גל עמית: 207383027, ליטל בנימין: 315637231

ניהול נתונים באינטרנט: שיעורי בית 2

- **תיאור הקוד שבונה את האונטולוגיה:**

הקוד שבונה את האונטולוגיה נמצא תחת create_ontology.py כשהוא נקרא מהקובץ geo.qa.py.

תחילה נייצר גרף חדש, שאליו נכניס את כל היחסים הבאים:

נעבור על כל המדינות (*). עבור כל מדינה נמצא תחילה את השאילתות המתאימות לאיסוף המידע בהתאם לנושאים הרלוונטיים בשאלות עצמן: נשיא, ראש ממשלה, אוכלוסייה, ממשלה, עיר בירה. נבנה את השאילתות x-path הרלוונטית לכל נושא שכזה במידה והמערך שהשאילתה מחזירה לא ריק, ונייצר ממנו ישות המתאימה לגרף האונטולוגיות. למשל:

```
name_entity = create_ontology_entity(get_name_from_url(president[0]))
```

נשים לב שחילוץ הערך המתאים מהשאילתה תלוי בדאטה. לכן עבור כל נושא תחקרנו את אופן הכתיבה שלו וכך ניקינו "ערכי זבל" (כפי שיפורט בהמשך).

בנוסף, נבנה את היחס המתאים:

```
president_entity = create_ontology_entity("president_of")
```

ולבסוף נוסיף את נושא המשפט:

```
country_entity = create_ontology_entity(country)
```

ולאחר שהגדרנו את כל הישויות בשלישיה, נוסיף אותה לגרף שבנינו קודם:

```
ontology_graph.add(
    (name_entity, president_entity, country_entity))
```

כך נפעל באופן דומה עבור כל אחד מהנושאים הנ"ל.

כמו כן, בנוסף לנושאים הקודמים נייצר עוד 2 יחסים עבור כל נשיא ועבור כל ראש ממשלה של מדינה מסוימת. לצורך כך נקפוץ (*) לעמוד הוויקיפדיה שמכיל את אותם האנשים, ונגדיר עוד 2 יחסי עזר לכל תפקיד. למשל עבור הנשיא, נגדיר: president_when_born וגם president_where_born.

לבסוף נקודד את הגרף ונשמור אותו בקובץ nt.

(*) תחילה נקרא לפונקציה ()get_country שבונה מילון שהמפתח שלו מכיל את שם המדינה והערך שלו מכיל את הכתובת הרלוונטית לעמוד הוויקיפדיה. נעבור על כל ערכי המילון בלולאה, ובכל פעם נחלץ בעזרת ()get_page (*) את עמוד הוויקיפדיה הנוכחי.

- **תיאור שאלה שהוספנו למערכת:**

שאלה שהוספנו:

List all countries whose president was born in year {}

דוגמאות:

- עבור שנת 1948 נקבל: Finland, Portugal

- עבור שנת 1986 נקבל: Chile

נשים לב שעבור השאלה הזו המידע כבר נמצא בגרף ולכן אין צורך להוסיף עוד יחסים בחילוץ המידע.

כדי לחלץ את המידע מהגרף נגדיר בה את היחס president_when_born, כאשר שם המדינה לא מעניין אותנו, ונפלטר את התאריך כך שנדרוש שהסטרינג שלו יכיל את השנה הרלוונטית.

```
if re.search("list", question):
    if re.search("capital", question):
        name = (question.split("?")[0]).split(" ")
        name = "_".join(name[9:])
        q = "select ?country where { ?capital" + " <" + ONTOLOGY_PREFIX + "capital_of> ?country. " + " FILTER(regex(lcase(str(?capital)),"+ "\"" + \
            name.lower() + "\"" + "))}"
    elif re.search("year", question):
        name = (question.split("?")[0]).split(" ")
        name = "_".join(name[9:])
        q = "select ?country where { ?date" + " <" + ONTOLOGY_PREFIX + "president_when_born> ?country. " + " FILTER(regex(lcase(str(?date)),"+ "\"" + \
            name + "-\"" + "))}"
```

- **<u>שלושה מקרי קצה באיסוף המידע:</u>**

1) עבור הנושא population בחלק מדפים הגענו אליהם דרך שאילתה שמכילה census ובחלק לפי שאילתה שמכילה estimate. בגלל השונות הזו בין הדפים השונים, תחילה בדקנו עבור census:

```
population = country_infobox.xpath("//tr[th[contains(., 'census')]]/td//text()")
```

ואם הוחזר מערך ריק, אז בדקנו עבור estimate:

```
if len(population) == 0:
    population = country_infobox.xpath("//tr[th[*[contains(., 'estimate')]]]/td//text()")
    population = [x for x in population if re.search("(?=.*\d+)(?=. *[,].*)", str(x))]
```

כאשר נעזרנו ב-regex על מנת להוציא מהמערך המוחזר של שאילתת ה- x-path "ערכי זבל", ולהישאר רק עם הסטרינג המתאים.

2) עבור הנושא area תחקרנו את המחרוזות המתאימות, וראינו שהן מורכבות מהמספר ועוד ערך שאינו ניתן לקידוד nt שהוא במקור מהווה תו שמסמן את ערך השטח בריבוע. לאחר בדיקה של המחרוזות הנ"ל, ראינו כי ישנה מחרוזת שמפרידה בין ערכי הזבל הללו לבין המספר שאותו אנחנו רוצים. לכן עשינו ספליט עבור אותה המחרוזת וקיבלנו רק את המספר בעל הקידוד המתאים:

```
if len(area) > 0:
    area = area[0]
    area = (area.split(' '))[0]
    area = (area.split("\xa0"))
    area = "_".join(area)
```

3) גם עבור השאילתות שמכילות את המידע על המקומות שנשיאים/ראשי ממשלה נולדו קיבלנו ערכי זבל שמופיעים במקומות שונים ברשימה המוחזרת. לכן גם אותם ניקינו בהתאם על ידי ביטויי regex, למשל:

```
if len(prime_where_born) > 0:
    prime_country = [x for x in prime_where_born if re.search('[a-zA-Z]', str(x))][-1]
    prime_country = "_".join(re.findall("[a-zA-Z]+", prime_country))
```

כלומר חיפשנו מבין כל הערכים במערך את אלו שמכילים רק מילה במצופה משם של מדינה.