Moduł 6 - Obsługa plików.

1. Obsługa plików.

> Dokumentacja : https://docs.python.org/3.11/tutorial/inputoutput.html#reading-and-writing-files

Przejdźmy od razu do omówienia kilku przykładów.

Listing 1

```
# plik znajduje się w tej samej lokalizacji co uruchamiany skrypt
# lub jest to np. w PyCharm ustawiony folder roboczy, w którym ten plik się
znajduje
uchwyt = open('plik.txt')
# podana pełna ścieżka do pliku
uchwyt = open(r'C:\plik.txt', 'r')
```

Pierwsze polecenie otwiera plik, który znajduje się w folderze, w którym jest uruchamiany plik. Domyślnie plik otwierany jest tylko do odczytu. Drugie polecenie przyjmuje ścieżkę bezwzględną i dodatkowo kolejny parametr przekazuje tryb odczytu pliku, który tutaj również jest tylko do odczytu. Litera `r` poprzedzająca ścieżkę powoduje potraktowanie tego ciągu tekstowego jako ciągu surowego (ang. raw), czyli nie będą brane pod uwagę ewentualne wystąpienia znaków specjalnych, które trzeba by poprzedzać znakiem „\" bez użycia litery `r`.

Podstawowy odczyt danych z pliku można wykonać tak:

Listing 2

```
uchwyt = open('plik.txt')
uchwyt = open(r'C:\plik.txt', 'r')
dane = uchwyt.read()
print(dane)
uchwyt.close()
```

I tutaj możemy zauważyć pierwszy problem, jeżeli w pliku tekstowym znajdowały się polskie ogonki. Możemy temu zaradzić dodając dodatkowy parametr określający jak powinny być kodowane odczytywane znaki. Pamiętajmy również o zamykaniu uchwytu do pliku po odczytaniu danych.

Listing 3

```
uchwyt = open(r'C:\plik.txt', 'r', , encoding='utf-8'))
```

Typy i nazwy kodowania można znaleźć pod adresem https://docs.python.org/3.11/library/codecs.html#standard-encodings. Tryby otwarcia pliku przedstawione są

w tabelce poniżej.

Tryb	Opis
r	Tylko do odczytu. Plik musi istnieć
w	Tylko do zapisu. Jeżeli pliku nie ma to zostanie utworzony a jeżeli jest to jego zawartość zostanie zapisana nową
a	Do dopisywania. Dane dopisuje się na koniec pliku. Jeśli plik nie istnieje to zostanie utworzony
r+	Do odczytu i zapisu. Plik musi istnieć.
w+	Do odczytu i zapisu. Jeśli plik nie istnieje zostanie utworzony
a+	Do odczytu i zapisu. Jeżeli plik nie istnieje zostanie utworzony.

Możemy również odczytywać plik linia po linii z pomocą pętli:

Listing 4

```
uchwyt = open('plik.txt', 'r', encoding='utf-8')

for linia in uchwyt:
    print(linia)
uchwyt.close()
```

W tym przypadku może pojawić się sytuacja, gdzie po każdej wyświetlonej linii na wyjściu będzie wypisywana nowa linia. To dlatego, że funkcja print dodaje na końcu znak `\n`, który oznacza nową linię, a jeżeli taki znak został również odczytany z pliku to mamy odpowiedź dlaczego tak się dzieje.

Aby to zmienić można ustalić wartość parametru 'end' funkcji print na inną niż domyślna wartość.

Możemy również określić jakiej wielkości fragmenty pliku wyrażone w bajtach. Tym razem z pomocą pętli while:

Listing 5

```
uchwyt = open('plik.txt', 'r', encoding='utf-8')

while True:
    dane = uchwyt.read(1024)
    print(dane, end='')
    if not dane:
        uchwyt.close()
        break
```

Teraz kolej na zapisywanie do pliku.

Listing 6

```
uchwyt = open('plik2.txt', 'w', encoding='utf-8')
uchwyt.write('Zapisuję do pliku.')
uchwyt.close()
```

Istnieje bardziej nowoczesna metoda dostępu do plików, której wykorzystanie zwalnia nas z obowiązku pamiętania o zamknięciu uchwytu do pliku. Ta metoda gwarantuje również zamknięcie uchwytu przy wystąpieniu wyjątku. To zalecane rozwiązanie przy obsłudze plików w Python 3.

Listing 7

```
with open('plik.txt', 'r', encoding='utf-8') as file_reader:
    for linia in file_reader:
        print(linia, end='')
```

Na koniec jeszcze przykład z obsługą wyjątków:

Listing 8

```
try:
    with open('plik.txt', 'r', encoding='utf-8') as file_reader:
        for linia in file_reader:
            print(linia, end='')
except OSError as er:
    print(f'Wystąpił wyjątek OSError: {er}')
```

Więcej informacji o wyjątkach, również związanych z obsługą plików znajdziemy pod adresem https://docs.python.org/3.11/library/exceptions.html#exception-hierarchy.

2. Moduł `json` oraz `csv`.

Obsługa nieustrukturyzowanych plików nie jest najwygodniejszym sposobem utrwalania danych. Jednym z najbardziej popularnych formatów przechowywania struktur danych jest format `json` (ang. JavaScript Object Notation), który pozwala na przechowywanie informacji o całych hierarchieach oiektów. Proces zapisywania nazywa się `serializacją` a odtwarzania struktury z pliku `deserializacją`.

Listing 9

```
import json
import random

# lista
lista = [[random.randint(1, 100) for n in range(10)] for k in range(10)]

with open('dane_lista.json', 'w') as plik:
    json.dump(lista, plik)
```

```
with open('dane_lista.json', 'r') as plik:
    x = json.load(plik)

print(x)
print(type(x))

# słownik
slownik = {f'student_{x}': x**2 for x in range(1,11)}

with open('dane_slownik.json', 'w') as plik:
    json.dump(slownik, plik)

with open('dane_slownik.json', 'r') as plik:
    x = json.load(plik)

print(x)
print(type(x))

# typy inne niż list i dict nie działają out of the box
# ale można poszukać metod, które pozwalają zapisać inne struktury w
# postaci list lub słowników

class Pracownik:
    pass

p = Pracownik()
p.imie = 'Adam'
p.nazwisko = 'Malinowski'

# print(json.dumps(p)) # nie

# tak !
print(json.dumps(p.__dict__))
```

Format `csv` jest również bardzo popularny i mimo, że można go bez większych problemów obsługiwać za pomocą wbudowanych metod operujących na plikach oraz dzięki metodom `split()` oraz `join()` klasy `str` w dość efektywnie tworzyć i odtwarzać takie pliki mamy do dyspozycji moduł `csv`.

> Dokumentacja: https://docs.python.org/3.11/library/csv.html

Listing 10

```
import csv


lista = [[random.randint(1, 100) for n in range(10)] for k in range(10)]


# tworzymy plik csv korzystając z metody writerows, która przyjmuje iterowalny
# obiekt jako argument
```

```
with open('dane_lista.csv', 'w', newline='') as f:
    writer = csv.writer(f)
    writer.writerows(lista)

# otwieramy zapisany wcześniej plik linia po linii
with open('dane_lista.csv', newline='') as f:
    reader = csv.reader(f)
    for wiersz in reader:
        print(wiersz)

# parametry pliku csv można dostosować
with open('dane_lista.csv', newline='') as f:
    reader = csv.reader(f, delimiter=':', quoting=csv.QUOTE_NONE)
    for wiersz in reader:
        print(wiersz)

# przykład wykorzystania DictReader oraz DictWriter

# pierwszy wiersz w pliku traktowany jako lista kluczy słownika
# którym jest każdy zwracany wiersz
with open('dane.csv', newline='', encoding='utf-8') as csvfile:
    reader = csv.DictReader(csvfile, delimiter=';')
    for wiersz in reader:
        print(wiersz['Kraj'], wiersz['2006'])


# zapis
with open('dane_2.csv', 'w', newline='') as csvfile:
    kolumny = ['Kraj', '2020']
    writer = csv.DictWriter(csvfile, fieldnames=kolumny)

    writer.writeheader()
    writer.writerow({'Kraj': 'Polska', '2020': 37987654})
    writer.writerow({'Kraj': 'USA', '2020': 331002651})
```

Zadania

Zadanie 1

Wczytaj plik `zamowienia.csv` i dokonaj w nim kilku przekształceń:

- pozbądź się znaku `z` (właściwie zł) z kolumny `Utarg`
- zamień separator wartości dziesiętnej w tej samej kolumnie na `'.'`
- pozbądź się spacji jako separatora tysięcy w kolumnie `Utarg`
- zamień format daty w pliku na RRRR-MM-DD

Podziel plik na dwie części i zapisz je tak, aby dane dla każdego kraju (Polska, Niemcy) znajdowały się w oddzielnych plikach o nazwach zamowienia_polska.csv i zamowienia_niemcy.csv.

Zadanie 2

Napisz funkcję, która przyjmuje listę plików oraz nazwę nowego pliku jako argumenty wejściowe. Funkcja scala zawartość plików w jeden plik wynikowy o podanej wcześniej nazwie.

Zadania powtórzeniowe

Zadanie 3

Napisz funkcję, która będzie zwracała 3 największe wartości z listy numerycznej. Lista jest parametrem wejściowym funkcji.

Zadanie 4

Mając listę `mieszana = [1, 2.3, ‘Zbyszek’, 5, ‘Marian’, 3.0]` napisz funkcję, która zapisze wartości dla każdego typu do słownika gdzie pod kluczem równym nazwie typu zmiennej (int, float, str, itp.) wartością będzie lista elementów z listy 'mieszana' danego typu. Przykład: `{'int': [1,5], 'str': ['Zbyszek','Marian']}` itd.

Zadanie 5

Napisz funkcję:

- parametr wejściowy to lista stringów z nazwiskami
- funkcja zapisuje do dwóch oddzielnych plików o nazwach ‘A-M_nazwiska.txt’ oraz ‘N-Ż_nazwiska.txt’ odpowiednie nazwiska z podanej listy

Zadanie 6

Napisz funkcję, która wyświetla podany tekst odwracając kolejność liter w wyrazach. Np. „Ala ma kota” wyświetli „alA am atok”

Zadanie 7

Napisz funkcję, która:

- jako parametr wejściowy pobiera zdanie wpisywane z klawiatury,
- ponownie z klawiatury pobiera nazwę pliku, w którym zapisany zostanie wynik działania funkcji,
- do pliku zapisywane są unikalne wyrazy ze zdania pisane małymi literami po jednym w linii,
- ze zdania zostaną usunięte ewentualne przecinki i kropki.

Zadanie 8

Napisz funkcję, która losuje 5 liczb całkowitych z przedziału <1, 20> dopóki w jednym losowaniu nie wystąpi 1 i 20. Wyświetl ilość wykonanych losowań po spełnieniu warunku.

Zadanie 9

Napisz funkcję, która posiada zaszytą listę 3 nagród `['samochód', '10000 PLN', 'PS 4 Pro']`. Przygotuj plik z 10 imionami i nazwiskami zapisanymi po 1 w wierszu. Następnie funkcja wczytuje plik, losuje zwycięzcę dla każdej z trzech nagród i zapisuje wyniki w pliku o nazwie zwycięzcy.txt wpis postaci: Imię nazwisko, nagroda.

Zadanie 10

Napisz funkcję, która:

- „rozdaje” karty z talii 52 kart (np. As pik, Dama karo, itd.) dla 4 graczy
- karty rozdawane są bez powtórzeń po 5 dla każdego gracza
- wyświetlana jest informacja jak wygląda „ręka” każdego z graczy, np. Alan [As pik, 9 karo, itd.]

Zadanie 11

Napisz funkcję, która wczytuje z pliku zawierającego imiona i nazwiska osób zapisane po jednym w linii, np.

Marek Markowski
Paweł Budzikowski

Funkcja generuje dla podanej listy adresy e-mail postaci: imie.nazwisko@imgw.pl i zapisuje dane do nowego pliku dopisując przy wcześniejszym nazwisku wygenerowany adres, np.

Marek Markowski, marek.markowski@imgw.pl itd.

W nazwach e-mail powinny być zastępowane ewentualne polskie znaki (ż,ź na z, ą na a itd.).

Zadanie 12

Przygotuj plik z kilkoma hasłami, które nadają się do gry 'Koło fortuny'. Wczytaj te hasła do listy i losuj jedno z nich. Na ekranie wyświetlaj hasło bez ujawniania poszczególnych liter, np.:___ _____ ___ __ _____ dla hasła 'Bez pracy nie ma kołaczy'. Nastepnie w pętli pozwalaj uzytkownikowi na podawanie jednej litery, która będzie podstawiana w miejscach gdzie występuje lub wyświetlaj komunikat, że takiej litery nie ma w haśle. Dodaj też funkcjonalność, która pozwala na odgadnięcie (wpisanie) całego hasła.

Zadanie 13

Napisz funkcję, która będzie pobierała dwa parametry wejściowe:

- łańcuch znaków
- liczbę całkowitą

Funkcja ma skracać podany łańcuch znaków tak, żeby po dodaniu `...` na końcu jego długość nie była większa niż max (podany jako drugi parametr) i aby tekst nie był urwany w połowie wyrazu. Funkcja zwraca skrócony tekst.