Lab 5 – Słowniki

W Pythonie słownik jest kolekcją par klucz – wartość.
Każdy klucz jest połączony z wartością, za pomocą klucza można uzyskać dostęp do powiązanej z nim wartości.
Słownik możemy utworzyć za pomocą nawiasów klamrowych, parą kluczy i wartości rozdzieloną dwukropkiem. Wiele par klucz-wartość w słowniku rozdzielamy przecinkiem:

```
users = {
        "lukasz": "email@gmail.com",
        "joe": "joe@gmail.com",
        "test": "test@gmail.com",
        "admin": "admin@gmail.com"}
```

W tym wypadku przechowujemy jeden rodzaj informacji o wielu obiektach.
Możemy też przechować rożne informacje o jednym obiekcie:

```
user1 = {"name": "lukasz", "email": "email@gmail.com"}
user2 = {"name": "joe", "email": "joe@gmail.com"}
user3 = {"test": "test@gmail.com"}
user4 = {"admin": "admin@gmail.com"}
```

Aby pobrać wartość ze słownika używamy metody get():
Podajemy w niej klucz, opcjonalnie co należy zwrócić gdy klucz nie istnieje w słowniku:

```
print(users.get("admin2", "invalid username"))
```

Innym sposobem uzyskania dostępu do wartości słownika jest podanie nazwy słownika, oraz nazwy klucza w nawiasie kwadratowym:

```
user1["password"]
```

W tym wypadku, gdy klucz nie istnieje dostaniemy błąd:

```
Traceback (most recent call last):
  File "/Users/lukaszkwasniewicz/Downloads/BigPro/slowniki.py", line 23, in <module>
    print(user1["password"])
          ~~~~~^^^^^^^^^^^^^
KeyError: 'password'
```

Aby dodać parę klucz-wartość do słownika należy podać nazwę słownika, oraz nazwę nowego klucza w nawiasie kwadratowym oraz wartość przypisywaną do klucza po znaku „=”:

```
user1["password"] = "haslo"
```

Lub za pomocą metody update:

```
user2.update({"password": "ttt"})
```

Aby zmodyfikować wartość w słowniku analogicznie jak poprzednio należy podać nazwę słownika, oraz nazwę klucza w nawiasie kwadratowym oraz wartość przypisywaną do klucza po znaku „=”:

```
user1["password"] = "hasloTajne"
print(user1["password"])
```

Za pomocą polecenia del, można całkowicie usunąć parę klucz-wartość, podajemy nazwę słownika oraz klucz do usunięcia w nawiasie kwadratowym:

```
user1[1] = "test"
print(user1)
# usuwanie par klucz wartosc:
if 1 in user1:
	del user1[1]
```

ze zwracaniem wartości:

```
print( user1.pop("name"))
```

Aby wstawić imię na początek (od Python'a 3.7 słownik zachowuje kolejność wstawiania) należy wykonać update:

```
up_dict = {"name": "lukasz"}
up_dict.update(user1)
user1 = up_dict
print(user1)
```

Iteracja przez słownik:

Iteracja przez klucze – jest domyślnym zachowaniem podczas iteracji przez słownik:

```
for e in users:
	print(e)
klucze = users.keys()
print(klucze)
for e in users.keys():
	print(e)
```

Iteracja klucz-wartość – konieczne jest utworzenie dla pętli for dwóch zmiennych przechowujących klucz i wartość (nazwy dowolne – w przykładzie key, value). Iteracja następuje po widoku par klucz-wartość zwracanej przez metodę items() :

```
for key, value in users.items():
	print("Key is " + str(key) + " value is " + str(value))
```

Instrukcja warunkowa, np. sprawdzenie czy użytkownik posiada rolę administratora:

```
admin_role = ["admin", "lukasz"]

for key, value in users.items():
	print("Key is " + str(key) + " value is " + str(value))
	if key in admin_role:
		print(key + " is also an administrator !!")
```

Iteracja po wartościach:

```
for value in users.values():
	print(value)
```

Sortowanie kluczy:

```
sorted_dict = {}
for key in sorted(users.keys()):
    value = users[key]
    sorted_dict[key] = value
print(sorted_dict)

sorted_dict = {}
```

```
for key in sorted(users.keys(), reverse=True):
    value = users[key]
    sorted_dict[key] = value
print(sorted_dict)
```

Sortowanie po wartości wymaga użycia metody sorted z parametrem key – funkcja, która opisuje w jaki sposób będą sortowane elementy.

```
users = {
        "lukasz": 100,
        "joe": 90,
        "test": 60,
        "admin": 75}
print(users)
def get_value(item):
        return item[1]
sorted_values = sorted(users.items(), key=get_value)
```

Zamieniamy listę krotek na słownik:

```
print(sorted_values)
sorted_values = dict(sorted_values)
print(sorted_values)
```

Zwracanie klucza po wartości:

```
def get_key(value, d):
        for key, v in d.items():
                if v == value:
                        return key
        return None
# test:
print(get_key(100, users))
```

Zagnieżdżanie – przechowywanie słowników w liście, list w słownikach oraz słowników w słownikach nazywamy zagnieżdżaniem.

```
# lista słowników:
dict_list = []
dict_list.append(user1)
dict_list.append(user2)
dict_list.append(user3)
dict_list.append(user4)
```

Warunkowa zmiana hasła – jeżeli użytkownik nie posiadał hasła jest dodawane nowe:

```
for i in range(0, len(dict_list)):
        if "password" not in dict_list[i]:
                dict_list[i]["password"] = "haslo2"
```

Do słownika można dodać również listę:

```
for i in range(0, len(dict_list)):
```

```
        dict_list[i]["adresy"] = ["koszykowa 86",
                                  "akademicka 9"]
print(dict_list[0]["adresy"][0])
```

Słownik w słowniku:

```
for i in range(0, len(dict_list)):
        dict_list[i]["adresy"] = {"adres1": "koszykowa 86",
                                  "adres2": "akademicka 9"}
print(dict_list[0]["adresy"]["adres2"])
```

Czyszczenie słownika oraz kopia głęboka:

```
import copy
```

```
users_copy = copy.deepcopy(users)
print(users_copy)
users_copy.clear()
print(users_copy)
print(users)
```

Wczytywanie pliku linia po linii:

```
filepath = "students2.txt"
with open(filepath) as file_object:
        for line in file_object:
                print(line.rstrip())
```

Zapisywanie do pliku:

```
filepath = "studentsOut.txt"
students = ["jan", "nowak"]
with open(filepath, "w") as file_object:
        for e in students:
                line = f"witaj {e} !\n"
                file_object.write(line)
```

Wysyłanie e-maili za pomocą smtp:

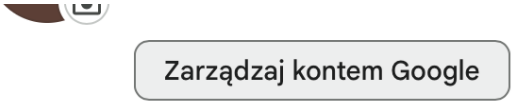
```
import smtplib
from email.mime.text import MIMEText
def send_email(subject, body, sender, recipients, password):
        msg = MIMEText(body)
        msg['Subject'] = subject
        msg['From'] = sender
        msg['To'] = ', '.join(recipients)
        smtp_server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
        smtp_server.login(sender, password)
        smtp_server.sendmail(sender, recipients, msg.as_string())
        smtp_server.quit()
```

Przykład użycia:

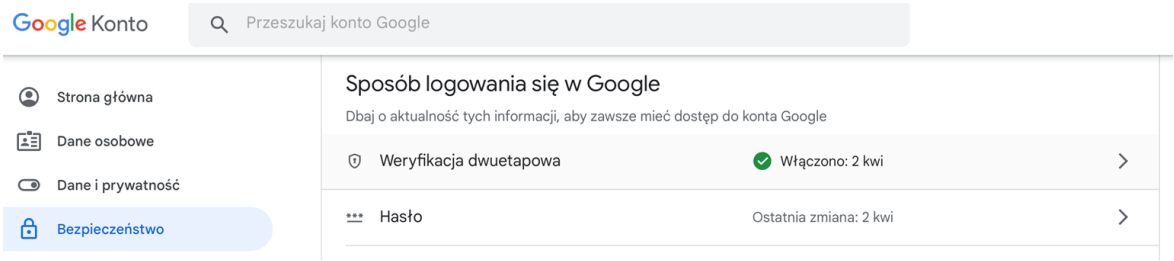
```
subject = "Email wysłany z Python'a"
body = "To jest wiadomość wysłana za pomocą SMTP"
sender = "test@gmail.com"
recipients = ["test2@gmail.com"]
password = "haslo app gmaila"
send_email(subject, body, sender, recipients, password)
```

Więcej o wysyłaniu emaili:
https://realpython.com/python-send-email/

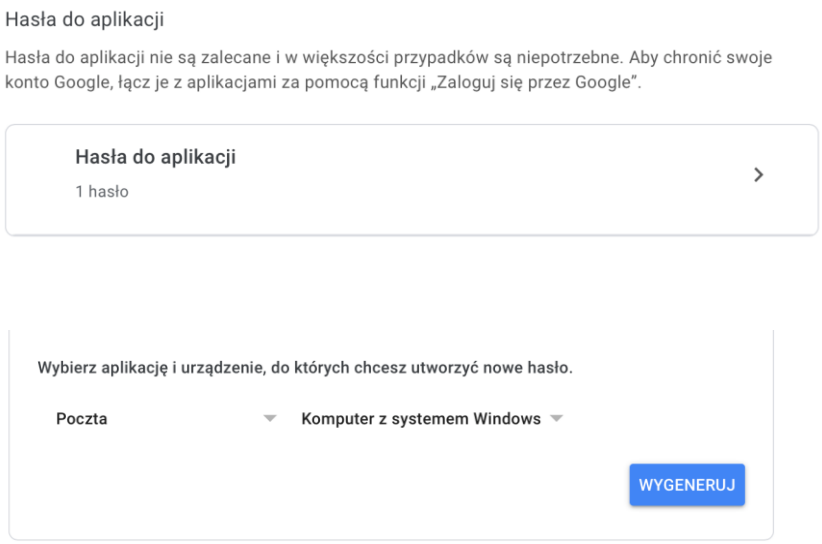
UWAGA!
Do wysyłania email najlepiej utworzyć nowe konto email – nie należy korzystać z konta do codziennego użytku.
W kodzie wysyłanym do sprawdzenia **nie podajemy** adresu emaila ani tym bardziej hasła.
Nie wysyłamy pliku z emailami.
Aby móc wysyłać maile za pomocą gmaila należy ustawić uwierzytelnianie 2 etapowe oraz utworzyć hasło do aplikacji:



Po



kliknięciu na włączoną weryfikację 2 etapową można utworzyć hasło do aplikacji:



Używamy hasła do aplikacji, nie hasła do konta email.

Zadanie

Napisz program który:

1. Wczyta z pliku dane dotyczące studentów – email, imię, nazwisko oraz liczbę uzyskanych punktów z przedmiotu Podstawy Programowania Python. Dodatkowo w pliku mogą znajdować się w tej samej linii dane dotyczące oceny końcowej oraz statusu (‘’, GRADED, MAILED). Zakładamy, że plik istnieje – może być pusty lub zawiera podstawowe informacje: email, imię, nazwisko, punkty. Do przechowywania danych w programie użyj słownika oraz zagnieżdżania (20%).

2. Umożliwi automatyczne wystawianie oceny wszystkim studentom, którzy jeszcze nie mają wystawionej oceny (status różny od GRADED oraz MAILED), zgodnie z punktacją zaproponowaną w regulaminie zaliczenia przedmiotu (20%).

3. Umożliwi usuwanie oraz dodawanie studentów ręcznie (sprawdzanie czy email jest już zajęty) (20%).

4. Umożliwi wysyłanie emaila z informacją o wystawionej ocenie wszystkim studentom ze statusem innym niż MAILED (20%).

5. Każda zmiana - dodawanie / usuwanie danych studenta, wysłanie maila, ocena – zapisze zmiany również w pliku (20%).

```
'''
50 i mniej - 2
51 -60 pkt - 3
61 – 70 pkt – 3.5
71 – 80 pkt - 4
81 - 90 pkt – 4.5
91 - 100 pkt – 5
'''
```