INFORMATYKA II

ANALIZA DANYCH W JĘZYKU PYTHON

CEL LABORATORIUM

W trakcie zajęć zaprezentowane zostaną podstawy operacji na plikach w formacie JSON i CSV w języku Python. Dane zapisane w pliku JSON zostaną poddane analizie przeglądowej, w tym ekstrakcji informacji takich jak średnia, mediana, wartość minimalna, maksymalna, wariancja, odchylenie standardowe czy IQR.

MATERIAŁY POMOCNICZE

Wykorzystywany w trakcie zajęć plik JSON można znaleźć pod następującym adresem: http://www.stawarz.edu.pl/informatyka2/katalog.json.

Wykorzystywany w trakcie zajęć plik CSV można znaleźć pod następującym adresem: http://www.stawarz.edu.pl/informatyka2/zakupione.csv.

Dokumentacja pakietu Numpy znajduje się pod adresem: https://numpy.org/doc/.

Specyfikację formatu JSON można znaleźć na stronie https://www.json.org/json-pl.html.

Format CSV szczegółowo opisuje dokument RFC 4180, w pełni dostępny pod adresem https://tools.ietf.org/html/rfc4180.

Baza danych może zostać uruchomiona na każdym komputerze, który jest zaopatrzony w system zarządzania bazami danych MySQL. Darmowym programem pozwalającym pracować z bazą danych w zaciszu domowym jest pakiet XAMPP. O tym skąd go pobrać, jak zainstalować i wykorzystać, dowiedzieć się można z materiałów pomocniczych udostępnionych pod adresem http://www.stawarz.edu.pl/informatyka2/dodatek_xampp.pdf.

STRUKTURA DANYCH

Pierwszym krokiem pracy z danymi powinno być poznanie ich struktury. Na poprzednich zajęciach, gdy praca odbywała się z bazą danych MySQL, jej struktura była przedstawiona na załączonym do instrukcji diagramie i powinna być już znana. Dodatkowe dane, które będą wykorzystywane na aktualnych zajęciach, mają odmienną strukturę.

Dane w pliku JSON przyjmują strukturę zagnieżdżonej, wielowymiarowej listy asocjacyjnej. „Na górze", w wymiarze nadrzędnym, lista zawiera wyłącznie dwa wpisy, o kluczach „lampy" i „telewizory". Wartość każdego wpisu to odpowiednio lista zawierająca informacje o lampach i telewizorach. Każdy rodzaj produktu posiada inne cechy. Zostały one przedstawione w poniższej tabeli. Plik JSON można otworzyć w dowolnym edytorze tekstowym (jak np. notatniku).

DANE ZAPISANE W PLIKU JSON			
lampy		telewizory	
nazwa	Ciąg znaków	nazwa	Ciąg znaków
producent	Ciąg znaków	producent	Ciąg znaków
cena	Liczba zmiennoprzecinkowa	cena	Liczba zmiennoprzecinkowa
kod	Ciąg znaków	kod	Ciąg znaków
typ	Ciąg znaków	przekątna	Liczba całkowita
oprawka	Ciąg znaków	matryca	Ciąg znaków
ilość punktów światła	Liczba całkowita	format	Ciąg znaków
strumień świetlny	Liczba całkowita	technologia 3D	Ciąg znaków
efektywność	Ciąg znaków	VESA	Ciąg znaków
materiał	Ciąg znaków	złącza	Liczba całkowita
kolor	Ciąg znaków	smartTV	Ciąg znaków
styl	Ciąg znaków	efektywność	Ciąg znaków
ściemnianie	Ciąg znaków	kolor	Ciąg znaków

Dane w pliku CSV przyjmują strukturę tablicy dwuwymiarowej. Pierwsza kolumna zawiera id urządzenia, odpowiadający polu devices.id w bazie danych MySQL. Druga kolumna zawiera kod produktu, odpowiadający kodowi produktu w pliku JSON. Pierwszy wiersz jest nagłówkiem. Tak jak plik formatu JSON, tak i plik zapisany w formacie CSV można otworzyć zwykłym edytorem tekstowym.

MODUŁ KATALOGU PRODUKTÓW

Po uruchomieniu projektu z poprzednich zajęć, należy utworzyć nowy plik o nazwie „*modułKatalogu.py*". W pliku tym znajdzie się kolejny moduł aplikacji. Moduł ten będzie pozwalał wyświetlić bazę danych lamp i telewizorów, z której mieszkańcy hipotetycznego inteligentnego domu, mogą szybko zamówić produkty zastępcze.

Dane są podzielone na dwa pliki. Pierwszy z nich, zapisany w formacie JSON, zawiera dane o produktach. Drugi, zapisany w formacie CSV, przechowuje informacje o identyfikatorach urządzeń w bazie danych MySQL i odpowiadających im kodach produktów w pliku JSON.

W skład podstawowych opcji wchodzić będzie wyświetlenie listy wszystkich dostępnych lamp, wyświetlenie listy wszystkich dostępnych telewizorów, wyświetlenie listy przedmiotów już zakupionych oraz wyświetlenie analizy przeglądowej. Pracę należy rozpocząć od przepisania do nowego pliku następującego szkieletu klasy, który będzie stopniowo uzupełniany:

```
from moduł import *
import mysql.connector

class ModułKatalogu(Moduł):
    # zmienna przechowująca zawartość pliku JSON
    dane = []
```

```
# zmienna przechowująca zawartość pliku CSV
zakupione = []

def __init__(self):
    self.identyfikator = "K"
    self.nazwa = "Katalog urządzeń"
    # Wczytywanie danych katalogu z pliku JSON
    # Wczytywanie danych zakupionych urządzeń z pliku CSV

def przejmijKontrolę(self):
    print("\nKatalog urządzeń:")
    print("========================")
    kursor = self.aplikacja.połączenie.cursor(dictionary=True)
    while (True):
        print("[1] Wypisz lampy")
        print("[2] Wypisz telewizory")
        print("[3] Wypisz urządzenia w domu")
        print("[4] Analiza katalogu")
        print("[KONIEC] Wyjdź z modułu")

        opcja = input("Wybierz opcję z menu: ")
        if opcja.upper() == "KONIEC":
            break
        elif(opcja == "1"):
            self.wypiszLampy()
        elif(opcja == "2"):
            self.wypiszTelewizory()
        elif(opcja == "3"):
            self.wypiszUżywane(kursor)
        elif(opcja == "4"):
            self.analizaDanych()
        else:
            print("Nie ma takiej opcji w menu. Spróbuj ponownie!")
        print("\n")
    kursor.close()

def wypiszLampy(self):
    # Wypisanie linijka po linijce, zawartości podzbioru "dane"
    # zawierającego dane o lampach.
    for pozycja in self.dane["lampy"]:
        print(pozycja)

def wypiszTelewizory(self):
    # Wypisanie linijka po linijce, zawartości podzbioru "dane"
    # zawierającego dane o telewizorach.
    for pozycja in self.dane["telewizory"]:
        print(pozycja)

def wypiszUżywane(self,kursor):
    # Dla każdego produktu na liście
    for pozycja in self.zakupione:
        # Znajdź nazwę urządzenia w bazie
        # Znajdź nazwę produktu w zmiennej "dane"
        # Wypisz informacje

def analizaDanych(self):
    # Przetworzenie danych do postaci wejściowej
    # Wykonanie obliczeń
    # Analiza przeglądowa
    # Analiza korelacji
```

Należy zwrócić uwagę, że parametr *kursor* przekazywany jest wyłącznie do funkcji *wypiszUżywane*. Sugeruje to, że tylko w tej funkcji potrzebne będzie skorzystanie z bazy danych MySQL.

INKLUZJA LIST

Przydatnym narzędziem w realizacji niniejszego ćwiczenia jest inkluzja list[1]. Jest to konstrukt językowy pozwalający jednocześnie przeglądnąć wszystkie elementy listy wejściowej, wybranie elementów spełniających pewien warunek oraz wykonanie na nich operacji i przypisanie do listy wyjściowej. Składnia inkluzji wygląda następująco:

```
wyjście = [f(element) for element in wejście if warunek]
```

Dla przykładu, następujący kod języka Python:

```
x = [1,2,3,4,5,6,7,8,9,10]
y = []
for liczba in x:
    if liczba>3:
      y.append(liczba*2+0.5)
```

Może być zastąpiony następującą inkluzją list:

```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y = [liczba*2+0.5 for liczba in x if liczba>3]
```

Takie operacje będą pomocne podczas pracy z bazami danych zapisanymi w formacie CSV i JSON. O ile język SQL pozwala na wprowadzanie dodatkowych warunków i funkcji do zapytania, o tyle formaty CSV i JSON są statyczne i dostęp do ich treści uzyskuje się w sposób holistyczny. Ich selekcja i przetworzenie leżą więc po stronie programisty.

Dla przykładu, chcąc z danych zapisanych w pliku JSON wydobyć wyłącznie te, których cena jest mniejsza niż 200zł i zakładając, że wszystkie informacje zostały wcześniej pobrane do zmiennej *dane*, wykorzystać należy następującą inkluzję:

```
tanie = [x for x in dane["telewizory"]+dane["lampy"] if x["cena"]<200]
```

Wykonanie tej samej operacji metodą „klasyczną", wymagałoby napisanie większej ilości kodu i przyjęłoby następującą postać:

```
tanie = []
for x in dane["telewizory"]+dane["lampy"]:
```

[1] Oryginalna angielska nazwa brzmi „*list comprehension*". Nie ma zgody co do sposobu tłumaczenie tego zwrotu na język polski. W literaturze proponuje się określenie „wyrażenia listowe" lub „odwzorowywanie list", jednak tłumaczenia te nie oddają w pełni charakterystyki zagadnienia. Sformułowanie użyte w niniejszej instrukcji zostało stworzone w oparciu o inkluzję zbiorów, to jest stosunek zachodzący między dwoma zbiorami, z których jeden zawiera się w drugim. Ta sama idea przyświecała utworzeniu angielskiego odpowiednika.

```
    if x["cena"]<200:
      tanie.append(x)
```

WCZYTYWANIE DANYCH Z PLIKU JSON

Aby móc pracować z danymi w formacie JSON, należy zaimportować pakiet *json*:

```
import json
```

Następnym krokiem jest modyfikacja funkcji *__init__* klasy *ModułKatalogu*. W odpowiednim miejscu tej funkcji należy otworzyć plik JSON, a potem wywołać metodę *load* i przypisać jej wynik do zmiennej składowej klasy, nazwanej *dane*:

```
# Wczytywanie danych z katalogu z pliku JSON
plikKatalogu = open('katalog.json', 'r')
self.dane = json.load(plikKatalogu)
```

Od tego momentu, z danych można korzystać jak ze zwykłej **listy asocjacyjnej**. Oczywiście plik *katalog.json* powinien znaleźć się w tym samym katalogu, co pozostałe pliki projektu.

WCZYTYWANIE DANYCH Z PLIKU CSV

Aby móc pracować z danymi w formacie CSV, należy zaimportować pakiet *csv*:

```
import csv
```

Następnym krokiem jest modyfikacja funkcji *__init__* klasy *ModułKatalogu*. W odpowiednim miejscu tej funkcji należy otworzyć plik CSV. Później należy utworzyć obiekt klasy *DictReader*, jako parametr przekazując **uchwyt** do pliku CSV. Czytnik pozwala na dostęp do danych. Dane należy przeglądnąć i przepisać do zmiennej składowej *zakupione*:

```
# Wczytywanie danych zakupionych urządzeń z pliku CSV
plikZakupionych = open('zakupione.csv')
czytnik = csv.DictReader(plikZakupionych)
for pozycja in czytnik:
    self.zakupione.append({"id":int(pozycja["id"]),"kod":pozycja["kod"]})
```

Od tego momentu, z danych można korzystać jak ze zwykłej **listy asocjacyjnej**. Program zadziała poprawnie tylko jeżeli plik *zakupione.csv* znajdować się będzie w tym samym katalogu, co pozostałe pliki projektu.

WYPISANIE UŻYWANYCH PRODUKTÓW

We wnętrzu pętli rozpoczynającej funkcję *wypiszUżywane* wyszczególnione zostały kroki algorytmu. Pierwszym z nich jest znalezienie **nazwy urządzenia** w bazie MySQL. Należy pobrać wyłącznie to urządzenie, którego *id* jest zgodne z tym, który jest zapisany w zmiennej *pozycja["id"]*. Możliwe jest zrealizowanie tego zagadnienia poprzez zastosowanie omawianego na kilku poprzednich zajęciach schematu. Nazwę urządzenia warto zachować w zmiennej pomocniczej, widocznej spoza bloku *else*. W dalszej części rozdziału, zakłada się, że informacje pobrane z bazy MySQL zapisano do zmiennej *nazwa*.

Drugim etapem jest znalezienie **nazwy produktu** w zmiennej *dane*. Nieco przeszkadza fakt, że produktem może być zarówno lampa jak i telewizor, a te są od siebie oddzielone. Warto dokonać połączenia podlist w jedną:

```
wszystkieProdukty = self.dane["lampy"] + self.dane["telewizory"]
```

Lista wszystkieProdukty wymaga przefiltrowania. Interesujące są jedynie te produkty, których kod jest równy *pozycja["kod"]*. Jako że kody produktów są **unikalne**, przefiltrowana lista składać się będzie z zaledwie jednego elementu. Można go od razu z listy wydobyć:

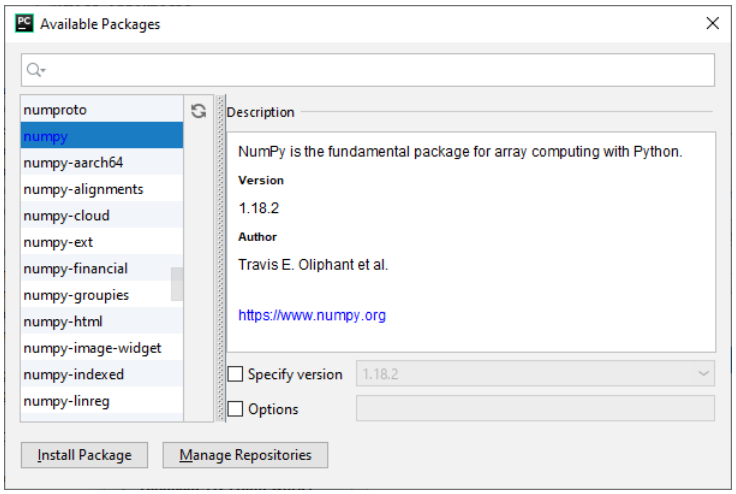
```
produkt = [x for x in wszystkieProdukty if x["kod"] == pozycja["kod"]][0]
```

Ostatnim krokiem jest wypisanie informacji na ekranie. Można tego dokonać za pomocą następującej instrukcji:

```
print(nazwa+": "+produkt["nazwa"]+" (kod: "+pozycja["kod"]+")")
```

IMPLEMENTACJA FUNKCJI ANALIZADANYCH

Istnieje kilka pakietów, które można użyć do przeprowadzenia analizy danych w języku Python. Jednym z nich jest pakiet *numpy*. Przed jego użyciem, należy go **zintegrować** ze środowiskiem PyCharm. Zagadnienie to zostało omówione w instrukcji do pierwszych zajęć. Pakiet nazywa się **numpy**, tak jak to pokazuje poniższy zrzut ekranu:



Oczywiście, aby możliwe było wykorzystanie pakietu, należy go **zaimportować** w pliku „*modułInformacjiOUrządzeniach.py*", tak jak pakiety *json* i *csv*. Analizie przeglądowej podlegać będę ceny lamp. Warto rozpocząć implementację funkcji *analizaDanych* od utworzenia zmiennej pomocniczej, zawierającej wyłącznie ceny lamp:

```
# Przetworzenie danych do postaci wejściowej
cenyLamp = [x.get('cena') for x in self.dane["lampy"]]
```

W kolejnym kroku, utworzone zostaną zmienne pomocnicze, które zawierać będą wynik obliczeń. Podstawienie konkretnych wartości nastąpi w dalszej części instrukcji:

```
# Wykonanie obliczeń
minCenaLampy = ?
medianaCenLamp = ?
średniaCenaLampy = ?
maxCenaLampy = ?
Q1cenLamp = ?
Q3cenLamp = ?
IQRcenLamp = Q3cenLamp - Q1cenLamp
odchylenieCenLamp = ?
wariancjaCenLamp = ?
```

Ciało funkcji powinno kończyć wypisanie obliczonych danych na ekranie:

```
# Analiza przeglądowa cen lamp
print("Analiza przeglądowa cen lamp:")
print("================================")
print("- Minimalna:",minCenaLampy)
print("- Pierwszy kwartyl:",Q1cenLamp)
print("- Średnia:",średniaCenaLampy)
print("- Mediana:",medianaCenLamp)
print("- Trzeci kwartyl:",Q3cenLamp)
print("- Maksymalna:",maxCenaLampy)
print("- IQR:",IQRcenLamp)
print("- Odchylenie standardowe:",odchylenieCenLamp)
print("- Wariancja:",wariancjaCenLamp)
print("================================")
```

ANALIZA DANYCH ZA POMOCĄ BIBLIOTEKI NUMPY

Biblioteka *numpy* udostępnia szereg prostych metod, które obliczają i zwracają pożądane w funkcji *analizaDanych* wartości. Większość argumentów zwróci poprawny wynik, jeżeli pierwszym i jedynym przekazanym argumentem będzie lista wartości. Nieliczne, jak funkcje zwracające wartości percentyli, wymagają podania o który percentyl chodzi. Lista wartości **nie musi być posortowana**, zajmie się tym pakiet *numpy*. Potrzebne do realizacji ćwiczenia funkcje zostały zebrane w poniższej tabeli. Są także dostępne w **dokumentacji** pakietu.

Funkcja	Informacje
MIN	Wartość minimalna z listy
MAX	Wartość maksymalna z listy
MEAN	Średnia arytmetyczna elementów listy
MEDIAN	Wartość mediany elementów listy
STD	Odchylenie standardowe elementów listy
VAR	Wariancja elementów listy
PERCENTILE	Zwraca wartość konkretnego percentyla. Wymaga podania drugiego argumentu, określającego numer percentyla (25 dla Q1 i 75 dla Q3).

Do zaokrąglenia wyników do drugiego miejsca po przecinku, można się posłużyć funkcją *round* z pakietu *math* (który należy oczywiście zaimportować). Przykładowo, uzupełnienie kodu przedstawionego w poprzednim rozdziale instrukcji o obliczanie maksymalnej ceny lamp, miałoby następującą postać:

```
maxCenaLampy = numpy.max(cenyLamp)
```

Lub dodatkowo zaokrąglając wynik do dwóch miejsc po przecinku:

```
maxCenaLampy = math.round(numpy.max(cenyLamp),2)
```

W analogiczny sposób należy obliczyć pozostałe wartości.

ANALIZA KORELACJI

Pakiet *numpy* pozwala także dokonać analizy korelacji. Warto skorzystać z tej możliwości i odpowiedzieć na pytanie: od czego (i jak bardzo) zależy cena lampy?

Najpierw sprawdzona zostanie zależność ceny od siły strumienia świetlnego, gdyż wydaje się, że wartości te powinny być od siebie zależne. Warto utworzyć zmienną pomocniczą, przechowującą listę strumieni świetlnych wszystkich lamp:

```
siłaStrumienia = [x.get('strumień świetlny') for x in self.dane["lampy"]]
```

Obliczenie współczynnika korelacji Pearsona można wykonać wywołując funkcję *corrcoef* biblioteki numpy. Jako argument należy podać listę, której elementami są listy testowanych wartości. W analizowanym przypadku, funkcję corrcoef należy wywołać w następujący sposób:

```
testowaneZmienne = [cenyLamp, siłaStrumienia]
współczynnikKorelacji = numpy.corrcoef(testowaneZmienne)
```

Wynikiem działania funkcji jest macierz współczynników korelacji.

ZADANIA DO SAMODZIELNEGO WYKONANIA

ZADANIE 1

Wykonaj analizę przeglądową cen telewizorów.

ZADANIE 2

Wykonaj analizę korelacji pomiędzy siłą strumienia świetlnego lampy, a ilością punktów światła.

ZADANIE 3 Ⓟ

Zmień działanie funkcji *wypiszUżywane* w taki sposób, aby obok nazwy urządzenia, wypisywała także nazwę pomieszczenia, w którym dane urządzenie się znajduje.

ZADANIE 4 Ⓟ

Wykonaj analizę korelacji pomiędzy ceną lampy, a jej efektywnością. Efektywność może przyjąć jedną z następujących wartości: C, B, A, A+, A++, A+++.

Podpowiedź: w celu obliczenia korelacji, wartości tekstowe należy skonwertować na numeryczne.

ZADANIE 5 Ⓟ

Wykonaj analizę przeglądową przekątnej telewizorów.

ZADANIE 6 Ⓟ

Wykonaj analizę korelacji pomiędzy ceną telewizora, a jego przekątną

ZADANIE 7 Ⓟ

Wykonaj analizę korelacji pomiędzy ceną telewizora, a jego typem matrycy. Matryca może być następującego typu: TN, VA, IPS, Plazma lub OLED.

ZADANIE 8 Ⓟ

Wykonaj analizę korelacji pomiędzy ceną telewizora, a formatem jego ekranu. Format może być następującego typu: 3:2, 4:3, 16:9 lub 1.85:1.

ZADANIE 9 Ⓟ

Wykonaj analizę korelacji pomiędzy ceną telewizora, a jego producentem. W bazie danych znajdują się telewizory następujących producentów: Sharp, Manta, Kruger & Matz, Hitachi, Hyundai, LG, Samsung, Panasonic, Sony i Philips

ZADANIE 10 Ⓟ (NAGRADZANE 2 PLUSAMI)

Wykonaj analizę korelacji pomiędzy ceną telewizora, a tym, czy jest to telewizor typu smartTV oraz tym, czy oferuje możliwość oglądania obrazu w trzech wymiarach. Zwróć uwagę, że pakiet *numpy* pozwala dokonać analizy korelacji jedynie pomiędzy właściwościami numerycznymi.