შეხვედრა 5: მონაცემთა სტრუქტურები (ლექსიკონები) – ვქმნით ბოტის ტვინს

შენ უკვე იცი სიების გამოყენება და ბევრი ინფორმაციის ერთ ცვლადში შენახვა. დღეს კი უფრო მოწესრიგებულ და მოხერხებულ მონაცემთა სტრუქტურას **ლექსიკონს (Dictionary)** გავეცნობით, რომელიც ჩვენი ჩატბოტის „ტვინად“ იქცევა. ამ შეხვედრის ბოლოს შენ შეძლებ, მომხმარებლის შეტყობინებას დაუკავშირო კონკრეტული პასუხი. ეს შენს ბოტს უფრო „ჭკვიანს“ და პროგნოზირებადს გახდის. შენი პროგრამა აღარ გასცემს შემთხვევით პასუხებს. ის მიხვდება, თუ რა უთხრა მომხმარებელმა და შესაბამისად უპასუხებს. ეს არის ჩატბოტების ფუნდამენტური პრინციპი, რომლის გამოყენებასაც შენ თავად შეძლებ.

1. მონაცემთა სტრუქტურები: ლექსიკონები (Dictionaries)

ლექსიკონები Python-ში ერთ-ერთი ყველაზე მძლავრი და ხშირად გამოყენებადი მონაცემთა სტრუქტურაა. ისინი საშუალებას გვაძლევენ, შევინახოთ ინფორმაცია წყვილების სახით, სადაც თითოეულ მნიშვნელობას მინიჭებული აქვს თავისი უნიკალური სახელი.

1.1. ლექსიკონი, როგორც წყვილების საცავი: გასაღები-მნიშვნელობა

ლექსიკონი ჰგავს ენციკლოპედიას ან ტელეფონის წიგნაკს, სადაც ინფორმაცია მოწესრიგებულად არის განლაგებული. სიისგან განსხვავებით, აქ ყოველ მონაცემს თავისი „სათაური“, ანუ **გასაღები (Key)** აქვს. თითოეული ელემენტი შედგება ორი ნაწილისგან: გასაღები (Key) და მნიშვნელობა (Value).

ანალოგია 1: ტელეფონის წიგნაკი

წარმოიდგინე, რომ ტელეფონის წიგნაკში ინახავ მეგობრების შესახებ საკონტაქტო ინფორმაციას. ყოველი კონტაქტი შედგება სახელისა და ნომრისგან. იმისათვის, რომ იპოვო მეგობრის ნომერი, შენ ეძებ მის სახელს. ამ შემთხვევაში, სახელი არის **გასაღები (Key)**, ხოლო ტელეფონის ნომერი - **მნიშვნელობა (Value)**. შენი ტელეფონის წიგნაკი არის ლექსიკონის კლასიკური მაგალითი, სადაც გასაღები გეხმარება მარტივად და სწრაფად მიწვდე სასურველ მნიშვნელობას. პროგრამირებაშიც ზუსტად იგივე პრინციპი მუშაობს:

- **გასაღები (Key):** მეგობრის სახელი (მაგ. `"ლუკა"`)
- **მნიშვნელობა (Value):** მისი ტელეფონის ნომერი (მაგ. `"555-123-456"`)

ანალოგია 2: განმარტებითი ლექსიკონი

განმარტებით ლექსიკონში ინფორმაცია ანბანის მიხედვით არის დალაგებული. კონკრეტული სიტყვის მნიშვნელობის გასაგებად, ჯერ თავად სიტყვას ვეძებთ. ამ შემთხვევაში, სიტყვა არის **გასაღები**, ხოლო მისი განმარტება - **მნიშვნელობა**. ასეთი სტრუქტურა ინფორმაციის ორგანიზებასა და მოძიებას ძალიან აადვილებს.

- **გასაღები (Key):** სიტყვა (მაგ. `"კომპიუტერი"`)
- **მნიშვნელობა (Value):** ამ სიტყვის განმარტება („ელექტრონული გამომთვლელი მანქანა...“)

პროგრამირებაში ლექსიკონს ფიგურული ფრჩხილებით `{}` ვქმნით და გასაღები-მნიშვნელობის წყვილებს ვათავსებთ შიგნით, ორწერტილით (`:`) გამოყოფილს.

```
# ტელეფონის წიგნაკის შექმნა
phone_book = {
    "ლუკა": "555-123-456",
    "ანა": "555-987-654",
    "ნინო": "555-111-222"
}
```

1.2. ლექსიკონის შექმნა, ახალი წყვილის დამატება და მნიშვნელობაზე წვდომა გასაღების გამოყენებით

ლექსიკონის შექმნის შემდეგ შეგვიძლია მასში ცვლილებები შევიტანოთ.

ახალი წყვილის დამატება: ახალი ელემენტის დასამატებლად, უბრალოდ ვიყენებთ კვადრატულ ფრჩხილებს და ვუთითებთ ახალ გასაღებსა და მის მნიშვნელობას.

```
# ახალი კონტაქტის დამატება
phone_book["დავითი"] = "555-444-333"

print(phone_book)
# შედეგი: {'ლუკა': '555-123-456', 'ანა': '555-987-654', 'ნინო':
'555-111-222', 'დავითი': '555-444-333'}
```

მნიშვნელობაზე წვდომა: მნიშვნელობაზე წვდომისთვის ვიყენებთ კვადრატულ ფრჩხილებს და ვუთითებთ სასურველ გასაღებს.

```
# დავითის ნომერზე წვდომა
print(phone_book["დავითი"]) # შედეგი: 555-444-333
```

მნიშვნელოვანია: თუ ისეთ გასაღებს მიუთითებ, რომელიც ლექსიკონში არ არსებობს, პროგრამა შეცდომას გამოიტანს!

2. ჩატბოტის მოდელი: მიზანი-პასუხი – ვქმნით ბოტის „აზროვნების“ საფუძვლებს

ლექსიკონები იდეალურია ჩვენი ჩატბოტის „ტვინის“ ასაგებად, რადგან მისი მუშაობის ძირითადი პრინციპი წყვილებისგან შედგება. ამ მოდელში ჩვენ ვქმნით **„მიზეზ-შედეგობრივ“ კავშირს** მომხმარებლის მიერ გამოხატულ მიზანსა და ბოტის პასუხს შორის. ეს არის ყველაზე მარტივი, მაგრამ ყველაზე მნიშვნელოვანი ნაბიჯი, რათა ბოტმა ადამიანთან „გონივრული“ დიალოგის წარმართვა შეძლოს. ის შემთხვევითობის პრინციპით კი არ მოქმედებს, არამედ ირჩევს ოპტიმალურ პასუხს. პასუხს, რომელიც ლოგიკურად არის დაკავშირებული მომხმარებლის შეტყობინებასთან.

2.1. მიზანი (Intent): მომხმარებლის განზრახვა

როდესაც მომხმარებელი რამეს გვეუბნება, მნიშვნელოვანია გავიაზროთ რა არის მისი ნამდვილი მიზანი - რისი გაკეთება ან მიღება სურს. ეს გვეხმარება სწორი პასუხის გაცემაში.

მარტივი მაგალითები:

- მომხმარებელი წერს „გამარჯობა" → მისი მიზანი (Intent): მისალმება
- მომხმარებელი წერს „ნახვამდის" → მისი მიზანი (Intent): დამშვიდობება
- მომხმარებელი წერს „რა ამინდია?" → მისი მიზანი (Intent): ამინდის შესახებ ინფორმაციის მიღება

სწორედ ამ მიზნის (Intent) ამოცნობის შემდეგ შეგვიძლია შესაბამისი და სწორი პასუხი გავცეთ მომხმარებელს.

2.2. პასუხი (Response): ბოტის რეაქცია

ეს არის ბოტის შესაბამისი პასუხი, რომელიც ჩვენ მიერ განზრახვის (Intent) ამოცნობის შედეგად დაბრუნდება. თუ მომხმარებლის მიზანი იყო მისალმება, ბოტის პასუხი (Response) შეიძლება იყოს „გამარჯობა! რით შემიძლია დაგეხმაროთ?“. პასუხი არის ის, რასაც ბოტი ეტყვის მომხმარებელს მისი მიზნის გათვალისწინებით.

2.3. ლექსიკონის გამოყენება ამ მოდელის ასაგებად:

ჩვენ შეგვიძლია შევქმნათ ლექსიკონი, სადაც გასაღები იქნება მომხმარებლის მიზანი, ხოლო მნიშვნელობა - ბოტის პასუხი.

```
# ჩატბოტის პასუხების ლექსიკონი
bot_responses = {
    "გამარჯობა": "გამარჯობა! რით შემიძლია დაგეხმაროთ?",
    "როგორ ხარ?": "მადლობა, კარგად!",
    "ნახვამდის": "ნახვამდის! წარმატებულ დღეს გისურვებ."
}
```

3. ლექსიკონისა და if პირობის კომბინირება

როდესაც მომხმარებელი პასუხს ჩაწერს, ჩვენ უნდა შევამოწმოთ, არსებობს თუ არა მისი შეტყობინება ჩვენი bot_responses ლექსიკონის გასაღებებს შორის.

3.1. get() მეთოდი უსაფრთხო პასუხის მისაღებად

ჩვეულებრივი წვდომისგან განსხვავებით, get() მეთოდი უსაფრთხოა. თუ მის მიერ მითითებული გასაღები არ მოიძებნა ლექსიკონში, პროგრამა შეცდომის ნაცვლად დააბრუნებს None-ს (ან ნებისმიერ სხვა მნიშვნელობას, რომელსაც მეორე არგუმენტად გადავცემთ).

მაგალითად:

```
feeling = "კარგად ვარ"
response = bot_responses.get(feeling)
print(response) # შედეგი: None
```

თუ გვინდა, რომ პროგრამამ შეცდომის ნაცვლად კონკრეტული ტექსტი დააბრუნოს, შეგვიძლია მეორე არგუმენტიც გადავცეთ:

```
feeling = "კარგად ვარ"
# თუ გასაღები ვერ მოიძებნა, დააბრუნე "ვერ გავიგე"
response = bot_responses.get(feeling, "ვერ გავიგე. სცადე თავიდან.")
print(response) # შედეგი იქნება: "ვერ გავიგე. სცადე თავიდან."
```

პრაქტიკული დავალება 5: ლექსიკონის პრინციპით შექმნილი ბოტი

შექმენი ლექსიკონი, სადაც გასაღები იქნება მომხმარებლის სავარაუდო შეტყობინება ("გამარჯობა", "როგორ ხარ", "ნახვამდის"), ხოლო მნიშვნელობა - ბოტის პასუხი. დაწერე კოდი, რომელიც მომხმარებლის შეტყობინებას მიიღებს, შეამოწმებს არსებობს თუ არა ის ლექსიკონში და შესაბამის პასუხს დაბეჭდავს. თუ მომხმარებლის შეტყობინება ლექსიკონში არ არის, დაბეჭდე ნაგულისხმევი პასუხი.

დავალება 5.1: შეცდომის პოვნა და გასწორება

მოცემულ კოდში დაშვებულია შეცდომა. შენი ამოცანაა იპოვო და გაასწორო ის, რათა პროგრამამ პირობის შესაბამისად იმუშაოს.

```
responses = {
    "გამარჯობა": "გამარჯობა!",
    "როგორ ხარ": "მადლობა, კარგად.",
    "ნახვამდის": "ნახვამდის!"
}

user_message = input("დაწერე რამე: ")

print(responses.get(user_message))
```

დავალება 5.2: კოდის დასრულება
მოცემულ კოდს აკლია ერთი ან რამდენიმე სტრიქონი. დაამატე მხოლოდ ის, რაც აუცილებელია, რომ პროგრამამ გამართულად იმუშაოს.

```
responses = {
    "გამარჯობა": "გამარჯობა!",
    "როგორ ხარ": "მადლობა, კარგად.",
    "ნახვამდის": "ნახვამდის!"
}

user_message = input("დაწერე რამე: ")

# იპოვე შესაბამისი პასუხი აქ

print(response)
```

დავალება 5.3: კოდის დაწერა ნულიდან
დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც პასუხებისთვის შექმნის ლექსიკონს, მომხმარებლისგან მიიღებს შეტყობინებას და დაბეჭდავს შესაბამის პასუხს, ნაგულისხმევი პასუხის ჩათვლით.

```
# აქ დაწერე შენი კოდი:
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
# ამ კოდის საშუალებით შენი ნამუშევრის გადამოწმება შეგიძლია
# შექმენი ლექსიკონი პასუხებისთვის
responses = {
    "გამარჯობა": "გამარჯობა!",
    "როგორ ხარ": "მადლობა, კარგად.",
    "ნახვამდის": "ნახვამდის!"
}

# მიიღე შეტყობინება მომხმარებლისგან
user_message = input("დაწერე რამე: ")

# იპოვე შესაბამისი პასუხი ლექსიკონში. თუ ვერ იპოვე, დააბრუნე ნაგულისხმევი ტექსტი.
```

```
response = responses.get(user_message, "ვერ გავიგე, სცადე თავიდან.")

# დაბეჭდე ბოტის პასუხი
print(response)
```