INDICE

INTRODUZIONE ED OBIETTIVO

INTRODUZIONE: L'esercizio facoltativo proposto si concentra sulla creazione di una funzione dedicata alla generazione automatica di password.
L'attività ha come obiettivo l'applicazione pratica dei concetti di modularità del codice, dell'uso dei cicli e della gestione delle stringhe, mostrando come il linguaggio possa essere utilizzato per risolvere problemi concreti della vita digitale quotidiana, come la creazione di chiavi sicure e personalizzabili.
Il programma integra l'utilizzo dei moduli **random e string**, strumenti fondamentali per la generazione di valori casuali e per la manipolazione di insiemi di caratteri predefiniti.

OBIETTIVO: L'obiettivo dell'attività è sviluppare una funzione capace di generare una password in modo completamente automatico, offrendo all'utente la possibilità di scegliere tra due livelli di complessità: una password "semplice" di 8 caratteri alfanumerici o una password "complessa" di 20 caratteri comprendenti anche simboli ASCII.
Attraverso la realizzazione di questo programma, si mira a rafforzare la comprensione dei cicli iterativi, della costruzione di stringhe dinamiche e della gestione condizionale delle scelte dell'utente.

IMPORTAZIONE LIBRERIE

```
1   #IMPORTO LIBRERIRE
2   import random #PER SCELTE CASUALI
3   import string #MODULO STRING CONTENTENTE LETTERE,NR E SIMBOLI
4
```

Nella fase iniziale del programma vengono importate le librerie fondamentali per il corretto funzionamento dello script: **random e string**.

La prima consente di generare scelte casuali, elemento essenziale per creare password imprevedibili, mentre la seconda mette a disposizione insiemi predefiniti di caratteri (lettere maiuscole, minuscole, numeri e simboli)utili per costruire la base su cui il programma potrà operare.

In questo contesto, **l'unione di random e string** permette di generare una combinazione sempre diversa di caratteri, garantendo unicità e complessità a ogni password prodotta.

FUNZIONE PRINCIPALE

```python
#FUNZIONE PRINCIPALE
def genera_password(): # CREA UNA PSW IN BASE ALLA SCELTA DELL'UTENTE
    print("Questo programma genera una password automaticamente.")

    scelta = input("Vuoi una password semplice o complessa? (S/C): ").upper() #CHIEDE IL TIPO DI PSW

    if scelta == "C": #SE L'UTENTE SCEGLI COMPLESSA
        lunghezza = 20
        caratteri = string.ascii_letters + string.digits + string.punctuation #USA LETTERE,NR E SIMBOLI
    else: #SE L'UTENTE SCEGLIE SEMOLICE
        lunghezza = 8
        caratteri = string.ascii_letters + string.digits # USA NR E LETTERE

    password = "" #VARIABILE VUOTA PER COSTITUIRE PSW
    for i in range(lunghezza):   #AGGIUNGE UN CARATTERE CASUALE FINO ALLA LUNGHEZZA DESIDERATA
        password += random.choice(caratteri)

    print(f"La tua password è: {password}") #MOSTRA LA PSW A SCHERMO
```

La parte centrale del programma è dedicata alla definizione della **funzione genera_password()**, cuore logico dell'intero esercizio.
All'interno della funzione, viene innanzitutto mostrato un messaggio introduttivo che comunica all'utente lo scopo del programma: generare automaticamente una password sicura. Successivamente, attraverso **il comando input()**, l'utente è invitato a scegliere tra due opzioni **una password "semplice"** o una **"complessa"** digitando la lettera corrispondente.
L'uso del **metodo .upper()** garantisce che la risposta venga interpretata correttamente anche se inserita in minuscolo, migliorando così la flessibilità e l'affidabilità del codice.
La struttura **condizionale if-else** permette di personalizzare il comportamento del programma in base alla scelta effettuata: nel caso di una **password complessa**, la lunghezza viene impostata a *20 caratteri* e vengono utilizzati lettere, numeri e simboli (**string.ascii_letters + string.digits + string.punctuation**), garantendo così un elevato livello di sicurezza.
Per la **versione semplice**, invece, la lunghezza è limitata a *8 caratteri* e la combinazione si riduce a lettere e numeri, mantenendo comunque una buona varietà ma con una struttura più essenziale.
A seguire, il programma crea una **variabile vuota** chiamata **password**, che fungerà da contenitore per la costruzione della stringa finale.
Ciclo for scorre poi per il numero di volte definito dalla variabile lunghezza, selezionando casualmente un carattere per ogni iterazione grazie alla **funzione random.choice().**
In questo modo, la password prende forma progressivamente, fino a raggiungere la lunghezza desiderata.
Infine, il risultato viene mostrato all'utente tramite la **funzione print(),** che visualizza la password appena generata.

ESECUZIONE

```
#ESECUZIONE PROGRAMMA
genera_password() #AVVIA FUNZIONE PER GENERARE PSWcd "C:\EP. ESERCIZI\PYTHON"
```

```
Questo programma genera una password automaticamente.
Vuoi una password semplice o complessa? (S/C): s
La tua password è: 94hZ9x3L
PS C:\EP. ESERCIZI\PYTHON> python FACOLTATIVOM2W7D1.py
>>
Questo programma genera una password automaticamente.
Vuoi una password semplice o complessa? (S/C): c
La tua password è: '"6;y@|/N<f_Fqv;,(^x
PS C:\EP. ESERCIZI\PYTHON>
```

La parte finale del programma consiste **nell'esecuzione della funzione genera_password()**, che viene richiamata al di fuori della sua definizione per avviare effettivamente il processo di generazione della password.
Questa singola istruzione rappresenta il momento in cui il codice scritto diventa operativo, trasformando la logica implementata in un'azione concreta.
Una volta eseguito il file tramite terminale, il programma accoglie l'utente con un messaggio introduttivo e una semplice domanda: scegliere se generare una password semplice o complessa.
Dopo la selezione, la funzione elabora in tempo reale una combinazione di caratteri casuali e la visualizza a schermo.
La password risultante varia ad ogni esecuzione, garantendo così unicità e sicurezza (due aspetti fondamentali per qualsiasi sistema di generazione automatica di credenziali).
Durante il test, il programma ha prodotto correttamente password sia di otto caratteri (nel caso della modalità semplice) sia di venti caratteri (per la modalità complessa), includendo lettere, numeri e simboli secondo la configurazione scelta.
Il comportamento osservato conferma la stabilità del codice e l'efficacia dell'uso combinato delle librerie random e string, che assicurano la completa casualità delle password generate.

CONCLUSIONE

La funzione sviluppata combina semplicità strutturale e potenza logica, mostrando come poche righe di codice possano svolgere operazioni complesse grazie all'integrazione di moduli standard e cicli iterativi.
Dal punto di vista formativo, l'attività ha offerto l'occasione di approfondire la gestione delle scelte condizionali, l'uso dei moduli random e string, e la costruzione dinamica di stringhe.
Oltre al valore tecnico, il progetto mette in evidenza un principio fondamentale della programmazione: la capacità di risolvere problemi pratici con eleganza e chiarezza.
In un certo senso, la generazione di una password può essere paragonata alla scrittura di un piccolo algoritmo di difesa personale, dove ogni carattere casuale rappresenta un tassello di sicurezza in un mondo sempre più digitale.

GLOSSARIO

Funzione (def) – Blocco di codice che esegue un'azione specifica. In questo esercizio, la funzione genera_password() è responsabile della creazione automatica della password in base alle scelte dell'utente.
Libreria (import) – Collezione di moduli predefiniti che estendono le funzionalità base. In questo caso sono state utilizzate random e string.
Modulo random – Strumento che consente di generare numeri o elementi in modo casuale, utile per ottenere combinazioni imprevedibili e uniche.
Modulo string – Raccolta di costanti che rappresentano insiemi di caratteri predefiniti (lettere, numeri e simboli), fondamentali per costruire password diversificate.
Ciclo for – Struttura iterativa che permette di ripetere un'azione più volte. Qui viene utilizzata per aggiungere progressivamente i caratteri casuali fino a comporre la password completa.
Condizione if-else – Costrutto logico che consente di scegliere fra due percorsi di esecuzione differenti in base alla decisione dell'utente.
Variabile – Spazio di memoria identificato da un nome, utilizzato per conservare valori modificabili, come la password in costruzione o la lunghezza scelta.
Funzione input() – Comando che permette all'utente di interagire con il programma, fornendo un valore o una scelta durante l'esecuzione.
Funzione print() – Istruzione che mostra a schermo il risultato finale, consentendo di visualizzare la password generata.
Metodo .upper() – Funzione che converte le lettere in maiuscolo, assicurando che la scelta dell'utente venga riconosciuta anche se scritta in minuscolo.