



PLÁN [OBNOVY]

Nitrianske kompetenčné centrum kybernetickej bezpečnosti
Fakulta prírodných vied a informatiky
Univerzita Konštantína Filozofa v Nitre
Tr. A. Hlinku 1, 949 01 Nitra

SPRACOVANIE OTVORENÝCH DÁT V JAZYKU PYTHON

Téma

Workshop je určený pre učiteľov informatiky. Cieľom je získať vedomosti o získavaní, spracovaní, prezentovaní a interpretácii autentických dát so zámerom podporiť kritické myslenie a formulovanie úsudkov založených na objektívnych dátach.

Gabriela Lovászová
glovaszova@ukf.sk

ABSTRAKT	2
Autori	2
CIEĽOVÁ SKUPINA	2
Vzdelávacie ciele.....	2
Východiskové predpoklady a požiadavky na účastníkov	2
Použité metódy	2
Výstupy pre účastníka	2
POŽIADAVKY	3
Prístrojové vybavenie	3
Priestorové požiadavky.....	3
Veľkosť skupiny.....	3
Ďalšie požiadavky	3
DĹŽKA A FORMÁT.....	3
ÚVOD.....	3
PRIEBEH.....	4
1. Kritické myslenie (45 min) – prednáška s diskusiou	4
2. Autentické a otvorené dáta (10 min) – expozícia pojmov	5
3. Práca so súborami v jazyku Python (20 min) – prehľad poznatkov.....	5
4. Jupyter notebook, Google Colab, Gemini (15 min) – technologický úvod	7
5. Spracovanie jednoduchého súboru autentických dát (45 min) – praktické cvičenie.....	7
6. Spracovanie rozsiahleho súboru otvorených dát (45 min) – praktické cvičenie.....	17
ZÁVER	29

Abstrakt

Workshop je určený pre učiteľov informatiky. Cieľom je rozšíriť vedomosti a zručnosti učiteľov o získavaní, spracovaní, prezentovaní a interpretácii autentických dát so zámerom podporiť kritické myslenie a formulovanie úsudkov založených na objektívnych dátach. Účastníci workshopu získajú praktické vedomosti o zdrojoch otvorených dát, o práci so súbormi dát v jazyku Python, o spracovaní štruktúrovaných dát, o prezentovaní a interpretácii informácií získaných z dát prostredníctvom jazyka Python a webovej platformy Google Colab. Zároveň získajú metodickú pomoc k implementácii témy do vyučovania programovania na strednej škole spôsobom, ktorý je primeraný veku, záujmom, vedomostiam a schopnostiam žiakov.

Autori

Autor1: Gabriela Lovászová

Email1: glovaszova@ukf.sk

Univerzita: Fakulta prírodných vied a informatiky, Univerzita Konštantína Filozofa v Nitre

Cieľová skupina

učitelia informatiky na ZŠ a SŠ (prednostne)

Vzdelávacie ciele

- Chápať význam dátovej gramotnosti ako súboru kompetencií potrebných na prácu s dátami – schopnosť čítať, rozumieť, vytvárať a komunikovať dáta ako informácie,
- získavať súbory otvorených dát z portálov na internete,
- poznať štandardné textové formáty štruktúrovaných dát CSV, JSON, XML,
- deserializovať štruktúrované dáta z textových formátov, serializovať štruktúrované dáta do štandardných textových formátov s využitím modulov jazyka Python,
- spracovať štruktúrované dáta programovaním v jazyku Python,
- vizualizovať dáta prostredníctvom rôznych typov grafov s využitím modulu matplotlib jazyka Python,
- vytvoriť komplexný dokument, ktorý prezentuje výsledky spracovania dát a ich interpretáciu kombinovaním programového kódu s výstupmi výpočtov, textom, obrázkami prostredníctvom platformy Google Colab.

Východiskové predpoklady a požiadavky na účastníkov

- Pokročilé vedomosti z programovania v jazyku Python – práca s údajovými štruktúrami list, tuple, dict, set, práca s textovými súbormi.

Použité metódy

- Prednáška, praktické cvičeniami, diskusia.

Výstupy pre účastníka

- Vzdelávacie materiály – prezentácia a vzorové riešenia príkladov z prednášky,

- získané vedomosti a skúsenosti so získavaním informácií z otvorených dát programovaním v jazyku Python,
- námety na využitie v pedagogickej praxi na rozvoj dátovej gramotnosti žiakov, kritického myslenia, rozhodovania založeného na dôkazoch.

Požiadavky

Prístrojové vybavenie

- počítače s pripojením na internet,
- dataprojektor,
- softvér: Python, Google Colab.

Priestorové požiadavky

- počítačová učebňa

Veľkosť skupiny

- 10 až 20 učiteľov informatiky alebo budúcich učiteľov informatiky

Ďalšie požiadavky

- nie sú

Dĺžka a formát

- Trvanie: 3 hodiny (4 x 45 minút)
- Forma: prezenčne / online (Zoom, MS Teams, Meet)

Úvod

Na workshope sa venujeme téme kritického myslenia a spôsobom jeho rozvíjania prostredníctvom práce s autentickými dátami. Konceptualizujeme chápanie pojmu kritické myslenie a pomenujeme možné príčiny krízy kritického myslenia súvisiace s informatizáciou spoločnosti. Priestor venujeme diskusii s učiteľmi o ich skúsenostiach a názoroch na úroveň kritického myslenia žiakov a na úlohu kritického myslenia, ktorú zohráva pri bezpečnom používaní digitálnych technológií a eliminácii rizík.

V praktickej časti workshopu sa venujeme spracovaniu autentických dát v programovacom jazyku Python. Predstavíme niektoré zdroje otvorených dát a na príkladoch demonštrujeme, ako sa dajú získavať informácie zo súborov otvorených dát. Dôraz je kladený na kritické myslenie – formulovanie problému, analýzu dát, hodnotenie, rozhodovanie, interpretáciu a argumentáciu, nie na samotné programovanie. Preto používame pri tvorbe programového kódu technológie umelej inteligencie. Práca s nástrojmi umelej inteligencie na workshope má za cieľ aplikovať v praxi základné princípy práce s umelou inteligenciou, ktoré definujú úlohy človeka a umelej inteligencie pri riešení problému.

alebo práci na projekte: človek rieši koncepčné návrhy, realizuje strategické rozhodnutia a nesie zodpovednosť za výsledok, umelá inteligencia rieši realizáciu.

Prínosom workshopu pre učiteľov informatiky má byť vlastná skúsenosť so spracovaním súborov autentických dát s využitím moderných technológií ako inšpirácia pre využitie vo vyučovaní na rozvoj generických zručností – dátovej gramotnosti, kritického myslenia a rozhodovania založeného na dôkazoch.

Priebeh

1. Kritické myslenie (45 min) – prednáška s diskusiou

Kritické myslenie je taký spôsob myslenia, ktorý umožňuje jednotlivcovi

- interpretovať, analyzovať a hodnotiť kvalitu informácií a myšlienkových postupov (vlastných alebo cudzích),
- na základe toho formulovať úsudky, prezentovať a obhajovať závery.

Dimenzie kritického myslenia

- kognitívna
 - interpretácia
 - analýza
 - hodnotenie
 - usudzovanie
 - argumentácia
 - rozhodovanie
- osobnostná
 - sebareflexia a sebaregulácia
 - dôvera vo vlastný úsudok a silu rozumu
 - chápanie rozmanitých hľadísk, nezaujatosť
 - otvorená myseľ, flexibilita
 - odmietanie povrchnosti
 - široký okruh záujmov

Problémy s kritickým myslením

- slobodný a jednoduchý prístup k informáciám rôznej kvality
- pretlak informácií → frustrácia, skepsa
- efektívnosť digitálnych technológií → povrchnosť (syndróm copy-paste)
- „vojna“ proti memorovaniu spôsobom „všetko je na webe, stačí vedieť hľadať“ → memorovanie (zapamätanie bez porozumenia) sa stotožňuje s akýmkoľvek osvojovaním vedomostí → nízka úroveň vedomostí
- „neomylnosť“ digitálnych technológií → nekritická viera v správnosť fungovania hardvéru a softvéru prenášaná aj na údaje
- anonymita virtuálneho prostredia → argumentácia založená na afektoch

Stratégie rozvoja kritického myslenia vo vyučovaní programovania

- analýza a interpretácia dát,
- identifikácia a pomenovanie problémov, formulácia otázok,

- riešenie problémov, argumentácia,
- tvorba originálnych riešení, myšlienok, produktov,
- argumentácia.

Umelá inteligencia a kritické myslenie

- rozdelenie úloh: človek sa sústreďuje na *koncepčné návrhy*, strategické *rozhodovanie*, umelá inteligencia rieši *realizáciu*
- riziká:
 - UI potrebuje veľké množstvá dát – riziko *neetického zberu* a používania dát
 - algoritmus UI môže generovať *zaujaté alebo nesprávne závery*
 - zneužitie UI na *neetické ciele*
 - autorstvo – človek má *rozhodovaciú právomoc a zodpovednosť* za dielo, jeho kvalitu a spôsob použitia
 - právne rámce zaostávajú za technológiou

2. Autentické a otvorené dáta (10 min) – expozícia pojmov

Autentické dáta - reálne dáta pochádzajúce z overiteľného a legitímneho zdroja

Otvorené dáta

- voľne a bezplatne dostupné pre každého za rovnakých podmienok,
- použiteľné na akýkoľvek účel komerčného či nekomerčného charakteru,
- sprístupnené na internete v štruktúrovanej forme, ktorá umožňuje ich hromadné strojové spracovanie.

Metadáta - dáta o dátach (autor, platnosť, štruktúra)

Zdroje otvorených dát

- Národný katalóg otvorených dát <https://data.slovensko.sk/>
- Eurostat (štatistiky a dáta o Európskej únii) <https://ec.europa.eu/eurostat/web/main/home>
- Kaggle (komunita záujemcov o umelú inteligenciu a strojové učenie)
<https://www.kaggle.com/datasets>

3. Práca so súbormi v jazyku Python (20 min) – prehľad poznatkov

Formáty otvorených dát

CSV (Comma-Separated Values), TSV (Tab-Separated Values)

- textový súbor na uchovávanie tabuľkových dát
- riadky sú oddelené '\n'
- stĺpce sú oddelené čiarkou (bodkočiarkou) alebo tabulátorom '\t'

```
Meno,Vek,Mesto
Jana,30,Bratislava
Peter,45,Košice
```

JSON (JavaScript Object Notation)

- textový súbor na uchovávanie štruktúrovaných nelineárnych dát
- v Pythone objekty typu reťazec (str), číslo (int, float), logická hodnota (bool), slovník (dict), zoznam (list), n-tica (tuple)

```
{
  "nazov_knihy": "Hobit",
  "autor": {
    "krstne_meno": "J.R.R.",
    "priezvisko": "Tolkien",
    "rok_narodenia": 1892
  },
  "zanre": [
    "Fantazy",
    "Dobrodružné"
  ],
  "dostupna": true,
  "cena": 15.99,
  "vydavatel": null
}
```

XML (eXtensible Markup Language)

- *textový súbor* na uchovávanie štruktúrovaných *nelineárnych dát*

```
<?xml version="1.0" encoding="UTF-8"?>
<katalog>
  <kniha id="456" dostupna="ano">
    <nazov>Meno ruže</nazov>
    <autor>Umberto Eco</autor>
    <rok_vydania>1980</rok_vydania>
    <zanre>
      <zaner>Historický</zaner>
      <zaner>Detektívka</zaner>
    </zanre>
  </kniha>
</katalog>
```

Čítanie zo súboru v jazyku Python

with open(súbor, mód) as premenná:
príkazy

- Mód môže byť:
- 'r' (read) – zo súboru sa dá iba čítať – predvolená hodnota
- 'w' (write) – do súboru sa dá zapisovať – vytvorí sa nový súbor, alebo sa existujúci prepíše novými hodnotami
- 'a' (append) – do súboru sa dá zapisovať na koniec pôvodného obsahu
- Do premennej sa priradí referencia na dátový prúd otvorený funkciou open() a vykonajú sa vnorené príkazy.
- Súbor je otvorený len vo vnútri štruktúry with, mimo nej sa automaticky zatvorí.

Metódy na čítanie zo súborového prúdu dát

`súbor.read()` – vráti obsah súborového prúdu ako jeden znakový reťazec

`súbor.read(počet)` – vráti daný počet znakov zo súborového prúdu dát ako znakový reťazec

`súbor.readline()` – vráti aktuálny riadok zo súborového prúdu dát ako znakový reťazec

`súbor.readlines()` – vráti zoznam riadkov (znakových reťazcov) v súborovom prúde dát

4. Jupyter Notebook, Google Colab, Gemini (15 min) – technologický úvod

Jupyter Notebook – webová aplikácia, ktorá umožňuje vytvárať a zdieľať dokumenty, ktoré môžu obsahovať:

- Kód – spustiteľný kód (v Pythone)
- Výstup – výsledky výpočtov (texty, tabuľky, grafy, obrázky)
- Text – textové popisy, poznámky, vložené obrázky formátované pomocou značkovacieho jazyka (Markdown)

Google Colab – cloudová služba od Google na prácu s Jupyter notebookmi:

- Nevyžaduje si žiadne nastavenie ani inštaláciu na počítači.
- Poskytuje prístup k výkonnému hardvéru na zložité výpočty.
- Umožňuje jednoducho zdieľať a spolupracovať na notebookoch s inými ľuďmi.

Gemini – v Google Colab funguje ako AI asistent na:

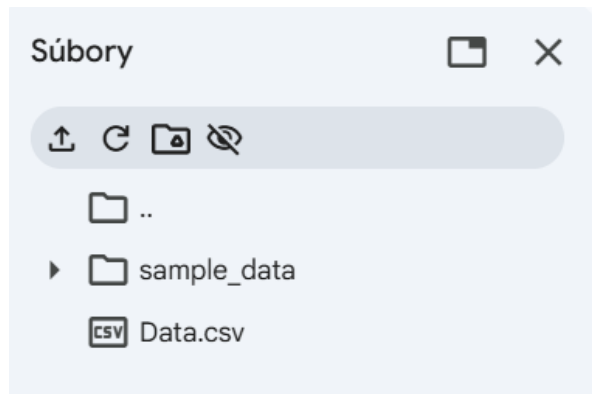
- generovanie kódu
- dopĺňanie kódu
- vysvetľovanie kódu
- opravu chýb
- chat – diskusiu o obsahu v notebooku

5. Spracovanie jednoduchého súboru autentických dát (45 min) – praktické cvičenie

Pracujeme so súborom autentických dát o premávke, ktoré boli reálne zozbierané pomocou mobilnej aplikácie. Súbor obsahuje záznamy o čase a type vozidla, ktoré v danom čase prešlo na sledovanom úseku cestnej komunikácie. Súbor má formát CSV. Úlohou je spracovať súbor a extrahovať informácie z dát. Pracuje sa metódou riešenia postupnosti úloh.

Úloha 1: Sprístupnenie súboru

Načítajte súbor `Data.csv` do dočasného priečinka `/content/` v prostredí Google Colab



Obrázok 1: Panel Súborny prostredia Google Colab

Úloha 2: Načítanie a úprava dát zo súboru

a, Do premennej data načítajte celý obsah súboru. Akého typu je premenná data?

```
# Nastavenie cesty k súboru
file_path = '/content/Data.csv'

# Načítanie obsahu súboru do premennej data
with open(file_path, 'r', encoding='utf-8') as file:
    data = file.read()

print(type(data)) # Vypíše typ premennej data
print(data[:200]) # Vypíše prvých 200 znakov pre kontrolu
```

```
<class 'str'>
1;Auto;08:41:04
10;Auto;08:41:27
100;Auto;08:46:16
101;Auto;08:46:17
102;Auto;08:46:18
103;Auto;08:4
```

b, Do premennej data teraz načítajte obsah súboru po riadkoch. Akého typu je teraz premenná data?

```
# Nastavenie cesty k súboru
file_path = '/content/Data.csv'

# Načítanie obsahu súboru do premennej data
with open(file_path, 'r', encoding='utf-8') as file:
    data = file.readlines()

print(type(data)) # Vypíše typ premennej data
print(data[:3]) # Vypíše prvé 3 riadky pre kontrolu
```

```
<class 'list'>
['1;Auto;08:41:04\n', '10;Auto;08:41:27\n', '100;Auto;08:46:16\n']
```

c, Jednotlivé riadky zoznamu data rozdeľte podľa bodkočiarky do podzoznamov.

d, Z prvého údaja v každom riadku vytvorte celé číslo.

e, Tretí údaj v riadku (čas) rozdeľte na hodiny, minúty, sekundy.

```
# Nastavenie cesty k súboru
file_path = '/content/Data.csv'

# Načítanie obsahu súboru po riadkoch do zoznamu, rozdelenie podľa
# bodkočiarky a konverzia prvého prvku na int a rozdelenie času
with open(file_path, 'r', encoding='utf-8') as file:
    data = []
    for line in file:
        cleaned_line = line.strip()
        split_line = cleaned_line.split(';')
        # Konvertujeme prvý prvok na celé číslo
        split_line[0] = int(split_line[0])

        # Rozdelíme tretí prvok (čas) na hodiny, minúty, sekundy
        time_str = split_line[2]
        hours, minutes, seconds = map(int, time_str.split(':'))

        # Nahradíme pôvodný čas rozdelenými hodnotami, alebo ich pridáme
        # ako nové prvky
        # Rozhodol som sa pridať ich ako nové prvky, aby sa zachovala
        # pôvodná štruktúra
        split_line.extend([hours, minutes, seconds])

        data.append(split_line)

print(type(data))

# Vypísanie prvých 5 spracovaných riadkov pre kontrolu
print(data[:5])
```

```
<class 'list'>
[[1, 'Auto', '08:41:04', 8, 41, 4], [10, 'Auto', '08:41:27', 8, 41, 27],
[100, 'Auto', '08:46:16', 8, 46, 16], [101, 'Auto', '08:46:17', 8, 46, 17],
[102, 'Auto', '08:46:18', 8, 46, 18]]
```

f, Usporiadajte riadky v zozname data podľa čísla na prvom mieste v riadku.

```
# Usporiadanie zoznamu data podľa prvého prvku (čísla)
data.sort(key=lambda riadok: riadok[0])

# Vypísanie prvých 10 riadkov zoznamu data
for riadok in data[:10]:
    print(riadok)
```

```
[1, 'Auto', '08:41:04', 8, 41, 4]
[2, 'Auto', '08:41:05', 8, 41, 5]
[3, 'Auto', '08:41:08', 8, 41, 8]
[4, 'Auto', '08:41:10', 8, 41, 10]
[5, 'Auto', '08:41:12', 8, 41, 12]
[6, 'Auto', '08:41:16', 8, 41, 16]
[7, 'Auto', '08:41:18', 8, 41, 18]
[8, 'Auto', '08:41:24', 8, 41, 24]
```

```
[9, 'Nakladne auto', '08:41:26', 8, 41, 26]
[10, 'Auto', '08:41:27', 8, 41, 27]
```

g, Pridajte do zoznamu data hlavičky stĺpcov.

```
# Pridanie hlavičiek stĺpcov na začiatok zoznamu
headers = ['Číslo', 'Typ', 'Čas', 'Hodiny', 'Minúty', 'Sekundy']
data.insert(0, headers)

# Vypísanie prvých 5 riadkov zoznamu data vrátane
for riadok in data[:6]:
    print(riadok)
```

```
['Číslo', 'Typ', 'Čas', 'Hodiny', 'Minúty', 'Sekundy']
[1, 'Auto', '08:41:04', 8, 41, 4]
[2, 'Auto', '08:41:05', 8, 41, 5]
[3, 'Auto', '08:41:08', 8, 41, 8]
[4, 'Auto', '08:41:10', 8, 41, 10]
[5, 'Auto', '08:41:12', 8, 41, 12]
```

Úloha 3: Získavanie informácií z dát

a, Zistite, koľko záznamov je v zozname data.

```
# Zistenie počtu záznamov v zozname data
pocet_zaznamov = len(data[1:])

# Vypísanie počtu záznamov
print(f"Počet záznamov v zozname data je: {pocet_zaznamov}")
```

```
Počet záznamov v zozname data je: 157
```

b, Zistite, aké jedinečné hodnoty sú v druhom stĺpci tabuľky data.

c, Zistite, koľko záznamov sa nachádza v tabuľke data pre každú jedinečnú hodnotu typ vozidla.

d, Vypočítajte aj percentuálne zastúpenie jednotlivých typov vozidiel.

```
# Získanie jedinečných hodnôt z druhého stĺpca (index 1), preskočíme
# hlavičku (prvý riadok)
jedinecne_hodnoty = set(riadok[1] for riadok in data[1:])

# Vypísanie jedinečných hodnôt
print("Jedinečné hodnoty v druhom stĺpci sú:")
for hodnota in jedinecne_hodnoty:
    print(hodnota)

# Spočítanie výskytu každej jedinečnej hodnoty
pocet_podla_typu = {}
for riadok in data[1:]:
    typ_vozidla = riadok[1]
    pocet_podla_typu[typ_vozidla] = pocet_podla_typu.get(typ_vozidla, 0) + 1
```

```
# Zistenie celkového počtu záznamov (bez hlavičky)
celkovy_pocet = len(data[1:])

# Vypísanie počtu záznamov a percentuálneho zastúpenia pre každý typ
# vozidla v jednom cykle
print("\nPôčet záznamov a percentuálne zastúpenie pre každý typ vozidla:")
for typ, pocet in pocet_podla_typu.items():
    percento = (pocet / celkovy_pocet) * 100 if celkovy_pocet > 0 else 0
    print(f"{typ}: {pocet} ({percento:.2f}%)")
```

Jedinečné hodnoty v druhom stĺpci sú:

```
Zachranne zlozky
Extra vozidlo
Motocykel
Nakladne auto
Kamion
Auto
Bicykel
```

Pôčet záznamov a percentuálne zastúpenie pre každý typ vozidla:

```
Auto: 127 (80.89%)
Nakladne auto: 13 (8.28%)
Kamion: 12 (7.64%)
Motocykel: 1 (0.64%)
Extra vozidlo: 1 (0.64%)
Zachranne zlozky: 1 (0.64%)
Bicykel: 2 (1.27%)
```

Úloha 4: Vizualizácia

a, Vytvorte stĺpcový graf, ktorý zobrazuje počty jednotlivých typov vozidiel za sledované časové obdobie.

```
import matplotlib.pyplot as plt
import numpy as np

# Získanie názvov typov vozidiel a ich počtov z dictionary
typy = list(pocet_podla_typu.keys())
pocety = list(pocet_podla_typu.values())

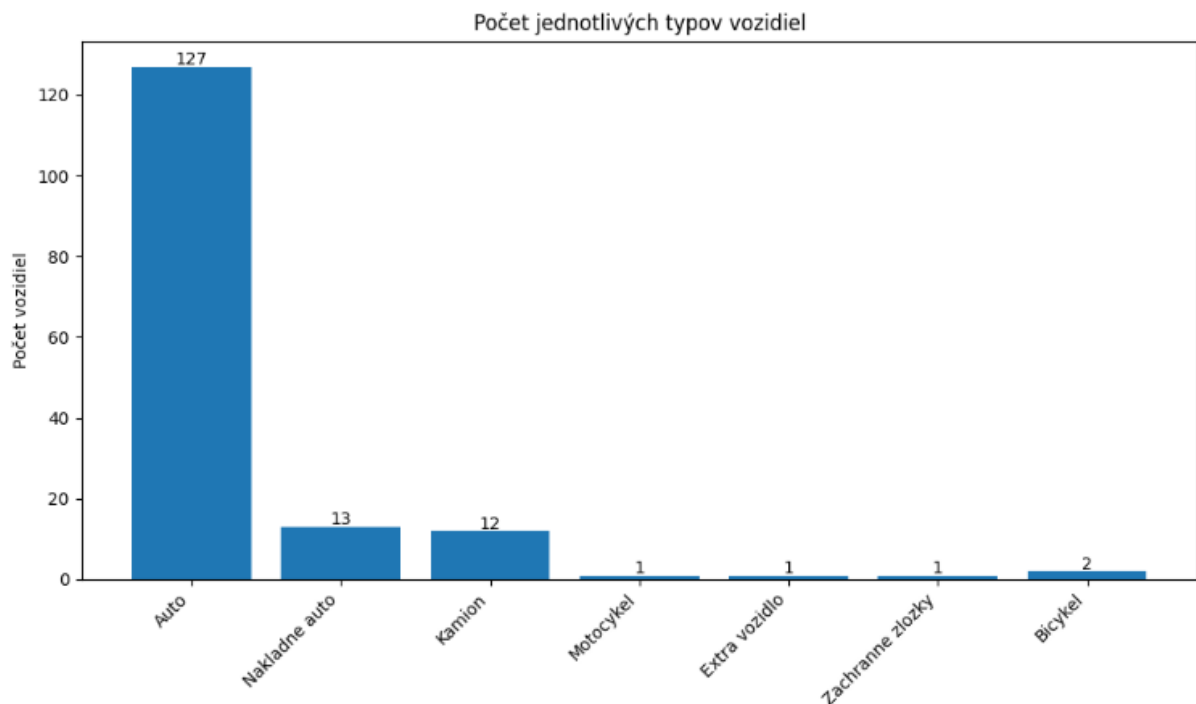
# Vytvorenie stĺpcového grafu
fig, ax = plt.subplots(figsize=(10, 6)) # Nastavenie veľkosti grafu

x_pos = np.arange(len(typy)) # Pozície stĺpcov na osi x
ax.bar(x_pos, pocety, align='center') # Vytvorenie zvislého stĺpcového grafu
ax.set_xticks(x_pos) # Nastavenie značiek na osi x
ax.set_xticklabels(typy, rotation=45, ha='right') # Nastavenie názvov
# značiek na osi x s rotáciou pre lepšiu čitateľnosť
ax.set_ylabel('Pôčet vozidiel') # Nastavenie popisu osi y
ax.set_title('Pôčet jednotlivých typov vozidiel') # Nastavenie názvu grafu

# Pridanie počtov k stĺpcom
for i, v in enumerate(pocety):
```

```
ax.text(i, v + 0.5, str(v), color='black', ha='center') # Umiestnenie
textu (počtu) nad stĺpcom

plt.tight_layout() # Upravenie rozloženia, aby sa popisy neprekrývali
plt.show()
```



Obrázok 2: Stĺpcový graf

b, Vytvorte koláčový graf percentuálnych podielov jednotlivých typov vozidiel za sledované časové obdobie.

```
import matplotlib.pyplot as plt

# Získanie názvov typov vozidiel a ich počtov z dictionary
typy = list(pocet_podla_typu.keys())
pocety = list(pocet_podla_typu.values())

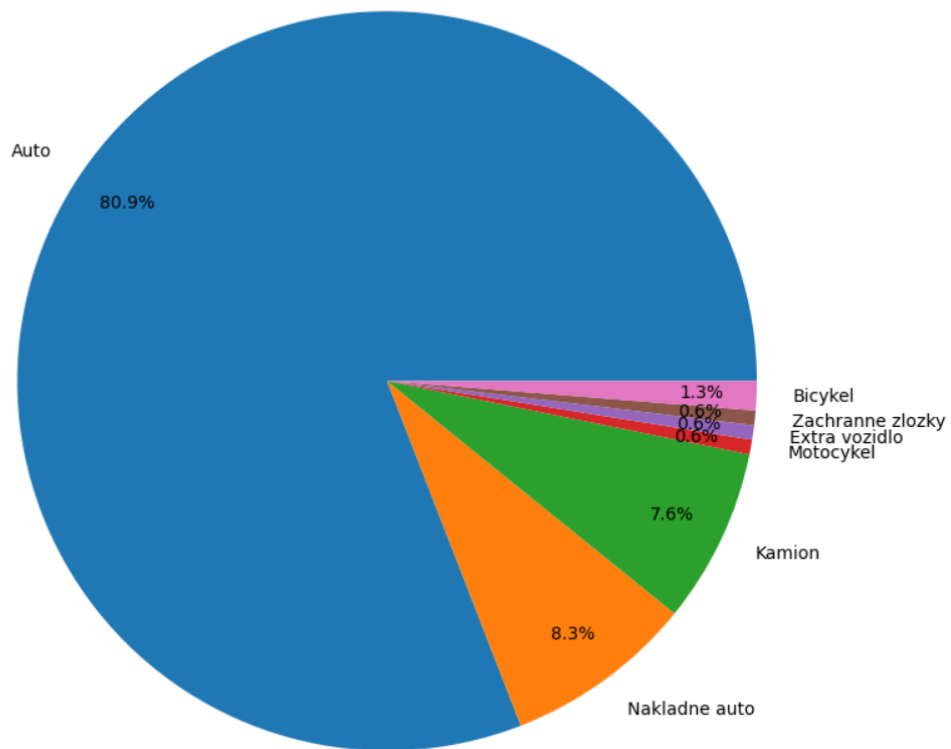
# Vytvorenie koláčového grafu
fig, ax = plt.subplots(figsize=(8, 8)) # Nastavenie veľkosti grafu

# Vytvorenie koláčového grafu s percentami a úpravou polohy percent
ax.pie(pocety, labels=typy, autopct='%1.1f%%', startangle=0,
       pctdistance=0.85) # Pridaný pctdistance

# Nastavenie názvu grafu
ax.set_title('Percentuálne zastúpenie jednotlivých typov vozidiel')

# Zabezpečenie, aby bol graf kruhový
ax.axis('equal')
plt.tight_layout() # Upravenie rozloženia, aby sa popisy neprekrývali
plt.show()
```

Percentuálne zastúpenie jednotlivých typov vozidiel



Obrázok 3: Koláčový graf

Úloha 5: Získanie ďalších informácií a ich vizualizácia

a, Zisti hustotu premávky – počet vozidiel za 1 minútu za celé sledované obdobie.

Task

Zisti hustotu premávky – počet vozidiel za 1 minútu za celé sledované obdobie.

Zistenie začiatku a konca sledovaného obdobia

Subtask:

Nájdite najskorší a najneskorší časový údaj v dátach.

```
# Najdenie najskoršieho a najneskoršieho časového údaj v dátach (index 2)
start_time = min(riadok[2] for riadok in data[1:])
end_time = max(riadok[2] for riadok in data[1:])

print(f"Najskorší čas: {start_time}")
print(f"Najneskorší čas: {end_time}")
```

```
Najskorší čas: 08:41:04
Najneskorší čas: 08:49:16
```

Výpočet celkovej dĺžky sledovaného obdobia v minútach

Subtask:

Vypočítajte rozdiel medzi najneskorším a najskorším časom a prevedte ho na minúty.

```
from datetime import datetime, timedelta

# Definujte funkciu, ktorá prevedie časový reťazec na sekundy od polnoci
def time_string_to_seconds(time_str):
    hours, minutes, seconds = map(int, time_str.split(':'))
    return hours * 3600 + minutes * 60 + seconds

# Prevedenie start_time a end_time na sekundy od polnoci
start_seconds = time_string_to_seconds(start_time)
end_seconds = time_string_to_seconds(end_time)

# Vypočítajte rozdiel v sekundách
duration_seconds = end_seconds - start_seconds

# Prevedte rozdiel v sekundách na minúty
total_duration_minutes = duration_seconds / 60

# Vypíšte celkovú dĺžku sledovaného obdobia v minútach
print(f"Celková dĺžka sledovaného obdobia je: {total_duration_minutes:.2f} minút")
```

Celková dĺžka sledovaného obdobia je: 8.20 minút

Získanie celkového počtu vozidiel

Subtask:

Použite už vypočítaný celkový počet záznamov (vozidiel).

```
# The total number of records (vehicles) was previously calculated and stored in the pocet_zaznamov variable.
# This subtask only requires retrieving this value.
print(f"Total number of records (vehicles): {pocet_zaznamov}")
```

Total number of records (vehicles): 157

Výpočet hustoty premávky

Subtask:

Vydeľte celkový počet vozidiel celkovou dĺžkou sledovaného obdobia v minútach.

```
# Vypočítajte hustotu premávky
traffic_density_per_minute = pocet_zaznamov / total_duration_minutes

# Vypíšte hustotu premávky
print(f"Hustota premávky za 1 minútu: {traffic_density_per_minute:.2f} vozidiel/minútu")
```

Hustota premávky za 1 minútu: 19.15 vozidiel/minútu

Zhrnutie:

Kľúčové poznatky z analýzy údajov

Najskorší čas zaznamenaný v súbore údajov je '08:41:04' a najneskorší čas je '08:49:16'.

- Celková dĺžka sledovaného obdobia je 8.20 minút.
- Celkový počet záznamov (vozidiel) je 157.
- Vypočítaná hustota premávky počas sledovaného obdobia je 19.15 vozidiel za minútu.

Poznatky alebo ďalšie kroky

- Hustota premávky 19.15 vozidiel/min poskytuje kvantitatívne meranie toho, aká bola premávka počas sledovaného časového rámca.
- Pre získanie hlbších poznatkov by sa hustota premávky mohla analyzovať pre kratšie intervaly v rámci sledovaného obdobia, aby sa identifikovali časy špičky.

b, Zistite, aký počet vozidiel bol zaznamenaný v jednotlivých minútach v sledovanom časovom období.

```
# Získanie minút z časového údajá pre každý záznam (preskočíme hlavičku)
# Vytvoríme kľúč pre každú minútu v rámci celého obdobia
# Predpokladáme, že sledované obdobie nepresahuje jednu hodinu,
# inak by sme museli zohľadniť aj hodiny pre unikátnosť minút
minuty_zaznamov = [riadok[4] for riadok in data[1:]]

# Spočítanie výskytu vozidiel pre každú minútu
pocet_podla_minuty = {}
for minuta in minuti_zaznamov:
    pocet_podla_minuty[minuta] = pocet_podla_minuty.get(minuta, 0) + 1

# Usporiadanie výsledkov podľa minúty
usporiadany_pocet_podla_minuty =
dict(sorted(pocet_podla_minuty.items()))

# Vypísanie počtu vozidiel pre každú minútu
print("Počet vozidiel pre jednotlivé minúty:")
for minuta, pocet in usporiadany_pocet_podla_minuty.items():
    print(f"Minúta {minuta}: {pocet} vozidiel")
```

Počet vozidiel pre jednotlivé minúty:

Minúta 41: 23 vozidiel
Minúta 42: 8 vozidiel
Minúta 43: 26 vozidiel
Minúta 44: 11 vozidiel
Minúta 45: 24 vozidiel
Minúta 46: 16 vozidiel
Minúta 47: 21 vozidiel
Minúta 48: 18 vozidiel

Minúta 49: 10 vozidiel

c, Vytvorte čiarový graf počtu vozidiel v jednotlivých minútach.

```
import matplotlib.pyplot as plt

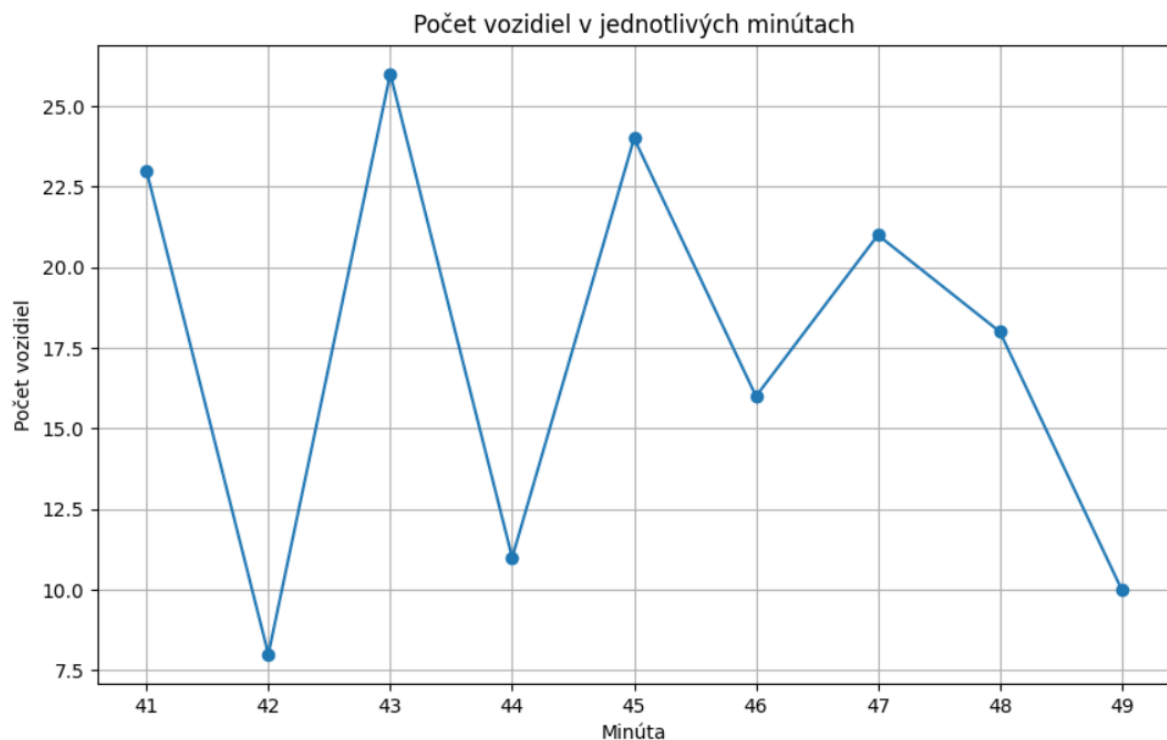
# Získanie minút a počtov vozidiel z usporiadanej dictionary
minuty = list(usporiadany_pocet_podla_minuty.keys())
pocety = list(usporiadany_pocet_podla_minuty.values())

# Vytvorenie čiarového grafu
fig, ax = plt.subplots(figsize=(10, 6)) # Nastavenie veľkosti grafu

ax.plot(minuty, pocety, marker='o', linestyle='-') # Vytvorenie
čiarového grafu

ax.set_xlabel('Minúta') # Nastavenie popisu osi x
ax.set_ylabel('Počet vozidiel') # Nastavenie popisu osi y
ax.set_title('Počet vozidiel v jednotlivých minútach') # Nastavenie
názvu grafu
ax.grid(True) # Zobrazenie mriežky

plt.show()
```



Obrázok 4: Čiarový graf – Počet vozidiel v minútových intervaloch

6. Spracovanie rozsiahleho súboru otvorených dát (45 min) – praktické cvičenie

Pracuje sa s rozsiahlym súborom otvorených dát vo formáte TSV projektovou metódou.

Riešený problém

Preskúmame, ako sa mení intenzita leteckej dopravy v krajinách Európy:

- ako sa celkovo menil počet cestujúcich leteckou dopravou v Európe počas rokov 2013-2024,
- ktoré krajiny zaznamenali najvyšší a najnižší počet cestujúcich,
- ako ovplyvnila pandémia COVID-19 leteckú dopravu (obdobie 2020-2022),
- či po roku 2022 došlo k obnoveniu leteckej dopravy na úroveň pred pandémiou.

Zdroj dát

Dátový súbor pochádza z databázy Eurostat a obsahuje mesačné štatistické údaje o počte cestujúcich v leteckej osobnej doprave v európskych krajinách za obdobie rokov 2013 až 2024. Dáta sú dostupné a prispôsobiteľné na:

https://ec.europa.eu/eurostat/databrowser/view/avia_paoc_custom_19801867/default/table

Spracovávaný súbor `avia_paoc_custom_19800805_monthly_tabular.tsv` je potrebné vložiť do dočasného priečinka `/content/` v prostredí Google Colab.

Postup riešenia

1. Načítanie a štruktúrovanie dát zo súboru `/content/avia_paoc_custom_19800805_monthly_tabular.tsv` do štruktúrovanej formy (zoznam zoznamov).
2. Úprava dát: Identifikácia a odstránenie redundantných stĺpcov a konverzia dátových typov.
3. Filtrovanie údajov podľa krajín a vizualizácia dát zobrazujúca vývoj počtu cestujúcich v čase.
4. Výpočet najnižšieho a najvyššieho počtu cestujúcich pre každý región.
5. Výpočet priemerného počtu cestujúcich v predpandemickom, pandemickom a postpandemickom období pre každý región, vizualizácia.
6. Formulácia záverov.

Úloha 1: Načítanie a úprava dát

Dáta zo súboru TSV načítame do zoznamu riadkov, riadky rozdelíme podľa tabulátora na stĺpce, prvý stĺpec ešte rozdelíme podľa čiarky. Výsledkom je tabuľka - zoznam zoznamov `processed_rows`.

```
file_path = "/content/avia_paoc_custom_19800805_monthly_tabular.tsv"

try:
    with open(file_path, 'r') as file:
        content = file.read()

    raw_rows = []
    for line in content.strip().split('\n'):
        raw_rows.append(line.split('\t'))

    # Spracovanie hlavičky
    original_header_first_col = raw_rows[0][0]
```

```

split_headers = original_header_first_col.split(',')
new_header = split_headers + raw_rows[0][1:]

# Spracovanie dátových riadkov
processed_rows = [new_header]
for row_idx, row in enumerate(raw_rows[1:]):
    if row:
        first_col_data = row[0]
        split_first_col_data = first_col_data.split(',')
        # Skontrolujte, či má prvý stĺpec dát dostatok častí na prirad
enie k hlavičke
        # Ak má len jednu časť, použijeme ju, inak rozdelíme.
        if len(split_first_col_data) == len(split_headers):
            new_row = split_first_col_data + row[1:]
        else:
            # Ak sa prvý stĺpec nedá úplne rozdeliť podľa hlavičky, pr
ipojíme ho ako jeden prvok
            new_row = [first_col_data] + row[1:]
        processed_rows.append(new_row)

print("Prvých 5 spracovaných riadkov dát:")
for i, row in enumerate(processed_rows):
    if i >= 5:
        break
    print(row)

except FileNotFoundError:
    print(f"Chyba: Súbor na ceste '{file_path}' nebol nájdený. Skúste skon
trolovať, či cesta k súboru je správna.")
except Exception as e:
    print(f"Nastala neočakávaná chyba pri čítaní súboru: {e}")

```

Analyzujeme obsah tabuľky. Prvý riadok obsahuje hlavičky stĺpcov tabuľky. Zistíme jedinečné hodnoty v prvých 6 stĺpcoch tabuľky.

```

unique_values_per_column = [set() for _ in range(6)]

# Preskočíme hlavičku a iterujeme cez spracované riadky
for row_data in processed_rows[1:]:
    for i in range(min(6, len(row_data))):
        unique_values_per_column[i].add(row_data[i].strip())

for i, unique_vals in enumerate(unique_values_per_column):
    print(f"Jedinečné hodnoty pre stĺpec '{new_header[i]}':")
    print(list(unique_vals))
    print("\n")

```

```

Jedinečné hodnoty pre stĺpec 'freq':
['M']

```

```
Jedinečné hodnoty pre stĺpec 'unit':
['PAS']

Jedinečné hodnoty pre stĺpec 'tra_meas':
['PAS_CRD_DEP', 'PAS_BRD_ARR', 'PAS_BRD_DEP', 'PAS_BRD', 'PAS_CRD_ARR',
'PAS_CRD']

Jedinečné hodnoty pre stĺpec 'tra_cov':
['TOTAL']

Jedinečné hodnoty pre stĺpec 'schedule':
['TOT']

Jedinečné hodnoty pre stĺpec 'geo\TIME_PERIOD':
['IE', 'PT', 'EE', 'EU28', 'LU', 'NL', 'ES', 'IT', 'RO', 'BG', 'HR', 'TR',
'BE', 'PL', 'SE', 'CZ', 'SI', 'BA', 'EU27_2007', 'DK', 'EL', 'LT', 'SK',
'ME', 'IS', 'CY', 'DE', 'FI', 'UK', 'HU', 'LV', 'MK', 'NO', 'CH', 'RS',
'FR', 'MT', 'AT', 'EU27_2020']
```

Z tabuľky odstránime stĺpce, v ktorých sa nachádza len jedna jedinečná hodnota.

```
# Identifikácia stĺpcov na odstránenie
columns_to_remove_indices = []
for i, unique_vals in enumerate(unique_values_per_column):
    if len(unique_vals) == 1:
        columns_to_remove_indices.append(i)

# Identifikácia riadkov na odstránenie
rows_to_remove_indices = []
for i, row in enumerate(processed_rows):
    if row[5] in ['EU28', 'EU27_2007', 'EU27_2020']:
        rows_to_remove_indices.append(i)

# Vytvorenie nových riadkov bez týchto stĺpcov
filtered_processed_rows = []
for i, row_data in enumerate(processed_rows):
    if i in rows_to_remove_indices:
        continue
    new_row = [value for idx, value in enumerate(row_data) if idx not in columns_to_remove_indices]
    filtered_processed_rows.append(new_row)

# Aktualizácia processed_rows na nové filtrované riadky
processed_rows = filtered_processed_rows

print("Prvých 5 riadkov po odstránení stĺpcov s jednou jedinečnou hodnotou:")
for i, row in enumerate(processed_rows):
    if i >= 5:
        break
    print(row)
```

Upravíme typy hodnôt v tabuľke - číselné hodnoty na celé čísla, dvojbodky na `None`.

```
for row_index in range(len(processed_rows)):
    for col_index in range(len(processed_rows[row_index])):
        try:
            if processed_rows[row_index][col_index].strip() == ':':
                processed_rows[row_index][col_index] = None
            else:
                processed_rows[row_index][col_index] = int(processed_rows[row_index][col_index])
        except ValueError:
            continue

print("Prvých 5 riadkov po úprave typu hodnôt v tabuľke na čísla:")
for i, row in enumerate(processed_rows):
    if i >= 5:
        break
    print(row)
```

Úloha 2: Filtrovanie dát a vizualizácia

Z tabuľky `processed_rows` vyfiltrujeme riadky pre krajiny identifikované v zozname `countries`. Výsledkom je tabuľka `filtered_processed_rows`.

```
countries = ['SK', 'CZ']
filtered_processed_rows = processed_rows[:1] + list(filter(lambda row: row[0] == 'PAS_BRD' and row[1] in countries, processed_rows))
for i, row in enumerate(filtered_processed_rows):
    if i >= 5:
        break
    print(row)
```

Vizualizujeme údaje pre dané krajiny z prefiltrovanej tabuľky v čiarovom grafe.

```
import matplotlib.pyplot as plt

# Extrahovanie hlavičiek časových období (od tretieho stĺpca ďalej)
time_periods = processed_rows[0][2:]

# Predspracovanie časových období, odstránenie medzier
time_periods = [tp.strip() for tp in time_periods]

# Zber dát pre graf
plot_data = {}
for row in filtered_processed_rows[1:]:
    geo_id = row[1]
    values = row[2:]
    plot_data[geo_id] = [v for v in values]
```

```

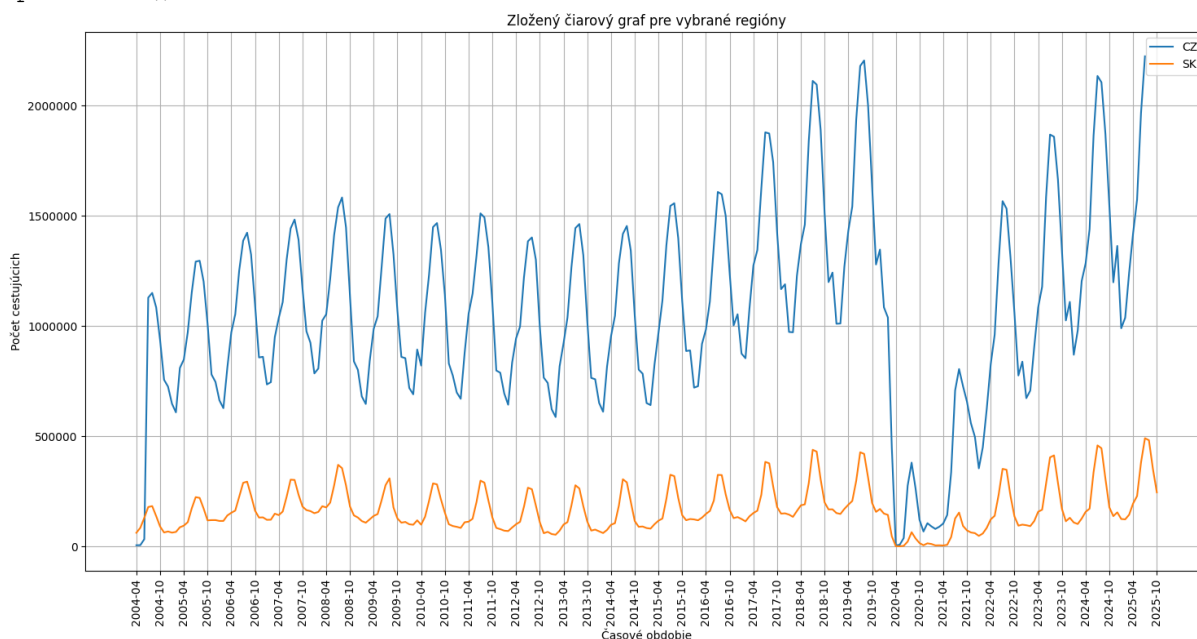
# Vytvorenie grafu
plt.figure(figsize=(15, 8))
max_y = 0
for geo_id, values in plot_data.items():
    # Filtrujeme None hodnoty pre daný rad, aby graf nepukol
    filtered_time_periods = []
    filtered_values = []
    for i, val in enumerate(values):
        if val is not None:
            filtered_time_periods.append(time_periods[i])
            filtered_values.append(val)
    max_y = max(max_y, max(filtered_values))
    plt.plot(filtered_time_periods, filtered_values, label=geo_id)

plt.xlabel('Časové obdobie')
plt.ylabel('Počet cestujúcich')
plt.title('Zložený čiarový graf pre vybrané regióny')

# Zobrazenie iba každého 12. popisku na osi X
plt.xticks(ticks=filtered_time_periods[::6], labels=filtered_time_periods[
::6], rotation=90) # Otočenie popiskov osi X pre lepšiu čitateľnosť
plt.yticks(ticks=list(range(0,max_y,500000)), labels=list(range(0,max_y,50
0000)),rotation=0)

plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

```



Obrázok 5: Vývoj intenzity leteckej dopravy v Českej a Slovenskej republike

Úloha 3: Minimálne a maximálne počty cestujúcich

Vypočítame minimálne a maximálne počty cestujúcich za celé obdobie pre každý región a mesiac, v ktorom boli dosiahnuté. Výsledky zobrazíme v tabuľke.

```
header = processed_rows[0]
zaciatok_pandemie = header.index('2019-12 ')
koniec_pandemie = header.index('2022-12 ')
min_max_values = {}
for row_index in range(1, len(processed_rows)):
    geo_id = processed_rows[row_index][1]

    values_with_headers = []
    for idx, value in enumerate(processed_rows[row_index][2:]):
        if value is not None:
            # Ensure header has enough elements to avoid IndexError for header[idx+2]
            if (idx + 2) < len(header):
                values_with_headers.append((value, header[idx+2]))

    if values_with_headers:
        min_val_tuple = min(values_with_headers)
        max_val_tuple = max(values_with_headers)
        min_max_values[geo_id] = (min_val_tuple, max_val_tuple)
    else:
        min_max_values[geo_id] = (None, None)

for key, (min_val, max_val) in min_max_values.items():
    if min_val is not None and max_val is not None:
        print(f'{key:10} minimum      {min_val[0]:10d} ({min_val[1]}) maximum {max_val[0]:10d} ({max_val[1]})')
    else:
        print(f'{key:10} No data available.')
```

AT	minimum	4705	(2020-04)	maximum	1855180	(2025-08)
BA	minimum	8906	(2021-04)	maximum	147199	(2024-08)
BE	minimum	6483	(2020-04)	maximum	1973279	(2019-07)
BG	minimum	13449	(2020-04)	maximum	1077964	(2018-08)
CH	minimum	12118	(2020-04)	maximum	3088345	(2025-07)
CY	minimum	2439	(2020-04)	maximum	908265	(2025-08)
CZ	minimum	1276	(2004-04)	maximum	1097593	(2025-07)
DE	minimum	103369	(2020-04)	maximum	12818453	(2019-07)
DK	minimum	12609	(2003-02)	maximum	1961664	(2024-07)
EE	minimum	1429	(2020-04)	maximum	179971	(2024-08)
EL	minimum	21195	(2020-04)	maximum	6270465	(2024-08)
ES	minimum	65830	(2020-04)	maximum	16842723	(2025-08)
FI	minimum	10533	(2020-04)	maximum	1189955	(2019-06)
FR	minimum	70034	(2020-04)	maximum	10659247	(2019-07)
HR	minimum	2088	(2020-04)	maximum	1209379	(2025-08)
HU	minimum	3603	(2020-04)	maximum	970002	(2025-08)
IE	minimum	12847	(2020-04)	maximum	2249291	(2025-07)
IS	minimum	3386	(2020-04)	maximum	623703	(2018-08)
IT	minimum	47665	(2020-04)	maximum	11825454	(2024-08)

LT	minimum	397	(2020-04)	maximum	360720	(2025-08)
LU	minimum	6	(2020-04)	maximum	279259	(2025-07)
LV	minimum	1381	(2020-04)	maximum	423188	(2019-08)
ME	minimum	423	(2020-05)	maximum	246936	(2019-08)
MK	minimum	24	(2020-06)	maximum	187583	(2024-08)
MT	minimum	1710	(2020-04)	maximum	503001	(2024-08)
NL	minimum	59089	(2020-04)	maximum	4097902	(2019-07)
NO	minimum	179758	(2020-04)	maximum	2731598	(2019-06)
PL	minimum	4828	(2020-05)	maximum	3671516	(2025-08)
PT	minimum	18808	(2020-04)	maximum	3790186	(2024-08)
RO	minimum	29621	(2020-04)	maximum	1360265	(2024-08)
RS	minimum	1579	(2020-04)	maximum	504584	(2023-08)
SE	minimum	27606	(2020-04)	maximum	2248381	(2018-06)
SI	minimum	30	(2020-04)	maximum	105685	(2019-08)
SK	minimum	309	(2020-05)	maximum	244609	(2025-07)
TR	minimum	5579152	(2023-02)	maximum	12889615	(2024-08)
UK	minimum	4708930	(1998-01)	maximum	15589801	(2019-07)

Vypočítame celkové maximum a minimum.

```

overall_min_value = float('inf')
overall_min_country = None
overall_min_time_period = None

overall_max_value = float('-inf')
overall_max_country = None
overall_max_time_period = None

for geo_id, (min_val_tuple, max_val_tuple) in min_max_values.items():
    if min_val_tuple is not None:
        current_min_value = min_val_tuple[0]
        if current_min_value < overall_min_value:
            overall_min_value = current_min_value
            overall_min_country = geo_id
            overall_min_time_period = min_val_tuple[1]

    if max_val_tuple is not None:
        current_max_value = max_val_tuple[0]
        if current_max_value > overall_max_value:
            overall_max_value = current_max_value
            overall_max_country = geo_id
            overall_max_time_period = max_val_tuple[1]

if overall_min_country:
    print(f"Krajina s najnižším počtom cestujúcich: {overall_min_country}
s {overall_min_value} cestujúcimi ({overall_min_time_period})")
else:
    print("Nie sú dostupné žiadne dáta pre zistenie najnižšieho počtu cest
ujúcich.")

if overall_max_country:
    print(f"Krajina s najvyšším počtom cestujúcich: {overall_max_country}

```

```
s {overall_max_value} cestujúcimi ({overall_max_time_period}))"
else:
    print("Nie sú dostupné žiadne dáta pre zistenie najvyššieho počtu cestujúcich.")
```

Krajina s najnižším počtom cestujúcich: LU s 6 cestujúcimi (2020-04)
 Krajina s najvyšším počtom cestujúcich: ES s 16842723 cestujúcimi (2025-08)

Úloha 4: Priemerné počty cestujúcich pred, počas a po pandémie COVID-19

Určíme priemerné počty cestujúcich pred, počas a po pandémie COVID-19 pre každý región. Výsledok je v premennej `comparison_data` typu štruktúrovaný slovník, ktorý obsahuje pre každý región tri hodnoty `avg_pre_pandemic`, `avg_pandemic`, `avg_post_pandemic`. Výsledky zobrazíme vo forme tabuľky.

```
header = processed_rows[0]

# Nájdite indexy pre začiatok a koniec predpandemického a popandemického obdobia
# '2019-12 ' ako koniec pred pandemiou (posledný mesiac pred 2020)
# '2020-01 ' ako začiatok pandémie (prvý mesiac 2020)
# '2023-01 ' ako začiatok po pandémii (prvý mesiac po 2022)

try:
    index_pre_pandemic_end = header.index('2019-12 ')
    index_pandemic_start = header.index('2020-01 ')
    index_pandemic_end = header.index('2022-12 ')
    index_post_pandemic_start = header.index('2023-01 ')
except ValueError as e:
    print(f"Chyba pri hľadaní časového obdobia v hlavičke: {e}")
    print("Uistite sa, že existujú '2019-12 ', '2020-01 ', '2022-12 ' a '2023-01 ' v hlavičke.")
    # Nastavte defaultne hodnoty, ak sa indexy nenajdu, alebo ukončite spracovanie
    index_pre_pandemic_end = -1
    index_pandemic_start = -1
    index_pandemic_end = -1
    index_post_pandemic_start = -1

comparison_data = {}

for row_index in range(1, len(processed_rows)):
    geo_id = processed_rows[row_index][1]

    pre_pandemic_values = []
    pandemic_values = []
    post_pandemic_values = []

    # Zber dát pre predpandemické obdobie (od prvých dátových stĺpcov po '
```

```

2019-12 ')
    # Dáta začínajú od indexu 2 (po 'tra_meas' a 'geo\\TIME_PERIOD')
    for col_index in range(2, index_pre_pandemic_end + 1):
        value = processed_rows[row_index][col_index]
        if value is not None:
            pre_pandemic_values.append(value)

    # Zber dát pre pandemické obdobie (od '2020-01 ' do '2022-12 ')
    for col_index in range(index_pandemic_start, index_pandemic_end + 1):
        value = processed_rows[row_index][col_index]
        if value is not None:
            pandemic_values.append(value)

    # Zber dát pre popandemické obdobie (od '2023-01 ' do konca)
    for col_index in range(index_post_pandemic_start, len(processed_rows[r
ow_index])):
        value = processed_rows[row_index][col_index]
        if value is not None:
            post_pandemic_values.append(value)

    # Výpočet priemerov
    avg_pre_pandemic = sum(pre_pandemic_values) / len(pre_pandemic_values)
    if pre_pandemic_values else None
    avg_pandemic = sum(pandemic_values) / len(pandemic_values) if pandemic
_values else None
    avg_post_pandemic = sum(post_pandemic_values) / len(post_pandemic_valu
es) if post_pandemic_values else None

    comparison_data[geo_id] = {
        "avg_pre_pandemic": avg_pre_pandemic,
        "avg_pandemic": avg_pandemic,
        "avg_post_pandemic": avg_post_pandemic
    }

print("Porovnanie priemerného počtu cestujúcich pred, počas a po pandémie:
")
for geo_id, data in comparison_data.items():
    pre = f"{data['avg_pre_pandemic']:.0f}" if data['avg_pre_pandemic'] is
not None else "N/A"
    during = f"{data['avg_pandemic']:.0f}" if data['avg_pandemic'] is not
None else "N/A"
    post = f"{data['avg_post_pandemic']:.0f}" if data['avg_post_pandemic']
is not None else "N/A"
    print(f"Región {geo_id:10}: Pred pandémiou: {pre:15}, Počas pandémie:
{during:15}, Po pandémii: {post:15}")

```

Porovnanie priemerného počtu cestujúcich pred, počas a po pandémie:		
Región AT: Pred pandémiou: 1032056	Počas pandémie: 654034	Po pandémii: 1459704
Región BA: Pred pandémiou: N/A	Počas pandémie: 56818	Po pandémii: 80479

Región BE:	Pred pandémiou:	1020717	Počas pandémie:	704447	Po pandémii:	1386749
Región BG:	Pred pandémiou:	345098	Počas pandémie:	253533	Po pandémii:	481871
Región CH:	Pred pandémiou:	1529668	Počas pandémie:	1085578	Po pandémii:	2336371
Región CY:	Pred pandémiou:	317531	Počas pandémie:	229743	Po pandémii:	520826
Región CZ:	Pred pandémiou:	549542	Počas pandémie:	279850	Po pandémii:	695877
Región DE:	Pred pandémiou:	8129265	Počas pandémie:	4237396	Po pandémii:	8668320
Región DK:	Pred pandémiou:	1134283	Počas pandémie:	675844	Po pandémii:	1449586
Región EE:	Pred pandémiou:	86050	Počas pandémie:	68284	Po pandémii:	135171
Región EL:	Pred pandémiou:	1800920	Počas pandémie:	1791406	Po pandémii:	2995974
Región ES:	Pred pandémiou:	8095343	Počas pandémie:	6033801	Po pandémii:	2580323
Región FI:	Pred pandémiou:	771061	Počas pandémie:	379887	Po pandémii:	806412
Región FR:	Pred pandémiou:	6446175	Počas pandémie:	4317865	Po pandémii:	7927799
Región HR:	Pred pandémiou:	294872	Počas pandémie:	229756	Po pandémii:	546131
Región HU:	Pred pandémiou:	390785	Počas pandémie:	293890	Po pandémii:	720154
Región IE:	Pred pandémiou:	1094099	Počas pandémie:	692156	Po pandémii:	1710076
Región IS:	Pred pandémiou:	175419	Počas pandémie:	150331	Po pandémii:	363972
Región IT:	Pred pandémiou:	5404802	Počas pandémie:	4138374	Po pandémii:	8722345
Región LT:	Pred pandémiou:	145438	Počas pandémie:	133284	Po pandémii:	272759
Región LU:	Pred pandémiou:	99124	Počas pandémie:	104232	Po pandémii:	212048
Región LV:	Pred pandémiou:	187422	Počas pandémie:	134918	Po pandémii:	284152
Región ME:	Pred pandémiou:	95588	Počas pandémie:	51871	Po pandémii:	107806
Región MK:	Pred pandémiou:	79942	Počas pandémie:	57076	Po pandémii:	130945
Región MT:	Pred pandémiou:	164670	Počas pandémie:	140901	Po pandémii:	354181
Región NL:	Pred pandémiou:	2298339	Počas pandémie:	1580118	Po pandémii:	3103412
Región NO:	Pred pandémiou:	1813467	Počas pandémie:	1238151	Po pandémii:	2184789
Región PL:	Pred pandémiou:	1038584	Počas pandémie:	1043602	Po pandémii:	2422155
Región PT:	Pred pandémiou:	1453356	Počas pandémie:	1390194	Po pandémii:	2884940
Región RO:	Pred pandémiou:	491972	Počas pandémie:	539168	Po pandémii:	1029428
Región RS:	Pred pandémiou:	205841	Počas pandémie:	156123	Po pandémii:	353412
Región SE:	Pred pandémiou:	1505764	Počas pandémie:	735853	Po pandémii:	1412049
Región SI:	Pred pandémiou:	59059	Počas pandémie:	23204	Po pandémii:	59260
Región SK:	Pred pandémiou:	87336	Počas pandémie:	43092	Po pandémii:	116202
Región TR:	Pred pandémiou:	N/A	Počas pandémie:	N/A	Po pandémii:	9137826
Región UK:	Pred pandémiou:	9626039	Počas pandémie:	9276844	Po pandémii:	N/A

Úloha 5: Vizualizácia a porovnanie

Vizualizujeme dáta v stĺpcovom grafe. Graf zobrazí priemerné počty cestujúcich pre každý región pred pandémiou, počas pandémie a po pandémii, čo umožní ľahké vizuálne posúdenie zmien.

```
import matplotlib.pyplot as plt
import numpy as np

regions = []
avg_pre = []
avg_during = []
avg_post = []

for geo_id, data in comparison_data.items():
    # Filtrujeme N/A data pre jednoduchšiu vizualizáciu, alebo premeníme n
    a 0
    pre_val = data['avg_pre_pandemic'] if data['avg_pre_pandemic'] is not
    None else 0
    during_val = data['avg_pandemic'] if data['avg_pandemic'] is not None
    else 0
    post_val = data['avg_post_pandemic'] if data['avg_post_pandemic'] is n
    ot None else 0
```

```

regions.append(geo_id)
avg_pre.append(pre_val)
avg_during.append(during_val)
avg_post.append(post_val)

x = np.arange(len(regions)) # Pozície pre skupiny stĺpcov
width = 0.25 # Šírka jedného stĺpca

fig, ax = plt.subplots(figsize=(20, 10))

rects1 = ax.bar(x - width, avg_pre, width, label='Pred pandémie', color='skyblue')
rects2 = ax.bar(x, avg_during, width, label='Počas pandémie', color='lightcoral')
rects3 = ax.bar(x + width, avg_post, width, label='Po pandémii', color='lightgreen')

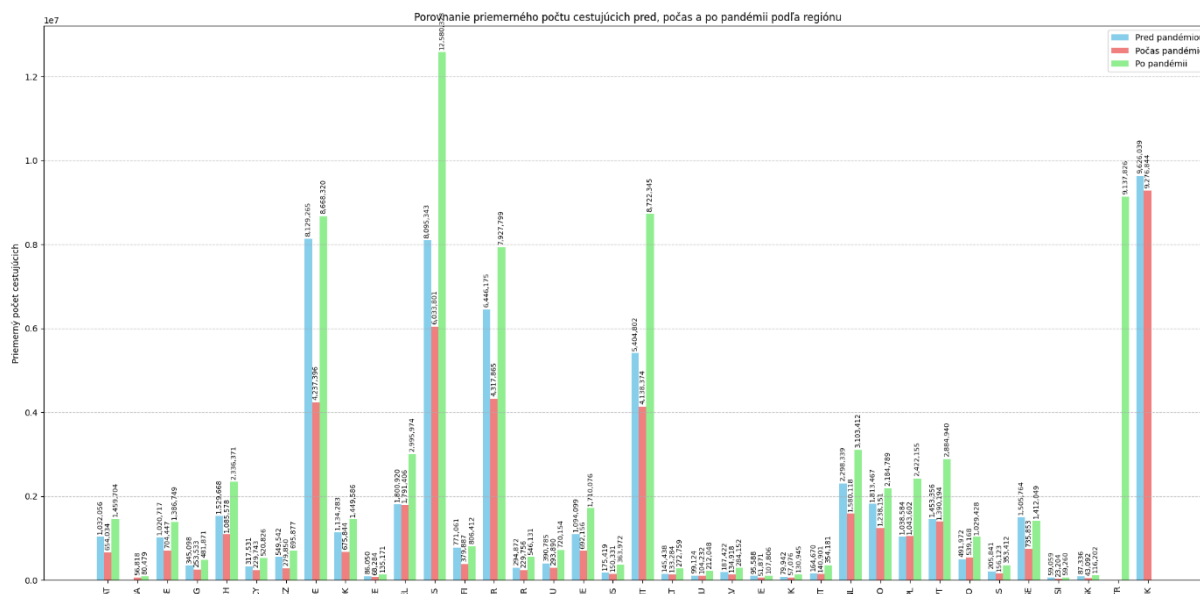
# Pridanie textov, titulkov a popiskov
ax.set_ylabel('Priemerný počet cestujúcich')
ax.set_title('Porovnanie priemerného počtu cestujúcich pred, počas a po pandémii podľa regiónu')
ax.set_xticks(x)
ax.set_xticklabels(regions, rotation=90)
ax.legend()
ax.grid(axis='y', linestyle='--', alpha=0.7)

# Funkcia pre pridanie popiskov na stĺpce
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        if height > 0: # Zobraz len, ak je hodnota > 0
            ax.annotate(f'{height:,.0f}',
                        xy=(rect.get_x() + rect.get_width() / 2, height),
                        xytext=(0, 3), # 3 body vertikálny offset
                        textcoords="offset points",
                        ha='center', va='bottom', fontsize=8, rotation=90)

# Zavolanie funkcie pre popiskovanie stĺpcov
autolabel(rects1)
autolabel(rects2)
autolabel(rects3)

plt.tight_layout()
plt.show()

```



Obrázok 6: Stĺpcový graf – Priemerné počty cestujúcich pred, počas a po pandémii

Zhrnutie výsledkov

- Ako sa celkovo menil počet cestujúcich leteckou dopravou v Európe počas rokov 2013-2024?

Počet cestujúcich sa periodicky mení počas roka vo všetkých sledovaných regiónoch - v lete rastie, v zime klesá. Globálne počet cestujúcich rastie.

- Ktoré krajiny zaznamenali najvyšší a najnižší počet cestujúcich?

Najnižší počet cestujúcich bol zaznamenaný počas pandémie vo väčšine krajín v mesiaci 2020-04. Medzi krajiny s najnižším zaznamenaným počtom cestujúcich za mesiac patria Luxembursko (LU - 6), Severné Macedónsko (MK - 24), Slovinsko (SI - 30), Slovensko (SK - 309). Najvyšší počet cestujúcich v jednotlivých krajinách sa vyskytuje v rôznych obdobiach mimo pandémie. Medzi krajiny s najvyšším počtom zaznamenaných cestujúcich za mesiac patria Španielsko (ES - 16842723), Spojené kráľovstvo (UK - 15589801) - dáta do roku 2020, Turecko (TR - 12889615) a Nemecko (DE - 12818453).

- Ako ovplyvnila pandémia COVID-19 leteckú dopravu (obdobie 2020-2022)?

Letecká doprava počas pandémie prudko klesla v porovnaní s predchádzajúcim obdobím vo všetkých krajinách, pre ktoré existujú dáta. Priemerný počet cestujúcich pred pandémiou bol tiež vo väčšine krajín vyšší ako počas pandémie. Výnimkou sú krajiny ako Rumunsko (RU) a Poľsko (PL), kde je priemer pred pandémiou nižší. Súvisí to s dynamickým rastom leteckej dopravy pred pandémiou v týchto krajinách, ktorý sa prejavil nižším predpandemickým priemerom.

- Došlo po roku 2022 k obnoveniu leteckej dopravy na úroveň pred pandémiou?

Áno, priemerný mesačný počet cestujúcich leteckou dopravou sa vo všetkých krajinách, pre ktoré existujú dáta, dostal na vyššiu úroveň ako pred pandémiou. Napríklad DE (Nemecko) malo pred pandémiou priemer 8 129 265 cestujúcich, počas pandémie klesol na 4 237 396 a po pandémii sa opäť zvýšil na 8 668 320. ES (Španielsko) malo pred pandémiou priemer 8 095 343 cestujúcich, počas pandémie klesol na 6 033 801 a po pandémii vzrástol na 12 580 323. SK (Slovensko) malo pred pandémiou priemer 87 336 cestujúcich, počas pandémie klesol na 43 092 a po pandémii vzrástol na 116 022. Pre niektoré regióny, ako napríklad BA (Bosna a Hercegovina) a TR (Turecko), neboli k dispozícii kompletne dáta pre všetky obdobia.

Záver

Workshop prispel k formovaniu dátovej gramotnosti učiteľov informatiky. V expozičnej časti workshopu sa účastníci naučili získavať súbory otvorených dát z portálov na internete, spoznali štandardné textové formáty štruktúrovaných dát CSV, TSV, JSON, XML, naučili sa ako deserializovať štruktúrované dáta z textových formátov a serializovať štruktúrované dáta do štandardných textových formátov s využitím modulov jazyka Python.

V praktickej časti sa účastníci zoznámili s prostredím Google Colab, v dvoch praktických cvičeniach o spracovaní autentických dát v jednom menšom a jednom rozsiahlom textovom súbore využili nástroje umelej inteligencie na generovanie programového kódu, kód analyzovali, upravovali, interpretovali výsledky.

Získané vedomosti a skúsenosti prispeli k rozvoju kritického myslenia účastníkov workshopu. Kritické myslenie je kľúčovým mechanizmom ochrany pred dezinformáciami, manipuláciami a propagandou. Účastníci workshopu tak absolvovali ukážku toho, ako môže vyučovanie programovania prispievať k rozvoju kritického myslenia žiakov, a tým k ich bezpečnosti v digitálnom svete.

Reflexia skúseností z workshopu:

Používanie umelej inteligencie na generovanie programového kódu je z didaktického hľadiska náročná metóda. Výsledný kód závisí od schopnosti používateľa dobre formulovať zadanie. Priebeh workshopu/vyučovacej hodiny je nedeterministický a vopred nepredvídateľný, vyžaduje od lektora dôslednú prípravu a individuálny prístup.

Didaktické odporúčania pre školskú prax:

- Umelú inteligenciu používať na prezenčnom vyučovaní len na riešenie čiastkových problémov, napríklad na grafickú vizualizáciu údajov, alebo pri samostatnom riešení projektových zadaní.
- Pri práci s autentickými dátami formulovať problém (najlepšie písomne) vo forme otázok, výsledky kriticky hodnotiť a interpretovať formou odpovedí na položené otázky (najlepšie písomne).
- Prostredie Google Colab poskytuje nástroje na kombinovanie textu a programového kódu, ako aj nástroje na generovanie kódu a konzultácie s umelou inteligenciou. Je preto vhodné pre riešenie programátorských úloh s dôrazom na rozvoj kritického myslenia (formulovať problém, rozložiť na podproblémy, riešiť problém, analyzovať/upravovať riešenia, rozhodovať, hodnotiť/interpretovať výsledky).

Tento dokument bol vypracovaný pre projekt Nitrianske kompetenčné centrum pre kybernetickú bezpečnosť (17R05-04-V01-00003)



Nitrianske kompetenčné centrum kybernetickej bezpečnosti



Financované
Európskou úniou
NextGenerationEU

PLÁN [OBNOVY]