


ESERCITAZIONE WEEK 7 DAY 3

 W7D1 - Pratica (2) PDF

Esercizio
Python per Hacker Pt. 2

Traccia:

Scrivi una funzione generatrice di password.

La funzione deve generare una stringa alfanumerica di 8 caratteri qualora l'utente voglia una password semplice, o di 20 caratteri ascii qualora desideri una password più complicata.

Per questo esercizio è stato scritto codice Python su un IDE online.

Si riporta il codice e l'esecuzione dei tre casi d'uso:

- Utente preme 'a' per avere una password semplice di 8 caratteri alfanumerici;
- Utente preme 'b' per avere una password più complessa di 20 caratteri ascii;
- Utente preme qualsiasi tasto diverso dai precedenti e il sistema risponderà con un messaggio di "Selezione non valida".

```
7
8 import string
9 import random
10 x=[]
11 y=[]
12 caratteri=string.digits + string.ascii_letters |
13
14 user_input= input("Digita a per una password semplice, b per una password complessa: ")
15 if(user_input=='a'):
16     for i in range(8):
17         x.insert(len(x),random.choice(caratteri))
18     password=''.join(x)
19     print("Ecco una password semplice:",password)
20 elif(user_input=='b'):
21     for i in range(20):
22         y.insert(len(y),random.choice(caratteri))
23     password=''.join(y)
24     print("Ecco una password complessa:",password)
25 else:
26     print("Selezione non valida")
27
```

input

Digita a per una password semplice, b per una password complessa: a
Ecco una password semplice: MTM6uXaf

Innanzitutto è necessario importare nel codice i moduli **string** e **random**, per avere a disposizione le funzioni e costanti che servono per generare un stringa alfanumerica.

Si definiscono due liste vuote che verranno usate successivamente e si definisce una stringa **caratteri** che concatena due costanti di tipo stringa del metodo **string**: `string.digits` (che contiene solo caratteri ASCII relativi alle cifre da 0 a 9) e `string.ascii_letters` (che contiene l'ASCII di lettere maiuscole e minuscole).

Con la funzione **input** si chiede all'utente quale scelta voglia fare tra le elencate in precedenza.

La scelta è gestita dal costrutto **if-elif-else**.

Nel caso si selezioni 'a' viene fatto un ciclo **for** fino a 8 (lunghezza definita dal primo caso) e si riempie la lista `x` con il metodo **insert**, con un carattere alfanumerico ottenuto randomicamente con la funzione **random.choice** facente parte del modulo importato.

Con la funzione **join** si concatenano gli elementi della lista `x` in una stringa chiamata **password** che verrà stampata con la funzione **print**. Gli apici vuoti indicano che non devono esserci separatori.

Nel caso 'b', il ragionamento è identico, ma il range del `for` arriva fino a 20.

Si riportano le esecuzioni degli altri casi:

```
7
8 import string
9 import random
10 x=[]
11 y=[]
12 caratteri=string.digits + string.ascii_letters |
13
14 user_input= input("Digita a per una password semplice, b per una password complessa: ")
15 if(user_input=='a'):
16     for i in range(8):
17         x.insert(len(x),random.choice(caratteri))
18     password=''.join(x)
19     print("Ecco una password semplice:",password)
20 elif(user_input=='b'):
21     for i in range(20):
22         y.insert(len(y),random.choice(caratteri))
23     password=''.join(y)
24     print("Ecco una password complessa:",password)
25 else:
26     print("Selezione non valida")
27
```

input

Digita a per una password semplice, b per una password complessa: b
Ecco una password complessa: hIf5lazjDp98Dv8AIjwC

```
7
8 import string
9 import random
10 x=[]
11 y=[]
12 caratteri=string.digits + string.ascii_letters |
13
14 user_input= input("Digita a per una password semplice, b per una password complessa: ")
15 if(user_input=='a'):
16     for i in range(8):
17         x.insert(len(x),random.choice(caratteri))
18     password=''.join(x)
19     print("Ecco una password semplice:",password)
20 elif(user_input=='b'):
21     for i in range(20):
22         y.insert(len(y),random.choice(caratteri))
23     password=''.join(y)
24     print("Ecco una password complessa:",password)
25 else:
26     print("Selezione non valida")
27
```

input

Digita a per una password semplice, b per una password complessa: c
Selezione non valida

Se si volessero creare password ancora più complesse con caratteri ASCII alfanumerici e simbolici, si può concatenare nella stringa **caratteri** anche il metodo `string.punctuation`:

```
7
8 import string
9 import random
10 x=[]
11 y=[]
12 caratteri=string.digits + string.ascii_letters + string.punctuation
13
14 user_input= input("Digita a per una password semplice, b per una password complessa: ")
15 if(user_input=='a'):
16     for i in range(8):
17         x.insert(len(x),random.choice(caratteri))
18     password=''.join(x)
19     print("Ecco una password semplice:",password)
20 elif(user_input=='b'):
21     for i in range(20):
22         y.insert(len(y),random.choice(caratteri))
23     password=''.join(y)
24     print("Ecco una password complessa:",password)
25 else:
26     print("Selezione non valida")
27
```

input

Digita a per una password semplice, b per una password complessa: a
Ecco una password semplice: 01-4^291

```
7
8 import string
9 import random
10 x=[]
11 y=[]
12 caratteri=string.digits + string.ascii_letters + string.punctuation
13
14 user_input= input("Digita a per una password semplice, b per una password complessa: ")
15 if(user_input=='a'):
16     for i in range(8):
17         x.insert(len(x),random.choice(caratteri))
18     password=''.join(x)
19     print("Ecco una password semplice:",password)
20 elif(user_input=='b'):
21     for i in range(20):
22         y.insert(len(y),random.choice(caratteri))
23     password=''.join(y)
24     print("Ecco una password complessa:",password)
25 else:
26     print("Selezione non valida")
27
```

input

Digita a per una password semplice, b per una password complessa: b
Ecco una password complessa: *.chM"X|?Uu9,ap@4tl}

Infine per migliorare l'esercizio, si definisce dal principio una funzione **genera_password** che verrà richiamata nel costrutto **if-elif-else** senza ripetere inutilmente codice più volte:

```
8 import string
9 import random
10 x=[]
11 caratteri=string.digits + string.ascii_letters + string.punctuation
12
13 def genera_password(numero_max):
14     global password
15     for i in range(numero_max):
16         x.insert(len(x),random.choice(caratteri))
17     password=''.join(x)
18     return password
19
20 user_input= input("Digita a per una password semplice, b per una password complessa: ")
21 if(user_input=='a'):
22     genera_password(8)
23     print("Ecco una password semplice:",password)
24 elif(user_input=='b'):
25     genera_password(20)
26     print("Ecco una password complessa:",password)
27 else:
28     print("Selezione non valida")
```

input

Digita a per una password semplice, b per una password complessa: b
Ecco una password complessa: bCitw*5eNgKwD!dh2Zrh

Per rendere visibile la variabile locale **password** al di fuori della funzione, viene resa globale con la parola *global*.