come terzo carattere la lettera x (ics).
- Se il nome e' costituito da due sole vocali, esse si rilevano nell'ordine, e si assume come terzo carattere la lettera x (ics).

Nome

- I nomi doppi, multipli o comunque composti, vengono considerati come scritti per esteso in ogni loro parte e secondo un'unica ed ininterrotta successione di caratteri.
- Se il nome contiene quattro o più consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la terza e la quarta consonante.
- Se il nome contiene tre consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la seconda e la terza consonante.
- Se il nome contiene due consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima e la seconda consonante e la prima vocale.
- Se il nome contiene una consonante e due vocali, i tre caratteri da rilevare sono, nell'ordine, quella consonante e quindi la prima e la seconda vocale.
- Se il nome contiene una consonante e una vocale, si rilevano la consonante e la vocale, nell'ordine, e si assume come terzo carattere la lettera x (ics).
- Se il nome e' costituito da due sole vocali, esse si rilevano nell'ordine, e si assume come terzo carattere la lettera x (ics).

```
consonanti + vocali + x
```

Nome

- I nomi doppi, multipli o comunque composti, vengono considerati come scritti per esteso in ogni loro parte e secondo un'unica ed ininterrotta successione di caratteri.
- Se il nome contiene quattro o più consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la terza e la quarta consonante.
- Se il nome contiene tre consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima, la seconda e la terza consonante.
- Se il nome contiene due consonanti, i tre caratteri da rilevare sono, nell'ordine, la prima e la seconda consonante e la prima vocale.
- Se il nome contiene una consonante e due vocali, i tre caratteri da rilevare sono, nell'ordine, quella consonante e quindi la prima e la seconda vocale.
- Se il nome contiene una consonante e una vocale, si rilevano la consonante e la vocale, nell'ordine, e si assume come terzo carattere la lettera x (ics).
- Se il nome e' costituito da due sole vocali, esse si rilevano nell'ordine, e si assume come terzo carattere la lettera x (ics).

```
consonanti + vocali + x
stringa[1,3,4] oppure stringa[:3]
```

Nome

```
def estraiCarNome(parola):
    parola = parola.replace(' ', '')
    parola = parola.upper()
    frammentoConsonanti = ''
    frammentoVocali = ''
    for carattere in parola:
        if carattere not in ['A', 'E', 'I', 'O', 'U']:
            frammentoConsonanti += carattere
        else:
            frammentoVocali += carattere
    if len(frammentoConsonanti) > 3:
        frammentoConsonanti = frammentoConsonanti[
            0] + frammentoConsonanti[2] +
    frammentoConsonanti[3]
    parolaRiordinata = frammentoConsonanti + frammentoVocali +
    'XXX'
    # print parolaRiordinata
    return parolaRiordinata[:3]
```

Data

- I due caratteri numerici indicativi dell'anno di nascita sono, nell'ordine, la cifra delle decine e la cifra delle unità dell'anno stesso.
- Il carattere alfabetico corrispondente al mese di nascita e' quello stabilito per ciascun mese nella seguente tabella:

Gennaio = A	Maggio = E	Settembre = P
Febbraio = B	Giugno = H	Ottobre = R
Marzo = C	Luglio = L	Novembre = S
Aprile = D	Agosto = M	Dicembre = T

Data

```
def estraiCarMese(parola):
    dizionarioMesi = {'01': 'A', '02': 'B', '03': 'C', '04':
    'D',
                      '05': 'E', '06': 'H', '07': 'L', '08':
    'M',
                      '09': 'P', '10': 'R', '11': 'S', '12':
    'T'}
    return dizionarioMesi[parola]
```

Sesso

- I due caratteri numerici indicativi del giorno di nascita e del sesso vengono determinati nel modo seguente:
  - Per i soggetti maschili il giorno di nascita figura invariato, con i numeri da uno a trentuno, facendo precedere dalla cifra zero i giorni del mese dall'uno al nove.
  - Per i soggetti femminili il giorno di nascita viene aumentato di quaranta unità, per cui esso figura con i numeri da quarantuno a settantuno.

```
def estraiCarGiorno(parola, carGenere):
    if carGenere == 'F':
        giorno = int(parola) + 40
        print giorno
        parola = str(giorno)
    return parola
```

Luogo

- I quattro caratteri alfanumerici indicativi del comune italiano o dello Stato estero di nascita, sono costituiti da un carattere alfabetico seguito da tre caratteri numerici, secondo la codifica stabilita dall'Agenzia del Territorio e contenuta nel file `CodiceComuniItalia.csv`

```
CODICI COMUNI D'ITALIA - COMUNI ATTUALI alla data d
Codice;Prov;Denominaz Italiana;Denominaz Estera;Cod
A154;VI;ALBETTONE;;D7AC;VI;VI;VI00;Vicenza;24002;;
A155;CZ;ALBI;;T2AC;CZ;CZ;CZ00;Catanzaro;79002;;
A157;TO;ALBIANO D'IVREA;;A1AF;TO;TO;TO40;Ivrea;1004
A158;TN;ALBIANO;;E1AD;;;;;22002;;
A159;MB;ALBIATE;;C1AI;MI;MI;MI20;Milano 2;108003;;
A160;CS;ALBIDONA;;T3AH;CS;CS;CS00;Cosenza;78006;;
A161;PD;ALBIGNASEGO;;D3AD;PD;PD;PD00;Padova;28003;;
```

`primo campo`

Luogo

```
def estraiCarComune(parola):
    dizComuni = {}
    nomeFile = 'CFCodiceComuniItalia.csv'
    fileComuni = open(nomeFile, 'r')
    listaComuni = fileComuni.readlines()
    fileComuni.close()
    for linea in listaComuni:
        frammenti = linea.split(';')
        # print frammenti
        dizComuni[frammenti[2]] = frammenti[0]
    return dizComuni[parola.upper()]
```

Carattere alfabetico di controllo

Il sedicesimo carattere ha funzione di controllo dell'esatta trascrizione dei primi quindici caratteri e viene determinato in questo modo:

- ciascuno degli anzidetti quindici caratteri, a seconda che occupi posizione di ordine pari o posizione di ordine dispari, viene convertito in un valore numerico, in base alle tabelle di corrispondenza successivamente riportate...

  Per i sette caratteri con posizione di ordine

  - pari: file `CFTabellaPari.tsv`
  - dispari: file `CFTabellaDispari.tsv`

CFTabellaPari.tsv

#Carattere	Valore	Carattere	Valore	Carattere	Valore	Carattere	Valore
0	0	9	9	I	8	R	17
1	1	A	0	J	9	S	18
2	2	B	1	K	10	T	19
3	3	C	2	L	11	U	20
4	4	D	3	M	12	V	21
5	5	E	4	N	13	W	22
6	6	F	5	O	14	X	23
7	7	G	6	P	15	Y	24
8	8	H	7	Q	16	Z	25

CFTabellaDispari.tsv

#Carattere	Valore	Carattere	Valore	Carattere	Valore	Carattere	Valore
0	1	9	21	I	19	R	8
1	0	A	1	J	21	S	12
2	5	B	0	K	2	T	14
3	7	C	5	L	4	U	16
4	9	D	7	M	18	V	10
5	13	E	9	N	20	W	22
6	15	F	13	O	11	X	25
7	17	G	15	P	3	Y	24
8	19	H	17	Q	6	Z	23

Controllo

```
def estraiCarControllo(parola):
    file = open('CFTabellaDispari.tsv', 'r')
    listaDispari = file.readlines()
    file.close()
    file = open('CFTabellaPari.tsv', 'r')
    listaPari = file.readlines()
    file.close()
    del listaDispari[0]
    del listaPari[0]
    dizDispari = {}
    for linea in listaDispari:
        frammenti = linea.strip().split('\t')
        for i in range(4):
            dizDispari[frammenti[2 * i]] = frammenti[2 * i + 1]
    # print dizDispari
    dizPari = {}
    for linea in listaPari:
        frammenti = linea.strip().split('\t')
        for i in range(4):
            dizPari[frammenti[2 * i]] = frammenti[2 * i + 1]
```

Carattere alfabetico di controllo

- I valori numerici così determinati vengono addizionati e la somma si divide per il numero 26.
- Il carattere di controllo si ottiene convertendo il resto di tale divisione nel carattere alfabetico ad esso corrispondente nel file `CFTabellaResto.tsv`

CFTabellaResto.tsv

```
0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H
8	I
9	J
10	K
11	L
12	M
13	N
14	O
15	P
16	Q
17	R
18	S
19	T
20	U
21	V
22	W
23	X
24	Y
25	Z
```

Controllo

```
    file = open('CFTabellaResto.tsv', 'r')
     listaResto = file.readlines()
     file.close()
     dizResto = {}
     for linea in listaResto:
         frammenti = linea.strip().split('\t')
         dizResto[frammenti[0]] = frammenti[1]
     # print dizResto
     i = 0
     somma = 0
     for el in parola:
         if i % 2 == 1:  # attenzione i è indice: 0 = primo
             somma += int(dizPari[el])
             print dizPari[el]
         else:
             somma += int(dizDispari[el])
             print dizDispari[el]
         i += 1
     print 'somma = ', somma, 'resto = ', somma % 26
     car = dizResto[str(somma % 26)]
     return car
```

```
nome = raw_input('inserire il nome ')
cognome = raw_input('inserire il cognome ')
dataNascita = raw_input('inserire la data di nascita nel
    formato ggmmaaaa ')
genere = raw_input('inserire il genere M o F ')
comune = raw_input('inserire il comune di nascita ')

nome = nome.replace(' ', '')
print nome
cognome = cognome.replace(' ', '')
print cognome
giorno = dataNascita[:2]
print giorno
mese = dataNascita[2:4]
print mese
anno = dataNascita[6:8]
print anno, "\n"
carCognome = estraiCarCognome(cognome)
carNome = estraiCarNome(nome)
carMese = estraiCarMese(mese)
carGiorno = estraiCarGiorno(giorno, genere)
carComune = estraiCarComune(comune)
CF = carCognome + carNome + anno + carMese + carGiorno +
    carComune
carControllo = estraiCarControllo(CF)
CF += carControllo
print CF
```

Esercizio - Tema d'esame 8/mag/2017

Il file telefonate.csv contiene informazioni sulle telefonate effettuate in un *unico* mese da alcuni clienti di un operatore di telefonia cellulare. La prima riga contiene l'intestazione delle colonne.

```
Cod_Chiamante;Cod_Destinatario;Cod_Cella_Chiamante;Cod_Cella_Destinata
```

- `Cod_Chiamante` un intero che identifica univocamente il cliente che ha effettuato la chiamata
- `Cod_Destinatario` un intero che identifica univocamente il cliente che ha ricevuto la chiamata
- `Cod_Cella_Chiamante` un intero che identifica l'area nella quale si trova il cliente che ha effettuato la chiamata
- `Cod_Cella_Destinatario` un intero che identifica l'area nella quale si trova il cliente che ha ricevuto la chiamata
- `GGI:HHI:MMI` rappresentano rispettivamente il giorno, l'ora e il minuto in cui e' iniziata la chiamata
- `GGF:HHF:MMF` rappresentano rispettivamente il giorno, l'ora e il minuto in cui e' finita la chiamata

Le righe che iniziano con # sono da scartare

Il \r\n rappresentano due caratteri di "a capo".

Specifica dei requisiti - ottieniDatiTelefonate()

Implementate la funzione `ottieniDatiTelefonate(nomeFile)` che accetta in ingresso il nome del file da elaborare e restituisce una lista di tuple, dove ogni tupla corrisponde ad una telefonata e contiene (`Cod_Chiamante`, `Cod_Destinatario`, `Cod_Cella_Chiamante`, `Cod_Cella_Destinatario`, `Numeo_Minuti_Di_Conversazione`)

I minuti di conversazione si calcolano includendo anche il minuto di inizio della conversazione
Devono essere escluse le righe che rispettano i seguenti criteri:

- `Cod_Chiamante e Cod_Destinatario` devono essere diversi
- `GG, HH e MM` devo appartenere ai rispettivi domini es $HH \in 0 \dots 23$
- L'inizio della chiamata non puo' essere successivo alla fine.
- Telefonate piu' lunghe di 10 ore sono errori e non entrano nel risultato restituito

Per il calcolo dei minuti di conversazione, si suggerisce di trasformare (data,ora,minuto) in un intero corrispondente al numero di minuti trascorsi dall'inizio del mese. I minuti di conversazione si calcolano includendo anche il minuto di inizio della conversazione ($+= 1$ sui min. di conversazione)

Implementazione - funzioni accessorie

```
def check(gg, hh, mm):
    if gg < 1 or gg > 31:
        return False
    if hh < 0 or hh > 23:
        return False
    if mm < 0 or mm > 59:
        return False
    return True  # controlli ok
```

```
def tempoTrascorso(gg, hh, mm):
    return mm + hh * 60 + gg * 24 * 60
```

Implementazione - ottieniDatiTelefonate()

```
def ottieniDatiTelefonate(nomeFile):
    # (Cod_Chiamante, Cod_Destinatario, Cod_Cella_Chiamante,
    # Cod_Cella_Destinatario, Numeo_Minuti_Di_Conversazione)
    data = []
    fi = open(nomeFile, 'r')
    fi.readline()  # salto la 1. riga
    for line in fi:
        if line[0]!='#':
            line = line.strip('\r\n')
            line = line.split(';')
            chiamante = int(line[0])
            destinatario = int(line[1])
            cellaChiamante = int(line[2])
            cellaDestinatario = int(line[3])
            timeInizio=line[4]
            timeFine=line[5]

            timeInizioLi=timeInizio.split(':')
            timeFineLi=timeFine.split(':')
            # ... continua
```

```
            # ...
            ggi = int(timeInizioLi[0])
            hhi = int(timeInizioLi[1])
            mmi = int(timeInizioLi[2])
            ggf = int(timeFineLi[0])
            hhf = int(timeFineLi[1])
            mmf = int(timeFineLi[2])

            inizio = tempoTrascorso(ggi, hhi, mmi)
            fine = tempoTrascorso(ggf, hhf, mmf)
            delta = fine - inizio
            if chiamante != destinatario and
                    check(ggi, hhi, mmi) and
                    check(ggf, hhf, mmf) and delta >= 0:
                durata = delta + 1  # conteggio 1. minuto
                el = ( chiamante, destinatario,
                       cellaChiamante, cellaDestinatario,
                       durata )
                data.append(el)
    fi.close()
    return data
```

Specifica dei requisiti - calcolaBollette()

Implementa la funzione `calcolaBollette(datiChiamate)`.
Parametro in ingresso la struttura dati restituita dalla funzione precedente.
Deve restituire un dizionario di coppie chiave valore `codice_cliente:importo` dove il valore e' un float con l'importo che il cliente deve pagare per le chiamate effettuate nel mese.
Un cliente paga 0.10 euro per ogni minuto di chiamata effettuata. Il cliente che riceve la chiamata non paga niente.
Ogni cliente ha un bonus di 20 minuti gratuiti al mese
Per esempio se il cliente 1 effettua chiamate per 55 minuti e il cliente 2 effettua chiamate per 15 minuti, il dizionario restituito dovra' contenere: `{1:3.5, 2:0}`.
L'insieme dei clienti contenuti nel dizionario e' formato da quei clienti che appaiono almeno una volta come chiamanti nella struttura dati fornita in ingresso.

Implementazione - calcolaBollette()

```
def calcolaBollette(datiChiamate):
    # [chiamante,destinatario, cellaChiamante,
    #  cellaDestinatario, durata]
    clienti2minuti = {}
    for el in datiChiamate:
        chiamante = el[0]
        durata = el[4]
        if chiamante in clienti2minuti:
            clienti2minuti[chiamante] += durata
        else:
            clienti2minuti[chiamante] = durata
    clienti2tariffa = {}
    for cliente in clienti2minuti:
        minuti = clienti2minuti[cliente]
        if minuti <= 20:
            clienti2tariffa[cliente] = 0
        else:
            clienti2tariffa[cliente] = (minuti - 20) * 0.10
    return clienti2tariffa
```

Specifica dei requisiti - calcolaCelleCongestionate()

Implementate la funzione `calcolaCelleCongestionate(datiChiamate)`.
La funzione accetta come parametro in ingresso la struttura dati creata dalla funzione ottieniDatiTelefonate.
La funzione deve identificare le celle (cioe' le aree) che hanno un elevato carico di telefonate.
Per far cio', dovete calcolare il numero totale di telefonate in partenza effettuate per cella.
Sono da considerarsi congestionate le celle nelle quali il numero di telefonate effettuate e' maggiore della media (calcolata sui totali precedentemente calcolati).
La funzione dovra' restituire una lista con i codici delle celle congestionate.

Implementazione - calcolaCelleCongestionate()

```
def calcolaCelleCongestionate(datiChiamate):
    # (chiamante,destinatario, cellaChiamante,
    cellaDestinatario, durata)
    celle2chiamate = {}
    for el in datiChiamate:
        cella = el[2]
        if cella in celle2chiamate:
            celle2chiamate[cella] += 1
        else:
            celle2chiamate[cella] = 1

    numChiamate = 0
    numCelle = 0
    # ... continua
```

```
    # ...
    for cella in celle2chiamate:
        n = celle2chiamate[cella]
        numChiamate += n
        numCelle += 1
    media = float(numChiamate) / numCelle
    congestionate = []
    for cella in celle2chiamate:
        n = celle2chiamate[cella]
        if n > media:
            congestionate.append(cella)
    return congestionate
```

Dall'RNA alle proteine

I moderni strumenti di laboratorio permettono di estrarre le sequenze di nucleotidi (RNA) espresse in un certo individuo. Queste informazioni vengono salvate in file di formato standard FASTA. Sapendo che i nucleotidi vengono letti in triplette (codoni) e che ciascuna di queste triplette corrisponde ad un ammino-acido nella sequenza proteica, si chiede di scrivere un programma che abbia come input il file fasta `A06662-RNA.fasta` e la tabella di conversione `codonTable` e restituisca le possibili sequenze proteiche. Inoltre, per chiarezza, si sostituiscano gli ammino-acidi `'STOP'` con il carattere `'*'`

A06662-RNA.fasta

```
>A06662.1 Synthetic nucleotide sequence of the human GSH
    transferase pi gene
UGGGACCAGUCAGCAGAGGCAGCGUGUGUGCGCGUGCGUGUGCGUGUGUGUGCGUGUGUGUGUG
UACGCUUGCAUUUGUGUCGGGUGGGUAAGGAGAUAGAGAUGGGCGGGCAGUAGGCCCAGGUCCC
GAAGGCCUUGAACCCACUGGUUUGGAGUCUCCUAAGGGCAAUGGGGGCCAUUGAGAAGUCUGAA
CAGGGCUGUGUCUGAAUGUGAGGUCUAGAAGGAUCCUCCAGAGAAGCCAGCUCUAAAGCUUUUG
CAAUCAUCUGGUGAGAGAACCCAGCAAGGAUGGACAGGCAGAAUGGAAUAGAGAUGAGUUGGCA
GCUGAAGUGGACAGGAUUUGGUACUAGCCUGGUUGUGGGGAGCAAGCAGAGGAGAAUCUGGGAC
UCUGGUGGUCUGGCCUGGGGCAGACGGGGGUGUCUCAGGGGCUGGGAGGGAUGAGAGUAGGAUG
AUACAUGGUGGUGUCUGGCAGGAGGCGGGCAAGGAUGACUAUGUGAAGGCACUGCCCGGGCAAC
UGAAGCCUUUUGAGACCCUGCUGUCCCAGAACCAGGGAGGCAAGACCUUCAUUGUGGGAGACCA
GGUGAGCAUCUGGCC
```

codonTable

```
GCU    A
GCC    A
GCA    A
GCG    A
CGU    R
CGC    R
CGA    R

...

UUC    F
UAU    Y
UAC    Y
AUG    M
UGG    W
UAG    STOP
UGA    STOP
UAA    STOP
```

fastaRNA2Prot.py

```
with open('codonTable', 'r') as codons:
    rna2cod = codons.readlines()
    codons.close()
# print rna2cod

dizRNA2Prot = {}
for line in rna2cod:
    tokens = line.strip().split('\t')
    dizRNA2Prot[tokens[0]] = tokens[1]
# print dizRNA2Prot

with open('A06662-RNA.fasta', 'r') as rna:
    rnaFile = rna.readlines()
    rna.close()
del rnaFile[0]
# print rnaFile

rnaString = ''
for line in rnaFile:
    rnaString += line.strip()
print rnaString
```

fastaRNA2Prot.py

```
prots = []
for readFrame in range(3):
    prot = ''
    for i in range(readFrame, len(rnaString) - readFrame - 1,
    3):
        codon = rnaString[i:i + 3]
        protFrag = dizRNA2Prot[codon]
        if protFrag == 'STOP':
            prot += '*'
        else:
            prot += protFrag
    prots.append(prot)
for prot in prots:
    print prot
```