OBSŁUGA I PRZETWARZANIE RÓŻNYCH TYPÓW DANYCH, UŻYCIE WYRAŻEŃ REGULARNYCH

ĆWICZENIA DO PREZENTACJI

MODUŁ 9

Altkom Akademia S.A., materiały własne

1 PRZETWARZANIE DANYCH

ĆWICZENIE 1.1:

Praca z plikami CSV

UMIEJĘTNOŚCI:

- Po wykonaniu ćwiczenia zdobędziesz umiejętność:
  - konwersji list do formatu CSV i na odwrót
  - konwersji słowników do formatu CSV i na odwrót

CELE I ZADANIA:

- Utwórz funkcję umożliwiającą zapis danych z listy list do pliku w formacie CSV oraz funkcję pobierającą dane z pliku w formacie CSV i odtwarzającą sekwencję list
- Utwórz podobne funkcje konwersji danych pomiędzy listą słowników, a formatem CSV
- Utwórz przykładowe dane i przetestuj działanie utworzonych funkcji

ALGORYTM WYKONANIA:

- Utwórz funkcję o nazwie *export_lists_to_csv*, z następującymi parametrami:
  - nazwa pliku CSV
  - dane do zapisu w pliku (lista list)
  - lista nazw nagłówków (nazw kolumn)
- Zadaniem funkcji jest zapis danych do pliku w formacie CSV
- Każdej liście z danymi powinna odpowiadać jedna linia w pliku
- Elementy listy należy odseparować symbolem przecinka
- Pierwsza linia pliku powinna zawierać nazwy nagłówków (kolumn) również odseparowane przecinkami
- Następnie utwórz funkcję o nazwie *import_lists_from_csv* z parametrem określającym nazwę pliku CSV z danymi
- Zadaniem funkcji jest odtworzenie listy list z danymi na podstawie zawartości pliku CSV
- Utwórz zestaw przykładowych danych, np. listę z danymi poszczególnych uczelni (każda uczelnia będzie opisana przez: nazwę i adres, czyli ulicę, kod pocztowy i miejscowość)
- Przetestuj poprawność działania obu funkcji

- Postępując podobnie, utwórz i przetestuj funkcje dokonujące zapisu danych z listy słowników do pliku CSV oraz funkcję odtwarzającą dane z pliku

ĆWICZENIE 1.2:

Parsowanie plików XML

UMIEJĘTNOŚCI:

- Po wykonaniu ćwiczenia zdobędziesz umiejętność parsowania danych XML z użyciem róznych metodyk

CELE I ZADANIA:

- Utwórz plik z przykładowymi danymi w formacie XML
- Przeparsuj zawartość pliku i utwórz listę słowników, odpowiadającą danym w dokumencie XML

ALGORYTM WYKONANIA:

- Utwórz plik XML z przykładowymi danymi, np.:

```
<?xml version="1.0" encoding="utf-8" ?>
<products>
    <product product-id="PN12345">
        <name>klin</name>
        <quantity>10</quantity>
    </product>
    <product product-id="PN54321">
        <name>cokół</name>
        <quantity>7</quantity>
    </product>
</products>
```

- Wykorzystaj moduł *xml.etree.ElementTree* do przeparsowania danych
- Każdy towar (element *product*) powinien być reprezentowany przez osobny słownik
- Role kluczy słownika powinny pełnić nazwy elementów-dzieci oraz nazwy atrybutów
- Słowniki należy zgrupować w listę

- Spróbuj osiągnąć ten sam cel wykorzystując metodykę SAX
- W tym celu należy utworzyć klasę dziedziczącą po *ContentHandler* i zaimplementować wybrane metody – w naszym przypadku: *startElement*, *characters* oraz *endElement*
- Przetestuj poprawność działania obu metodyk

ĆWICZENIE 1.3:

Tabela kursów walut

UMIEJĘTNOŚCI:

- Po wykonaniu ćwiczenia zdobędziesz umiejętność parsowania danych i wydobywania potrzebnych danych ze strony HTML przy użyciu biblioteki *BeautifulSoup*

CELE I ZADANIA:

- Pobierz ze strony HTML spod adresu: *https://www.nbp.pl/home.aspx?f=/kursy/kursya.html* tabelę kursów walut
- Utwórz listę list – każda lista wewnętrzna powinna zawierać dane dotyczące kursu jednej waluty, tzn.: nazwę waluty, jej kod i kurs średni
- Pamiętaj także o pobraniu ze strony nagłówka tabeli i jej nazwy

ALGORYTM WYKONANIA:

- Pobierz ze strony HTML spod adresu: *https://www.nbp.pl/home.aspx?f=/kursy/kursya.html* tabelę kursów walut
- Do tego celu możesz wykorzystac następujący kod:

```
import urllib.request

response = urllib.request.urlopen('https://www.nbp.pl/home.aspx?f=/kursy/kursya.html')
html_page = response.read()
```

- Podejrzyj kod źródłowy strony HTML i zapoznaj się z jej strukturą
- Zastanów się, w jaki sposób można dotrzeć do żądanych danych
- Utwórz obiekt typu *BeautifulSoup* wykorzystujący parser HTML
- Wyszukaj tytuł tabeli zawarty wewnątrz znacznika HTML $< h3 >$
- Następnie wyszukaj najbliższą występującą dalej tabelę
- Pobierz tytuły kolumn tabeli, a następnie w pętli dane tabeli i wstaw je do listy
- Na koniec wypisz zawartość listy i porównaj ją z treścią na stronie

2 SERIALIZACJA I DESERIALIZACJA

ĆWICZENIE 2.1:

Serializacja i deserializacja binarna

UMIEJĘTNOŚCI:

- Po wykonaniu ćwiczenia zdobędziesz umiejętność:
  - serializacji binarnej obiektu
  - odtworzenia obiektu (deserializacji) na podstawie jego reprezentacji binarnej

CELE I ZADANIA:

- Zdefiniuj przykładowe dane, które będziemy serializować, np. klasę kontaktu zawierającą dane personalne (imię, nazwisko, lista telefonów) oraz klasę telefonu zawierającą rodzaj kontaktu telefonicznego i numer
- Wykorzystując bibliotekę *pickle* dokonaj serializacji obiektu kontaktu do postaci binarnej, a następnie odtwórz ten obiekt za pomocą deserializacji
- Porównaj dane przed serializacją i po deserializacji

ALGORYTM WYKONANIA:

- Utwórz dane, które będziemy serializować – te dane będą wykorzystywane w kolejnych ćwiczeniach. W tym celu utwórz:
  - klasę o nazwie *Phone*, która będzie posiadała dwa atrybuty instancyjne reprezentujące typ telefonu (np. prywatny/służbowy albo stacjonarny/komórkowy) oraz numer telefonu
  - klasę o nazwie *Contact*, która będzie posiadała atrybuty instancyjne: imię, nazwisko i lista telefonów
- W obu klasach zdefiniuj metody konwersji na postać tekstową (*__str__* oraz *__repr__*)
- Utwórz instancję kontaktu zawierającą kilka telefonów
- Obie klasy oraz instancje umieść w osobnym module – dzięki temu bedzie można je wykorzystać w kolejnych ćwiczeniach
- Wykorzystując moduł *pickle* napisz funkcje, które umożliwią:
  - serializację binarną obiektu kontaktu do sekwencji bajtów
  - deserializację obiektu kontaktu z sekwencji bajtów
- Przetestuj działanie obu funkcji i sprawdź, czy udało się odtworzyć początkowy obiekt

ĆWICZENIE 2.2:

Serializacja i deserializacja do/z JSON

UMIEJĘTNOŚCI:

- Po wykonaniu ćwiczenia zdobędziesz umiejętność:
  - serializacji obiektu do formatu JSON
  - odtworzenia obiektu (deserializacji) na podstawie danych w formacie JSON

CELE I ZADANIA:

- Zdefiniuj przykładowe dane – możesz wykorzystać obiekty z poprzedniego ćwiczenia
- Wykorzystując bibliotekę *json* dokonaj serializacji obiektu kontaktu do formatu JSON, a następnie odtwórz ten obiekt za pomocą deserializacji
- Porównaj dane przed serializacją i po deserializacji

ALGORYTM WYKONANIA:

- Do serializacji obiektów do formatu JSON wykorzystaj obiekty utworzone w poprzednim ćwiczeniu (wykorzystaj instrukcję importu)
- Wykorzystując moduł *json* napisz funkcje, które umożliwią:
  - serializację obiektu kontaktu do formatu JSON
  - deserializację obiektu kontaktu z formatu JSON
- Ponieważ serializujemy instancje własnej klasy należy zadbać o konwersję obiektu do struktur Pythona (list i słowników)
- Zastanów się, jak tego dokonać
- Podobnie podczas deserializacji otrzymamy dane obiektów w postaci list i słowników
- Jak w prosty sposób można na podstawie tych danych odtworzyć instancje klas?
- Przetestuj działanie obu funkcji i sprawdź, czy udało się odtworzyć początkowy obiekt

ĆWICZENIE 2.3:

Serializacja i deserializacja do/z YAML

UMIEJĘTNOŚCI:

- Po wykonaniu ćwiczenia zdobędziesz umiejętność:
  - serializacji obiektu do formatu YAML
  - odtworzenia obiektu (deserializacji) na podstawie danych w formacie YAML

CELE I ZADANIA:

- Zdefiniuj przykładowe dane – możesz wykorzystać obiekty z poprzedniego ćwiczenia
- Wykorzystując bibliotekę *yaml* dokonaj serializacji obiektu kontaktu do formatu YAML, a następnie odtwórz ten obiekt za pomocą deserializacji
- Porównaj dane przed serializacją i po deserializacji

ALGORYTM WYKONANIA:

- Do serializacji obiektów do formatu YAML wykorzystaj obiekty utworzone w poprzednim ćwiczeniu (wykorzystaj instrukcję importu)
- Wykorzystując moduł *yaml* napisz funkcje, które umożliwią:
  - serializację obiektu kontaktu do formatu YAML
  - deserializację obiektu kontaktu z formatu YAML
- Przetestuj działanie obu funkcji i sprawdź, czy udało się odtworzyć początkowy obiekt

3 WYRAŻENIA REGULARNE

ĆWICZENIE 3.1:

Budowa wyrażeń regularnych

UMIEJĘTNOŚCI:

- Po wykonaniu ćwiczenia zdobędziesz umiejętność tworzenia i wykorzystania wyrażeń regularnych

CELE I ZADANIA:

- Zdefiniuj przykładowe dane
- Utwórz kilka wyrażeń regularnych i zweryfikuj, czy dane faktycznie pasują do definicji szablonu
- Wykorzystaj narzędzia on-line dostępne w sieci

ALGORYTM WYKONANIA:

- Uruchom narzędzie on-line umożliwiające ewaluację wyrażeń regularnych – możesz użyć: https://pythex.org/
- Napisz wyrażenia regularne, które:
  a. sprawdzi, czy dany tekst kończy się np. na '*foo*'
  b. sprawdzi, czy dany tekst rozpoczyna się np. na '*foo*'
  c. sprawdzi, czy tekst reprezentuje poprawny numer telefonu komórkowego w Polsce: '*+48xxxxxxxxx*'
  d. sprawdzi, czy tekst jest prostym adresem e-mail o postaci: '*xxx@xxx.xxx*'
  e. sprawdzi, czy tekst jest 4- lub 5-znakowym palindromem
  f. sprawdzi, czy ktoś się jąka – w tekście powtórzone są co najmniej 2 litery
  g. wyszuka wszystkie tagi w następującym XML'u: '*<tag1> <tag2> ala </tag2></tag1>*'
  h. sprawdzi poprawność następującego XML'a: '*<xxx>aaa</yyy>*'
  i. wyszuka w tekstach o postaci '*Jan Xxxxx*' wszystkich Janów o nazwiskach kończących się na '*-ski*', ale nie *Kowalskich*
  j. wyszuka spośród tekstów o postaci: '*PESEL NAZWISKO*' tylko:
     * kobiety z nazwiskiem kończącym się na '*-ska*' oraz
     * mężczyzn z nazwiskiem kończącym się na '*-ski*'

ĆWICZENIE 3.2:

Użycie wyrażeń regularnych w Pythonie

UMIEJĘTNOŚCI:

- Po wykonaniu ćwiczenia zdobędziesz umiejętność tworzenia i wykorzystania wyrażeń regularnych

CELE I ZADANIA:

- Zdefiniuj przykładowe dane
- Utwórz kilka wyrażeń regularnych i zweryfikuj, czy dane faktycznie pasują do definicji szablonu
- Wykorzystaj funkcje modułu *re*

ALGORYTM WYKONANIA:

- Wykonaj te same wyszukania i dopasowania, co w poprzednim ćwiczeniu, ale tym razem wykorzystaj możliwości pakietu *re*, a nie narzędzi on-line