Programmi Python: esempi

(in Python 3)

Prof.ssa Valeria Vittorini

Prof. Roberto Canonico

Corso di Programmazione I

a.a. 2018-2019

Esempio #1: calcolo numeri primi

- Come esempio di funzione, si riporta sotto il codice di un programma che calcola i numeri primi compresi tra 1 e 1000 mediante una funzione

```
def primes(up_to):
    primes = []
    for n in range(2, up_to + 1):
        is_prime = True
        for divisor in range(2, n):
            if n % divisor == 0:
                is_prime = False
                break
        if is_prime:
            primes.append(n)
    return primes

print(primes(1000))
```

- L'output prodotto è mostrato sotto

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163
, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251
, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349
, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443
, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557
, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647
, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757
, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863
, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983
, 991, 997]
```

Esempio #1-bis: esercizio sui numeri primi

- Problema: determinare il numero di numeri primi presenti in ciascuna centinaia fino a 1000
- Esercizio: si estenda il codice dell'esempio precedente, producendo una lista in cui
  - il primo elemento sia il numero di numeri primi compresi tra 0 e 99 cioè 25
  - il secondo elemento sia il numero di numeri primi compresi tra 100 e 199 cioè 21
  - ....
- Suggerimento:
  - si deve produrre e stampare una lista **`n`** di 10 elementi, in cui **`n[0]`** dovrà contenere il numero di numeri primi compresi tra 0 e 99, e così via
  - si costruisca la lista **`p`** di tutti i numeri primi fino a 1000 usando la funzione **`primes(1000)`**, dopodiché si esegua un ciclo di scansione degli elementi di questa lista **`p`** e, per ogni elemento, se ne determini la centinaia di appartenenza e si aggiorni il corrispondente elemento in **`n`**, incrementandolo di 1
  - Infine, si stampi il vettore **`n`**

Esempio #1-bis: esercizio sui numeri primi (soluzione)

```
def primes(up_to):
    primes = []
    for n in range(2, up_to + 1):
        is_prime = True
        for divisor in range(2, n):
            if n % divisor == 0:
                is_prime = False
                break
        if is_prime:
            primes.append(n)
    return primes

p = primes (1000)
n = [0]*10
for i in range(len(p)):
    c = p[i] / 100
    n[c] += 1
print(n)
```

- L'output prodotto è:

```
[25 , 21, 16, 16, 17, 14, 16, 14, 15, 14]
```

Esempio #2: manipolazione di una stringa

- Siccome le stringhe sono oggetti *immutable*, quando occorre eseguire una manipolazione di una stringa, il programmatore ne deve creare una nuova
- Un tentativo di alterazione diretta dei caratteri di una stringa produce un errore
  - `s = "pippo"; s[0] = 'P'` produce l'errore:
    `'str' object does not support item assignment`
- Il codice riportato sotto mostra una funzione che restituisce una stringa a partire da una stringa fornita come primo argomento, nella quale si opera la sostituzione del carattere fornito come secondo argomento con il carattere fornito come terzo argomento (che per default è '~')

```
def replace(origin_string, char_to_replace, new_char ='~'):
    new_string = ''
    for i in range(len(origin_string)):
        if (origin_string[i] == char_to_replace):
            new_string += new_char
        else:
            new_string += origin_string[i]
    return new_string

a = "Qui, Quo e Qua sono nipoti di Paperino"
print(a)
# produce come output: Qui, Quo e Qua sono nipoti di Paperino
print(replace(a, ' ', '_'))
# produce come output: Qui,_Quo_e_Qua_sono_nipoti_di_Paperino
print(replace(a, ' '))
# produce come output: Qui,~Quo~e~Qua~sono~nipoti~di~Paperino
```

Esempio #3: analisi di dati da file CSV (1)

- Scrivere un programma Python che estragga da un file dati testuale in formato CSV (*comma separated values*) i voti in trentesimi conseguiti da un insieme di studenti e successivamente rappresenti con un istogramma la distribuzione dei voti nel campione
- La struttura del file `voti.csv` è la seguente:

```
Cognome,Nome,Voto,Lode
Amato,Alfredo,21,NO
Andreolli,Antonio,24,NO
Baresi,Carlo,19,NO
Carbonara,Francesco,27,NO
....
```

- Ogni riga contiene una sequenza di dati separati da virgole
- Il significato dei dati è descritto nella prima riga del file

Esempio #3: analisi di dati da file CSV (2)

- Soluzione (prima parte):

```
# File: elabora-voti.py
csv_file = open("voti.csv", "r")
lines = csv_file.readlines()[1:] # Ignora la prima linea del file
votes = []
for line in lines:
    line = line.rstrip('\n')
    data_item = line.split(',')
    cognome = data_item[0]
    nome = data_item[1]
    voto = int(data_item[2])
    lode = data_item[3]
    if (voto == 30) and (lode == "SI"):
        voto = 31                # 30 e lode si rappresenta come 31
    votes.append(voto)

csv_file. close()
....
```

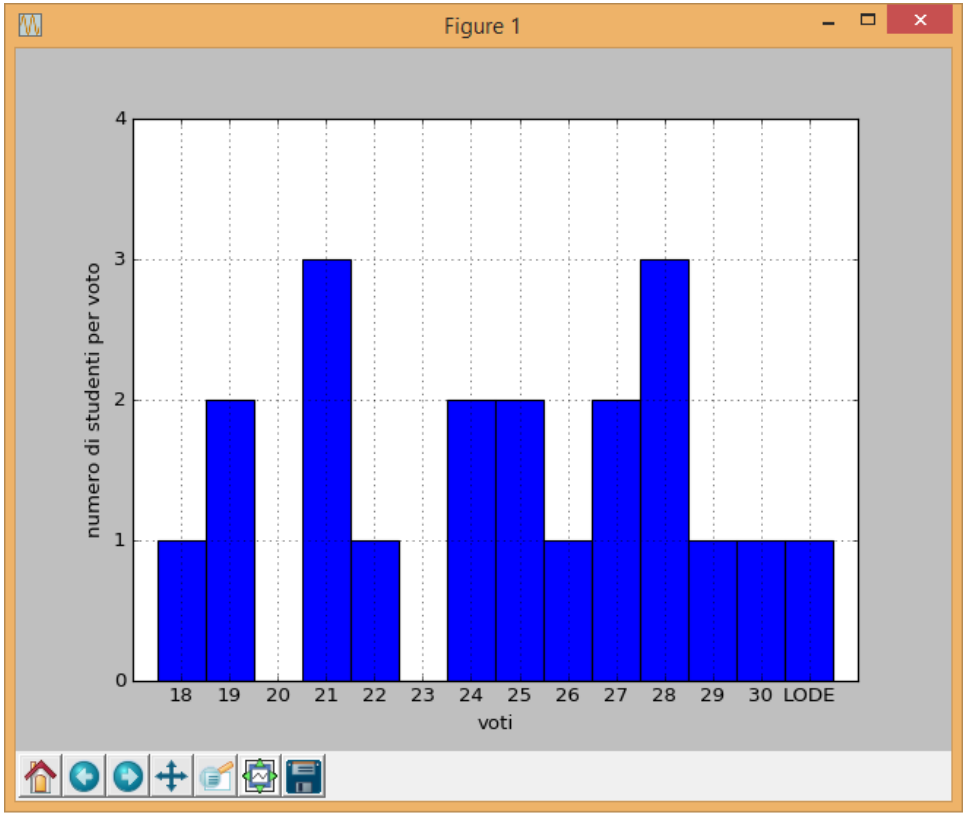
Esempio #3: analisi di dati da file CSV (3)

- Soluzione (seconda parte):

```
# File: elabora-voti.py
....
import matplotlib.pyplot as plt
import numpy as np

hist, bin_edges = np.histogram(votes, bins=np.arange(18,33))

x = range(18,32)
plt.bar (x, hist, align='center', width=1)

x_labels = x[0:-1] + ['LODE']
plt.xticks(x, x_labels)

y_labels = range(0, max(hist)+1+1)
plt.yticks(y_labels)

plt.xlim(17, 32)
plt.ylim(0, max(hist) +1)
plt.xlabel('voti')
plt.ylabel('numero di studenti per voto')
plt.grid(True)

plt.show()
```

Esempio #3: analisi di dati da file CSV (4)

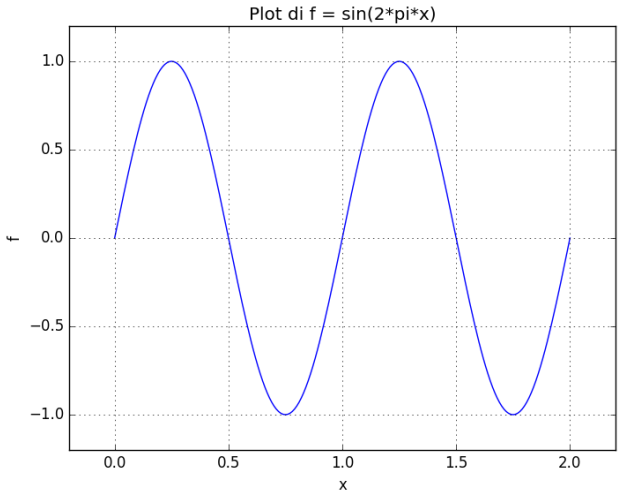
- Il programma produce come output la figura seguente:



Esempio #4: plot di una funzione

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0.0, 2.01, 0.01)
y = np.sin(2*np.pi*x)

plt.plot(x, y)
plt.margins(0.1)

plt.xlabel('x')
plt.ylabel('f')
plt.title('Plot di f = sin(2*pi*x)')
plt.grid(True)
plt.savefig("test.png")
plt.show()
```

- Output prodotto (**`test.png`**):



Esempio #4: plot di una funzione

```
x = np.arange(0.0, 2.01, 0.01)
y = np.sin(2*np.pi*x)
```

- **`np.arange(start,end,step)`** restituisce una lista di numeri **`float`** prodotti a partire da **`start`** con incremento **`step`** fino al valore massimo **`end`** (escluso)

```
[ 0.    0.01  0.02  0.03  0.04  0.05  0.06  0.07  0.08  0.09  0.1   0.11
  0.12  0.13  0.14  0.15  0.16  0.17  0.18  0.19  0.2   0.21  0.22  0.23
  0.24  0.25  0.26  0.27  0.28  0.29  0.3   0.31  0.32  0.33  0.34  0.35
  0.36  0.37  0.38  0.39  0.4   0.41  0.42  0.43  0.44  0.45  0.46  0.47
  0.48  0.49  0.5   0.51  0.52  0.53  0.54  0.55  0.56  0.57  0.58  0.59
  0.6   0.61  0.62  0.63  0.64  0.65  0.66  0.67  0.68  0.69  0.7   0.71
  0.72  0.73  0.74  0.75  0.76  0.77  0.78  0.79  0.8   0.81  0.82  0.83
  0.84  0.85  0.86  0.87  0.88  0.89  0.9   0.91  0.92  0.93  0.94  0.95
  0.96  0.97  0.98  0.99  1.    1.01  1.02  1.03  1.04  1.05  1.06  1.07
  1.08  1.09  1.1   1.11  1.12  1.13  1.14  1.15  1.16  1.17  1.18  1.19
  1.2   1.21  1.22  1.23  1.24  1.25  1.26  1.27  1.28  1.29  1.3   1.31
  1.32  1.33  1.34  1.35  1.36  1.37  1.38  1.39  1.4   1.41  1.42  1.43
  1.44  1.45  1.46  1.47  1.48  1.49  1.5   1.51  1.52  1.53  1.54  1.55
  1.56  1.57  1.58  1.59  1.6   1.61  1.62  1.63  1.64  1.65  1.66  1.67
  1.68  1.69  1.7   1.71  1.72  1.73  1.74  1.75  1.76  1.77  1.78  1.79
  1.8   1.81  1.82  1.83  1.84  1.85  1.86  1.87  1.88  1.89  1.9   1.91
  1.92  1.93  1.94  1.95  1.96  1.97  1.98  1.99  2.  ]
```

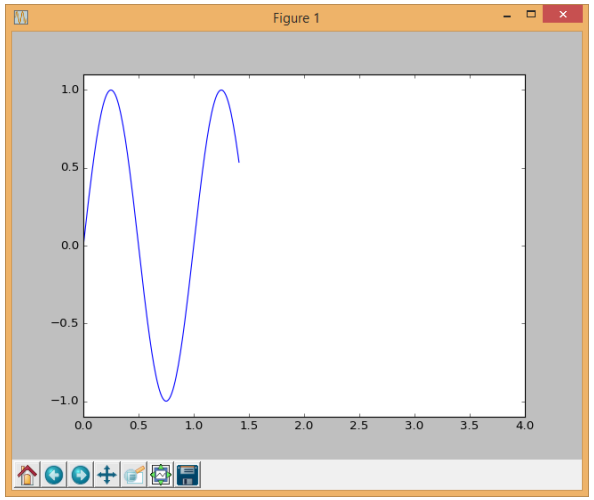
- **`np.sin(2*np.pi*x)`** (essendo **`x`** una lista) restituisce una lista di numeri **`float`** prodotti applicando la funzione sin(2*np.pi*x) a ciascun elemento di x

```
[  0.00000000e+00   6.27905195e-02   1.25333234e-01   1.87381315e-01
   2.48689887e-01   3.09016994e-01   3.68124553e-01   4.25779292e-01
   4.81753674e-01   5.35826795e-01   5.87785252e-01   6.37423990e-01
   6.84547106e-01   7.28968627e-01   7.70513243e-01   8.09016994e-01
   8.44327926e-01   8.76306680e-01   9.04827052e-01   9.29776486e-01
   9.51056516e-01   9.68583161e-01   9.82287251e-01   9.92114701e-01
...
```

Esempio #5: plot animato di una funzione

```
import math
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

x = []
y = []
x0 = 0.0
delta = 0.01

fig = plt.figure()
ax = plt.axes()
ax.set_ylim(-1.1, 1.1)
ax.set_xlim(0, 4)
lineS, = ax.plot(x, y)

def init():
    lineS.set_data([], [])
    return lineS,

def animate(i):
    current_x = x0+delta*i
    current_y = math.sin(2*math.pi*current_x)
    x.append(current_x)
    y.append(current_y)
    lineS.set_data(x, y)
    return lineS,

anim = animation.FuncAnimation(fig, animate, interval=100)
plt.show()
```

- Output prodotto:



Esempio #6: reperimento di un file dal web

```
import urllib.request
import os

filename = "Elenco-comuni-italiani.csv"
url = "https://www.istat.it/storage/codici-unita-amministrative/Elenco-comuni-italiani.csv"

if (not os.path.isfile(filename)):
    print("Il file", filename, "non e' presente. Lo scarico dal sito ISTAT.")
    urllib.request.urlretrieve(url, filename))
    print("Il file", filename, "e' stato scaricato dal sito ISTAT.")
else:
    print("Il file", filename, "e' gia' disponibile localmente.")
```

Esempio #7: analisi di un file CSV

```
import os

...


csv_file = open(filename, "r", encoding="latin-1")
lines = csv_file.readlines()[1:] # Ignora la prima linea del file
comuni = {}      # Si usa un dizionario per mantenere le coppie comune:codice_catastale
i = 1
for line in lines:
    line = line.rstrip('\n')
    data_item = line.split(';')
    comune = data_item[5]
    codice_catastale = data_item[18]
    if (comune == ""):
        continue
    comuni.update({comune : codice_catastale})

csv_file.close()

while (1):
    comune = input("Inserisci un comune italiano: ")
    codice = comuni.get(comune)
    if (codice != None):
        print("Codice catastale di", comune, "=", codice)
    else:
        print("Comune non esistente in Italia")
```