Python – Generator bezpiecznego hasła

O module **string** - opis:

String constants. The constants defined in this module are:

string.ascii_letters

The concatenation of the ascii_lowercase and ascii_uppercase constants described below. This value is not locale-dependent.

string.ascii_lowercase

The lowercase letters 'abcdefghijklmnopqrstuvwxyz'. This value is not locale-dependent and will not change.

string.ascii_uppercase

The uppercase letters 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'. This value is not locale-dependent and will not change.

string.digits

The string '0123456789'.

string.hexdigits

The string '0123456789abcdefABCDEF'.

string.octdigits

The string '01234567'.

string.punctuation

String of ASCII characters which are considered punctuation characters in the C locale: !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~.

string.printable

String of ASCII characters which are considered printable. This is a combination of digits, ascii_letters, punctuation, and whitespace.

string.whitespace

A string containing all ASCII characters that are considered whitespace. This includes the characters space, tab, linefeed, return, formfeed, and vertical tab.

Uruchamiamy nowy projekt.

Importujemy potrzebne biblioteki.

```
main.py
1    # praktyczny prokekt - generator hasła
2
3    # 1. importowane biblioteki
4    import sys      # --- pozwala na korzystanie z właściwości systemu
5    import random   # --- pozwala na obsługę liczb losowych
6    import string   # --- pozwala na obsługę ciągów znaków ASCII
```

Tworzymy zmienną tablicową przechowującą kolejne znaki naszego hasła:

```
8    # 2. pomocnicze zmienne
9    password = []
```

Tworzymy zmienną przechowującą długość tworzonego hasła (zmienna przechowuje liczbę integer):

```
10   password_length = int(input("Jak długie ma być hasło? "))
```

Oczywiście na tym etapie nie mamy kontroli wpisywanych znaków (mszą to być liczby całkowite).

Jeżeli użytkownik poda jakąś małą liczbę poniżej 8 to program zakończy działanie.

```
11   if password_length < 5:
12       print("Hasło musi mieć minimum 5 znaków, spróbuj jeszcze raz.")
13       sys.exit(0) # koniec programu
```

Ustalamy pozostałe warunki tworzenia hasła:

```
15   # 3. ustalamy pozostałe warunki hasła
16   # --- ile małych liter
17   lowercase_letters = int(input("Ile małych liter ma mieć hasło? "))
18
19   # --- ile dużych liter
20   uppercase_letters = int(input("Ile dużych liter ma mieć hasło? "))
21
22   # --- ile znaków specjalnych
23   special_characters = int(input("Ile znaków specjalnych ma mieć hasło? "))
24
25   # --- ile cyfr
26   digits = int(input("Ile cyfr ma mieć hasło? "))
27
```

Ze względu na możliwość podania abstrakcyjnych wartości znaków utworzymy pomocniczą zmienną przechowującą ilość znaków do wprowadzenia podczas wpisywania:

```
8    # 2. pomocnicze zmienne
9    password = []
10   characters_left = -1        # --- zmienna musi mieć wartość początkową
11   password_length = int(input("Jak długie ma być hasło? "))
```

Zmienna ta przyjmuje w ‘elsie’ wartość ilości znaków do wprowadzenia:

```
12    if password_length < 8:
13        print("Hasło musi mieć minimum 8 znaków, spróbuj jeszcze raz.")
14        sys.exit(0) # koniec programu
15    else:
16        characters_left = password_length
17        print("Pozostało do wpisania", characters_left,  "znaków:")
```

Podczas wprowadzania znaków musimu uaktualniać i kontrolować wprowadzony znak/wprowadzone znaki. Wykonujemy to w pętli/pętlach:

```
20    # --- ile małych liter
21    lowercase_letters = int(input("Ile małych liter ma mieć hasło? "))
22    if lowercase_letters > password_length or lowercase_letters < 0:
23        print("Liczba znaków jest poza przedziałem 0 ..",characters_left)
24        sys.exit(0)
25    else:
26        characters_left -= lowercase_letters # --- aktualizacja liczby znaków do użycia
27        print("Liczba znaków pozostałych do użycia:", characters_left)
```

Dla pozostałych znaków:

```
28    # --- ile dużych liter
29    uppercase_letters = int(input("Ile dużych liter ma mieć hasło? "))
30    if uppercase_letters > characters_left or uppercase_letters < 0:
31        print("Liczba znaków jest poza przedziałem 0 ..",characters_left)
32        sys.exit(0)
33    else:
34        characters_left -= uppercase_letters # --- aktualizacja liczby znaków do użycia
35        print("Liczba znaków pozostałych do użycia:", characters_left)
36    # --- ile znaków specjalnych
37    special_characters = int(input("Ile znaków specjalnych ma mieć hasło? "))
38    if special_characters > characters_left or special_characters < 0:
39        print("Liczba znaków jest poza przedziałem 0 ..",characters_left)
40        sys.exit(0)
41    else:
42        characters_left -= special_characters # --- aktualizacja liczby znaków do użycia
43        print("Liczba znaków pozostałych do użycia:", characters_left)
44    # --- ile cyfr
45    digits = int(input("Ile cyfr ma mieć hasło? "))
46    if digits > characters_left or digits < 0:
47        print("Liczba znaków jest poza przedziałem 0 ..",characters_left)
48        sys.exit(0)
49    else:
50        characters_left -= digits # --- aktualizacja liczby znaków do użycia
51        print("Liczba znaków pozostałych do użycia:", characters_left)
```

Ze względu na powtarzający się kod warto go wyodrębnić w formie funkcji i wywoływać tylko funkcję:

```
# 4. aktualizacja długości hasła - funkcja ogólna:
def update_characters_left(number_of_characters):
    global characters_left
    if number_of_characters < 0 or number_of_characters > characters_left:
        print("Liczba znaków spoza przedziału 0,", characters_left)
        sys.exit(0)
    else:
        characters_left -= number_of_characters
        print("Pozostało znaków:", characters_left)

# 3. ustalamy pozostałe warunki hasła
# --- ile małych liter
lowercase_letters = int(input("Ile małych liter ma mieć hasło? "))
update_characters_left(lowercase_letters)
# --- ile dużych liter
uppercase_letters = int(input("Ile dużych liter ma mieć hasło? "))
update_characters_left(uppercase_letters)
# --- ile znaków specjalnych
special_characters = int(input("Ile znaków specjalnych ma mieć hasło? "))
update_characters_left(special_characters)
# --- ile cyfr
digits = int(input("Ile cyfr ma mieć hasło? "))
update_characters_left(digits)

```

Jeżeli zadeklarujemy 20 znaków a wpiszemy hasło z przedziału 8 .. 20 to zostaną nam niewykorzystane znaki. W tej sytuacji możemy przyjąć, że będą one stanowić małe titery. Np. zadeklarowaliśmy:

Długość hasła 20 znaków oraz

- Małe: 5
- Duże: 3
- Specjalne: 2
- Cyfry: 5
  To zostaje 5 znaków dopisanych do małych liter więc małych liter w tym przykładzie powinno być 10.

Tą funkcję realizujemy za pomocą if:

```
update_characters_left(digits)
# 5. uzupełnienie ilości małych liter
if characters_left > 0:
    lowercase_letters += characters_left
    print("Nie wszystkie znaki zostały wykorzystane. Hasło zostanie uzupełnione małymi literami.")
    print("Małych liter ma być:",lowercase_letters)
```

Dodatkowe podsumowanie ilości liter do wpisania:

```
47  # 6. podsumowanie
48  print()
49  print("Długość hasła:", password_length)
50  print("Małe litery:", lowercase_letters)
51  print("Duże litery:", uppercase_letters)
52  print("Znaki specjalne:", special_characters)
53  print("Cyfry:", digits)
```

Wczytujemy hasło (używamy tyle znaków ile jest zadeklarowanych powyżej:

```
47  # 6. podsumowanie
48  print()
49  print("Długość hasła:", password_length)
50  print("Małe litery:", lowercase_letters)
51  print("Duże litery:", uppercase_letters)
52  print("Znaki specjalne:", special_characters)
53  print("Cyfry:", digits)
54  # 7. petla wczytywania znaków
55  for _ in range(password_length):    # symbol '_' zastepuje iterator 'i'
56      if lowercase_letters > 0:
57          password.append(random.choice(string.ascii_lowercase))
58          lowercase_letters -= 1
59      if uppercase_letters > 0:
60          password.append(random.choice(string.ascii_uppercase))
61          uppercase_letters -= 1
62      if special_characters > 0:
63          password.append(random.choice(string.punctuation))
64          special_characters -= 1
65      if digits > 0:
66          password.append(random.choice(string.digits))
67          digits -= 1
```

Pozostało wyświetlić dodatkowo hasło:

```
68  # wygenerowane hasło:
69  print("Wygenerowane hasło:", "".join(password))
70  # generowanie innego hasła z wylosowanych znaków
71  random.shuffle(password)
72  print("Wygenerowane hasło:", "".join(password))
```

Przykład 2-gi jednolinijkowy:

```
74  # 10. PRZYKLAD 2
75  password = "".join([random.choice(string.ascii_letters) for _ in range(10)])
76  print("Wygenerowane hasło 2:", password)
```

Modyfikacje: sprawdzanie wprowadzanych znaków bez przedwczesnego kończenia programu.