Python

Poziom średniozaawansowany

Plan na dzisiaj

1. Praca z plikami tekstowymi
2. Format csv
3. Format json
4. Strumienie - pickle

OPERACJE NA PLIKACH

Operacje na plikach

- Aby móc operować na pliku trzeba najpierw stworzyć do niego referencję. Służy do tego wbudowana funkcja **open**.
- **open** wymaga jednego parametru - jest nim ścieżka do pliku, który chcemy otworzyć. Ścieżka może być względna lub bezwzględna.
- Domyślnie plik otwarty jest w trybie tylko do odczytu i w formacie tekstowym. Jeśli chcemy operować na pliku binarnym albo zapisywać lub dopisywać do pliku, należy podać tryb odczytu jako drugi parametr.

```
In [1]: f = open('file.txt')

In [2]: f.write('ala ma kota')
---------------------------------------------------------------------------
UnsupportedOperation                                 Traceback (most recent call last)
<ipython-input-2-ff2e6ea74f55> in <module>
----> 1 f.write('ala ma kota')

UnsupportedOperation: not writable

In [3]: f.close()
```

Operacje na plikach - tryby odczytu, kodowanie

- Funkcja open posiada następujące tryby:
  - **r** - tylko do odczytu (domyślny)
  - **w** - do zapisu, jeśli plik nie istnieje to zostanie utworzony, jeśli istnieje jego stara zawartość zostanie usunięta.
  - **x** - do tworzenia, operacja nie powiedzie się, jeśli plik już istnieje.
  - **a** - dopisywanie, jeśli plik nie istnieje to zostanie utworzony, jeśli istnieje to nowa treść zostanie dopisana
  - **t** - tryb tekstowy (domyślny)
  - **b** - tryb binarny
  - **+** - oznacza, że po otwarciu kursor ustawi się na końcu pliku. Domyślnie ustawia się na początku.
- Tryby można stosować łącznie, np **a+b** oznacza otwarcie pliku binarnego do zapisu, przy czym plik, jeśli istnieje, nie zostanie skrócony a dopisywanie będzie się odbywać na końcu pliku.
- Ostatnim parameterem funkcji **open** jest **encoding**, czyli kodowanie plików. Domyślna wartość zależy od systemu operacyjnego, w systemie Windows jest to **cp1252** a pod Linuxem **utf-8**.

Operacje na plikach - zarządzanie zasobami

- Po skończeniu pracy na pliku należy go zamknąć. Służy do tego metoda close na obiekcie typu File.
- Warto zauważyć, że poniższy kod nie jest bezpieczny:

```
In [1]: f = open('file.txt')

In [2]: # do stuff ...

In [3]: f.close()
```

- Nie ma żadnej gwarancji, że pomiędzy otwarciem a zamknięciem pliku nie wystąpi jakiś wyjątek, w konsekwencji którego nie wykona się ostatnia linijka i zostaniemy z otwartym plikiem.
- Lepszym podejściem byłoby otworzenie pliku w bloku try i zamknięcie w bloku finally. Mamy wtedy pewność, że niezależnie od tego co się stanie plik zostanie zamknięty.

```
In [1]: try:
   ...:     f = open('file.txt')
   ...:     # do stuff...
   ...: finally:
   ...:     f.close()
   ...:
```

Operacje na plikach - zarządzanie zasobami

- Obie metody podane poniżej nie są specjalnie “pythonowe”.
- Najlepszym sposobem na otworzenie pliku jest użycie słowa kluczowego with.
- Słowo kluczowe with otwiera kontekst, który znajduje się we wciętym bloku poniżej niego.
- Kiedy opuścimy wcięty blok kontekstu, plik zostanie automatycznie zamknięty, także nie musimy wcale zamykać go sami.
- To eliminuje pomyłki spowodowane zapomnieniem o konieczności zamknięcia pliku kiedy nie używamy managera kontekstu.

```
In [1]: with open('file.txt', 'a+') as f:
   ...:     # do stuff...
   ...: # here the file is already closed
```

Operacje na plikach - zapis

- Aby zapisać jakąś treść do pliku należy go najpierw otworzyć w odpowiednim trybie (do zapisu albo do dopisania).
- Jeśli operacja otwarcia pliku powiedzie się (mamy odpowiednie uprawnienia, na dysku jest wystarczająco dużo miejsca) to możemy zapisać do niego treść używając metody write.
- Metoda write jako parametr przyjmuje ciąg znaków, który chcemy zapisać do pliku.
- Na przykładzie obok otworzyliśmy plik o nazwie file.txt w katalogu bieżącym (podaliśmy względną ścieżkę).
- Plik był otwarty w trybie do odczytu a więc jeśli nie istniał do został utworzony a jeśli istniał to jego stara zawartość została usunięta.
- We wciętym bloku zapisaliśmy dwie linijki treści, po opuszczeniu bloku plik został automatycznie zamknięty.

```
In [1]: with open('file.txt', 'w') as f:
   ...:     f.write('Ala ma kota\n')
   ...:     f.write('Kot ma Alę\n')
   ...:

In [2]: !cat file.txt
Ala ma kota
Kot ma Alę
```

Operacje na plikach - odczyt

- Aby odczytać treść z pliku można użyć metody **read**.
- Jeśli metoda **read** zostanie wywołana bez parametrów, wczyta na raz całą zawartość pliku.
- Jeśli plik jest bardzo duży to może być potencjalnie groźna operacja dlatego opcjonalnie można podać liczbę, która oznacza maksymalną ilość bajtów, które zostaną wczytane.
- Mówmy maksymalną, ponieważ zawartość pliku może być krótsza niż ilość bajtów, które zażądaliśmy przeczytać.
- Kolejne wywołania funkcji **read** przesuwają kursor odczytu do przodu, także zaczynamy czytać od momentu, w którym skończyliśmy.
- Kiedy dotrzemy do końca pliku, metoda **read** będzie zwracać pusty string.
- Aby przesunąć kursor w zadane miejsce od początku pliku należy użyć metody **seek**.
- Aby poznać obecną pozycję kursora w pliku należy użyć metody **tell**.

```
In [1]: with open('file.txt', 'r') as f:
   ...:     content = f.read(4)
   ...:     print(f'Content I read is: {content}')
   ...:     print(f"This is where I'm in the file: {f.tell()}")
   ...:     content = f.read()
   ...:     print(f'Some more content I read: {content}')
   ...:     print(f"And now I'm here: {f.tell()}")
   ...:     content = f.read()
   ...:     print(f'Even more content I read: {content}')
   ...:     print(f"And where I am now: {f.tell()}")
   ...:     f.seek(0)
   ...:     print(f"I should be back now: {f.tell()}")
   ...:
Content I read is: Ala
This is where I'm in the file: 4
Some more content I read: ma kota
Kot ma Alę

And now I'm here: 24
Even more content I read:
And where I am now: 24
I should be back now: 0
```

Operacje na plikach - odczyt

- W przypadku plików tekstowych bardzo często chcemy czytać plik linijka po linijce a nie bajt po bajcie.
- Istnieją trzy główne sposoby czytania pliku linijka po linijce.
  - Podejście prehistoryczne: użycie metody **readline**
  - Podejście średniowieczne: użycie metody **readlines**. **readlines** zwraca listę, której każdy element jest pojedynczą linijką przeczytaną z pliku. Wciąż może być przydatna kiedy nie chcemy czytać linijka po linijce ale poruszać się po pliku w wybrany przez nas sposób. Należy jednak pamiętać, że **readlines** wczyta cały plik, niezależnie od jego wielkości.
  - Podejście standardowe: użycie leniwego iteratora zwracanego przez obiekt typy File. Sam deskryptor pliku zwraca iterator przehodzący linijka po linijce przy czym jest on lepszy od readlines ponieważ nie wczytuje na raz całego pliku. Ten sposób jest preferowany.

```
In [1]: f = open('file.txt')

In [2]: while(True):
   ...:     line = f.readline()
   ...:     if not line:
   ...:         break
   ...:     print(line)
   ...:
Ala ma kota

Kot ma Alę
```

```
In [1]: with open('file.txt') as f:
   ...:     lines = f.readlines()
   ...:     for line in lines:
   ...:         print(line)
   ...:
Ala ma kota

Kot ma Alę
```

```
In [1]: with open('file.txt') as f:
   ...:     for line in f:
   ...:         print(line)
   ...:
Ala ma kota

Kot ma Alę
```

Pliki csv

Pliki CSV

- CSV to skrót od *comma separated values* (wartości rozdzielone przecinkiem).
- Są to pliki tekstowe o specjalnej strukturze przeznaczone do przechowywania danych tabularycznych.
- Wartości oddzielone są przecinkiem (lub innym separatorem np. tabulatorem, średnikiem).
- Jest to format przechowywania danych w plikach typu text/csv (to znaczy, że jest to jakiś znany i uznawany format).

```
1 "Index", "Year", "Age", "Name", "Movie"
2  1, 1928, 44, "Emil Jannings", "The Last Command, The Way of All Flesh"
3  2, 1929, 41, "Warner Baxter", "In Old Arizona"
4  3, 1930, 62, "George Arliss", "Disraeli"
5  4, 1931, 53, "Lionel Barrymore", "A Free Soul"
6  5, 1932, 47, "Wallace Beery", "The Champ"
```

Pliki CSV

UWAGA:

- Spacje i inne białe znaki (w szczególności te przyległe do separatorów) należą do pól.
- Pierwsza linia może stanowić nagłówek zawierający nazwy pól rekordów, jednak pierwszy wiersz pliku CSV wg standardu ma takie samo znaczenie jak pozostałe.
- Zazwyczaj pierwszy wiersz określa nazwy kolumn.

```
1 "Index", "Year", "Age", "Name", "Movie"
2  1, 1928, 44, "Emil Jannings", "The Last Command, The Way of All Flesh"
3  2, 1929, 41, "Warner Baxter", "In Old Arizona"
4  3, 1930, 62, "George Arliss", "Disraeli"
5  4, 1931, 53, "Lionel Barrymore", "A Free Soul"
6  5, 1932, 47, "Wallace Beery", "The Champ"
```

Pliki CSV

- Do obsługi plików csv służy moduł csv.
- Plik otwieramy tak jak w przypadku normalnych plików tekstowych (zalecane: menadżer kontekstu with).
- Funkcja do odczytu pliku nazywa się reader i znajduje się w module csv.
- Jako argumenty podajemy jej uprzednio otwarty plik oraz znak delimitera (przecinek, tabulator, średnik).

```
In [1]: import csv

In [2]: with open("plik.csv", "r") as csvfile:
   ...:     csvreader = csv.reader(csvfile, delimiter=",")
   ...:
```

Pliki CSV

- Po wykonaniu funkcji reader zwracającej iterator, możemy po nim przeiterować i uzyskać dostęp do każdego wiersza z pliku po kolei.

```
In [11]: with open("F:\pdf\plik.csv", "r") as csvfile:
    ...:     csvreader = csv.reader(csvfile, delimiter=",")
    ...:     for row in csvreader:
    ...:         print(row)
```

```
['Index', ' "Year"', ' "Age"', ' "Name"', ' "Movie"']
[' 1', ' 1928', ' 44', ' "Emil Jannings"', ' "The Last Command', ' The Way of All Flesh"']
[' 2', ' 1929', ' 41', ' "Warner Baxter"', ' "In Old Arizona"']
[' 3', ' 1930', ' 62', ' "George Arliss"', ' "Disraeli"']
[' 4', ' 1931', ' 53', ' "Lionel Barrymore"', ' "A Free Soul"']
[' 5', ' 1932', ' 47', ' "Wallace Beery"', ' "The Champ"']
```

- Każdy wiersz zostaje zapisany jako lista elementów (na podstawie delimitera (najczęściej przecinka) z pliku; UWAGA – liczby są zapisywane jako stringi!

Pliki CSV

- Oprócz funkcji reader z modułu csv, do odczytu danych z pliku csv możemy wykorzystać klasę DictReader.

```
with open("F:\pdf\plik.csv", "r") as csvfile:
    csvreader = csv.DictReader(csvfile)
    for row in csvreader:
        print(row)
```

Pliki CSV

- Obiekt typu DictReader odczytuje zawartość pliku csv z wykorzystaniem słownika, w którym kluczami są nazwy kolumn z pliku (ustalane na podstawie wartości pierwszego wiersza w pliku).

```
"Index", "Year", "Age", "Name", "Movie"
 1, 1928, 44, "Emil Jannings", "The Last Command, The Way of All Flesh"
 2, 1929, 41, "Warner Baxter", "In Old Arizona"
 3, 1930, 62, "George Arliss", "Disraeli"
 4, 1931, 53, "Lionel Barrymore", "A Free Soul"
 5, 1932, 47, "Wallace Beery", "The Champ"
```

- Oto przykład odczytania pierwszego wiersza z pliku z wykorzystaniem klasy DictReader.

```
OrderedDict([('Index', ' 1'), (' "Year"', ' 1928'), (' "Age"', ' 44'), (' "Name"', ' "Emil Jannings"'),
(' "Movie"', ' "The Last Command'), (None, [' The Way of All Flesh"'])])
```

- OrderedDict to słownik, który pamięta kolejność par klucz: wartość, które zostały do niego dodane (poza tym zachowuje się jak zwykły dict).

Odczytaj zawartość pliku students.csv z repozytorium i znajdź ucznia oraz uczennicę o najwyższym wzroście.

Pliki CSV

- Celem zapisania danych do formatu csv, korzystamy z funkcji writer z modułu csv.

```
with open("F:\pdf\plik.csv", mode="w") as csvfile:
    writer = csv.writer(csvfile, delimiter=",", quotechar='"')
    writer.writerow([6, 1950, 44, "An actor", "Any film"])
```

- Argumenty: csvfile – nasz otwarty plik, delimiter – znak oddzielający od siebie kolumny danych, quotechar – jakimi znakami będziemy oznaczać napisy (albo ‘napis’ albo ”napis”)
- Funkcja writerow obiektu zwróconego przez funkcję csv.writer zapisuje do pliku pojedynczy wiersz: każdy element to nowa kolumna w wierszu oddzielona od innych znakiem delimitera

Pliki CSV

- Podobnie jak w przypadku odczytu i przy zapisywaniu możemy skorzystać z słownikowego zarządzania plikami csv – zapis do pliku jest możliwy z wykorzystaniem klasy DictWriter.

```
with open('F:\pdf\plik.csv', mode='w', newline="") as csv_file:
    column_names = ['id', 'name']
    writer = csv.DictWriter(csv_file, fieldnames=column_names)

    writer.writeheader()
    writer.writerow({"id": 1, "name": "Anna"})
```

- Na początku tworzymy listę z nazwami kolumn, jakie będziemy chcieli umieścić w pliku (u nas lista nazywa się column_names)
- Przekazujemy tę listę jako parametr fieldnames, po czym metodą writeheader obiektu writer zapisujemy pierwszy wiersz z nazwami kolumn do pliku csv.

Pliki CSV

- writer.writerow zapisuje każdy kolejny wiersz do pliku.
- Metoda ta wymaga podania słownika z nazwami kluczy odpowiadającymi nazwom kolumn.

```
with open('F:\pdf\plik.csv', mode='w', newline="") as csv_file:
    column_names = ['id', 'name']
    writer = csv.DictWriter(csv_file, fieldnames=column_names)

    writer.writeheader()
    writer.writerow({"id": 1, "name": "Anna"})
```

- Parametr newline w funkcji open zapobiega dodatkowemu dodaniu nowej linii po każdym nowym wierszu w pliku csv.

Pobierz zawartość pliku snake_game.csv z repozytorium za pomocą biblioteki csv. Zaproponuj i stwórz strukturę przechowującą dane na temat przebiegu gry. Dla takich danych napisz:

a) funkcję sortującą dane, powinny być one poukładane według ilości zdobytych punktów (rosnąco),
b) funkcję obliczającą średnią ilość zdobytych punktów.
c) funkcję dodającą nowy wiersz danych do pliku csv (funkcja powinna automatycznie inkrementować indeks danych)

Pliki json

Pliki JSON

- JSON (ang. *JavaScript Object Notation*) to kolejny sposób na przechowywanie i przekazywanie danych tekstowych.
- Często używany do komunikacji i przekazywania informacji podczas tworzenia i korzystania z webowych API (np. w standardzie REST).
- Ma strukturę klucz-wartość.
- Pod danym kluczem może być zapisana pojedyncza wartość albo kolekcja.

```
{
  "dogs": [{
    "id": 0,
    "name": "maja",
    "race": "pies"
  }, {
    "id": 1,
    "name": "milus",
    "race": "pies"
  }],
  "cats": [{
    "id": 0,
    "name": "puszek",
    "race": "kot"
    }, {
    "id": 1,
    "name": "greebo",
    "race": "kot"
  }]
}
```

Pliki JSON

- Aby móc przetwarzać pliki JSON musimy zaimportować moduł json.
- Używamy funkcji open do otwarcia i utworzenia dostępu do pliku *example.json*, następnie korzystamy z metody *load* z pakietu *json*, by odczytać jego zawartość i zapisać w zmiennej *data*.

```
import json

with open("example.json") as in_file:
    data = json.load(in_file)

print(data['dogs'])
```

Powyższy kod spowoduje wypisanie na ekran wartości spod klucza 'dogs' ze słownika na obrazku.

```
[{'id': 0, 'name': 'maja', 'race': 'pies'}, {'id': 1, 'name': 'milus', 'race': 'pies'}]
```

Pliki JSON

- Tak jak poprzednio, używamy funkcji open do otwarcia i utworzenia dostępu do pliku *pracownicy.csv* z flagą 'w', dzięki czemu plik zostanie utworzony i będziemy mogli do niego zapisywać dane.

- Korzystamy z metody *dump* z pakietu *json*, by zapisać listę *kursanci* do pliku out_file.

- Parametr *indent* określa jak duże wcięcia (liczone w liczbie spacji) mają być na początku każdego kolejnego poziomu pliku.

```
import json

kursanci = [
    {
        'imie': "Adam",
        'nazwisko': "Kowalski",
        'liczba punktow': 20
    },
    {
        'imie': "Janusz",
        'nazwisko': "Smuga",
        'liczba punktow': 17
    }
]

with open("kursanci.json", 'w') as out_file:
    json.dump(kursanci, out_file, indent=2)
```

Odczytaj raz jeszcze plik students.csv. Wygeneruj z danych z pliku listę słowników – każdy słownik powinien mieć takie same klucze: name, gender, height, weight. Zapisz taką listę do pliku json.

PICKLE

pickle

- Pickle to biblioteka używana do serializacji i deserializacji obiektów pythonowych,
- Do tej pory zapisywaliśmy do pliku dane tekstowe, okazuje się, że to nie jedyne co potrafi Python,
- Każdy **obiekt** może zostać zserializowany i zapisany na dysku.
- Bibliteka pickle potrafi przekonwertować obiekt (list, set, dict itd.) do strumienia znaków.
- Taki strumień posiada wszystko, co jest potrzebne, do odtworzenia takiego obiektu w dogodnym czasie.

pickle

- Do zserializowania obiektu, wykorzystujemy funkcję **`dump`** z bilbioteki **`pickle`**.
- Funkcja ta przyjmuje dwa argumenty: obiekt, który chcemy zapisać do pliku oraz reprezentację tego pliku.
- Przy otwieraniu pliku w trybie do zapisu należy pamiętać, że otwierany jest on w trybie binarnym(!), stąd flaga trybu powinna być ustawiona na ‘wb’

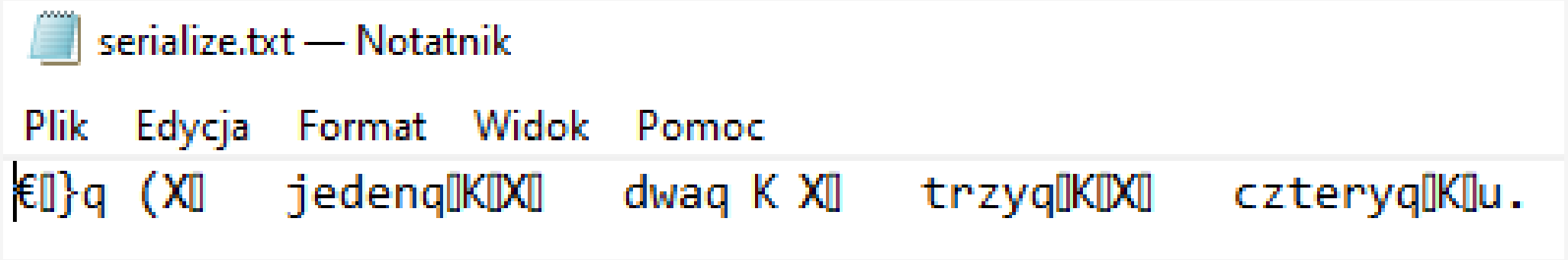
```
In [1]: import pickle

In [2]: testowy_slownik = {
   ...:     "jeden": 1,
   ...:     "dwa": 2,
   ...:     "trzy": 3,
   ...:     "cztery": 4
   ...: }

In [3]: with open("F:\pdf\serialize.txt", "wb") as binary_file:
   ...:     pickle.dump(testowy_slownik, binary_file)
   ...:
```

pickle

- Po wykonaniu funkcji pickle.dump do pliku trafia zserializowany obiekt w postaci strumienia znaków.
- Charakterystyczne znaczki (krzaczki) świadczą o zawartości binarnej pliku.
- Od tego momentu obiekt wraz z jego stanem i zawartością znajduje się bezpiecznie w pliku. Kiedy nadejdzie potrzeba, będzie go można odtworzyć.



pickle

- Do odtworzenia zserializowanego wcześniej obiektu, służy funkcja **`load`** z biblioteki **`pickle`**.
- Funkcja ta wymaga jednego argumentu: otworzonego wcześniej pliku, w którym znajduje się zapisany obiekt. Wydobyty obiekt można przypisać do nowej zmiennej.
- Należy pamiętać o otwarciu pliku w trybie binarnym! Podajemy tryb ‘rb’.

```
In [4]: pusty_slownik = {}

In [5]: with open("F:\pdf\serialize.txt", "rb") as binary_file:
   ...:     pusty_slownik = pickle.load(binary_file)
   ...:

In [6]: pusty_slownik
Out[6]: {'jeden': 1, 'dwa': 2, 'trzy': 3, 'cztery': 4}
```

pickle

- Dzięki pickle staje się możliwa wymiana obiektów w Pythonie między różnymi maszynami wirtualnymi, środowiskami, a nawet komputerami oddalonymi od siebie o wiele kilometrów.
- Zapisany obiekt zostaje zachowany i jest gotowy do odtworzenia nawet po ponownym włączeniu komputera.
- Zserializowane obiekty można umieścić w bazach danych.
- Należy pamiętać, że korzystanie z pickle i deserializacji obiektów nie jest w pełni bezpieczne! „Wczytując” z pliku obiekt nieznanego pochodzenia narażamy się na kłopoty – obiekt może działać jak mały wirus.

Odczytaj plik json utworzony z danych pobranych ze students.csv. Po pobraniu danych – zapisz je do pliku pickle w formacie binarnym.