შეხვედრა 4: მონაცემთა სტრუქტურები (სიები) და ციკლები – გავამრავალფეროვნოთ ბოტის პასუხები

ჩვენ უკვე ვისწავლეთ, თუ როგორ მივაღებინოთ ჩვენს პროგრამას პროგრამული კოდის საფუძველზე. გადაწყვეტილებები. ახლა კი ვისწავლით, როგორ შევინახოთ ბოტის რამდენიმე პასუხი ერთ ცვლადში. რაც მთავარია, როგორ გავიმეოროთ ერთი და იგივე მოქმედება ავტომატურად ცვლადში შენახული ინფორმაციის დამუშავებისას. მაგალითად, თუ ბოტს აქვს რამდენიმე სხვადასხვა ტიპის მისალმება, შეგვიძლია პროგრამას ვასწავლოთ, რომ ერთ-ერთი მათგანი შემთხვევითობის პრინციპით აირჩიოს. ეს დაგვეხმარება, ჩვენი ბოტის პასუხები გავხადოთ უფრო მრავალფეროვანი და საინტერესო.

1. მონაცემთა სტრუქტურები: სიები (Lists)

აქამდე ჩვენ ვინახავდით მხოლოდ ერთ ინფორმაციას ერთ ცვლადში, მაგალითად, `user_name = "ლუკა"`. მაგრამ რა მოხდება, თუ გვინდა, რომ ერთ ცვლადში შევინახოთ ჩვენი საყვარელი ფილმების მთელი სია? ასეთ დროს გამოიყენება პროგრამული ბრძანება, რომელსაც სია (**List**) ჰქვია.

1.1. ერთ ცვლადში გაერთიანებული ბევრი მნიშვნელობა

წარმოიდგინე, რომ საყიდლების სია გაქვს: "რძე", "პური", "კვერცხი". პროგრამირებაში ასეთი სიის შესაქმნელად ვიყენებთ კვადრატულ ფრჩხილებს `[]` და მასში განთავსებულ ელემენტებს მძიმით ვყოფთ. ეს ერთ დიდ გრძელ ყუთს ჰგავს, რომელსაც შიგნით რამდენიმე განყოფილება აქვს და თითოეულში სხვადასხვა ნივთია მოთავსებული.

```
# საყიდლების სიის შექმნა
shopping_list = ["რძე", "პური", "კვერცხი"]
```

ახლა `shopping_list` ცვლადში შენახულია სამივე ელემენტი.

1.2. სიის შექმნა, ელემენტების დამატება (`append`) და წაშლა (`remove`)

სიის შექმნის შემდეგ ჩვენ შეგვიძლია მასში ახალი ელემენტები ჩავამატოთ ან წავშალოთ. ამისთვის ვიყენებთ სპეციალურ ბრძანებებს, რომლებსაც **მეთოდები** ჰქვია. მეთოდები არის ფუნქციები, რომლებიც ობიექტთან არის დაკავშირებული (ამ შემთხვევაში, სიასთან) და საშუალებას გვაძლევს, კონკრეტული მოქმედება შევასრულოთ.

`append()`: ამ მეთოდის გამოყენებით სიის ბოლოში ახალი ელემენტის დამატება შეგვიძლია. მაგალითად, თუ საყიდლების სიას კიდევ ერთი პროდუქტი დაემატა.

```
# სიის შექმნა
my_list = ["ვაშლი", "ბანანი"]

# სიის ბოლოში ახალი ელემენტის დამატება
my_list.append("ფორთოხალი")

# სიის დაბეჭდვა, რომელშიც ახალი ელემენტიც იქნება
```

```
print(my_list) # შედეგი იქნება: ['ვაშლი', 'ბანანი', 'ფორთოხალი']
```

remove(): ამ მეთოდით კი საყიდლების სიიდან კონკრეტული ელემენტის წაშლა შეგვიძლია. მაგალითად, თუ გადაწყვიტე, რომ "ბანანის" ყიდვა აღარ გინდა.

```
# სიიდან ელემენტის წაშლა
my_list.remove("ბანანი")

# სიის დაბეჭდვა ცვლილების შემდეგ
print(my_list) # შედეგი იქნება: ['ვაშლი', 'ფორთოხალი']
```

1.3. სიის ელემენტზე წვდომა ინდექსის ([0], [1]) საშუალებით

სიის თითოეულ ელემენტს აქვს თავისი ნომერი, რომელსაც **ინდექსი** ჰქვია. წარმოიდგინე, რომ ხალხი რიგში დგას, ყველას თავისი რიგის ნომერი უჭირავს ხელში: პირველ ადამიანს ამ რიგში პირველი ნომერი კი არ უჭირავს ხელში, არამედ – ნომერი ნული. ზუსტად ასეა პროგრამირებაშიც. მნიშვნელოვანია დაიმახსოვრო, რომ Python-ში ინდექსირება ყოველთვის იწყება ნულიდან.

```
# სიის შექმნა
fruits = ["ვაშლი", "ბანანი", "ფორთოხალი"]

# პირველ ელემენტზე წვდომა (ინდექსი 0)
print(fruits[0]) # შედეგი: "ვაშლი"

# მესამე ელემენტზე წვდომა (ინდექსი 2)
print(fruits[2]) # შედეგი: "ფორთოხალი"
```

სავარჯიშო 1:

```
# მოცემულ სიაში დაამატე შენი საყვარელი ფერი
favorite_colors = ["ლურჯი", "მწვანე"]

# აქ დაწერე შენი კოდი:

# დაამატე კიდევ ერთი ფერი `append()` მეთოდის გამოყენებით.

print(favorite_colors)
```

2. მოქმედებების გამეორება: for ციკლი

ხშირად სიის ყველა ელემენტთან დაკავშირებით ერთი და იგივე მოქმედების გამეორება გვჭირდება. მაგალითად, თუ საყიდლების სიაში 10 პროდუქტი გვაქვს, თითოეული მათგანი სათითაოდ უნდა წავიკითხოთ. პროგრამირებაში ასეთი განმეორებადი ამოცანების შესრულების ავტომატიზაციისთვის გამოიყენება **ციკლები (Loops)**.

2.1. როგორ დავათვალიეროთ საყიდლების სიის ყველა პროდუქტი სათითაოდ

წარმოიდგინე, რომ ხელში საყიდლების სია გიჭირავს და დედა გთხოვს შეამოწმო ყველა პროდუქტი ჩადეთ თუ არა კალათაში. შენ რიგ-რიგობით კითხულობ სიის ყველა პუნქტს: "რძე... პური... კვერცხი...". ამ მოქმედებას შენ იმეორებ მანამ, სანამ სია არ დასრულდება. ეს არის ზუსტად ის, რასაც `for` ციკლი აკეთებს.

2.2. `for` ციკლის გამოყენება სიის ყველა ელემენტზე მოქმედების შესასრულებლად

`for` ციკლი საშუალებას გვაძლევს, რომ სიის ელემენტებს სათითაოდ მივწვდეთ და მასზე რაიმე მოქმედება შევასრულოთ.

```
# სიის შექმნა
shopping_list = ["რძე", "პური", "კვერცხი"]

# ციკლის დაწყება სიის თითოეული ელემენტისთვის
for item in shopping_list:
    # თითოეული ელემენტის დაბეჭდვა
    print(item)
```

ამ კოდში `for` ციკლი მუშაობს ასე:

1. **პირველად:** `item` ცვლადი იღებს სიის პირველი ელემენტის ("რძე") მნიშვნელობას.
2. **მეორედ:** `item` ცვლადი იღებს სიის მეორე ელემენტის ("პური") მნიშვნელობას.
3. **მესამედ:** `item` ცვლადი იღებს სიის მესამე ელემენტის ("კვერცხი") მნიშვნელობას.
4. როგორც კი სია დასრულდება, ციკლიც წყვეტს მუშაობას.

სავარჯიშო 2:

```
# მოცემულ სიაზე გამოიყენე for ციკლი და დაბეჭდე თითოეული პროდუქტი
# ტექსტთან ერთად, მაგალითად: "უნდა ვიყიდო: რძე".
shopping_list = ["რძე", "პური", "კვერცხი"]

# აქ დაწერე შენი კოდი:

# გამოიყენე for ციკლი და f-string.
```

3. შემთხვევითობის ელემენტი

ჩვენი ბოტი რომ უფრო ბუნებრივი და ნაკლებად პროგნოზირებადი იყოს, შეგვიძლია, მის პასუხებში შემთხვევითობის ელემენტი შევიტანოთ. ამისთვის Python-ში არსებობს random ბიბლიოთეკა.

3.1. random ბიბლიოთეკის იმპორტი

როგორც უკვე ვისაუბრეთ, ბიბლიოთეკები არის მზა ინსტრუმენტების ნაკრები. წარმოიდგინე, რომ Python-ს უზარმაზარი კარადა აქვს, რომელშიც ყველა ინსტრუმენტი ინახება. random ბიბლიოთეკის გამოსაყენებლად, ის კარადიდან უნდა ავიღოთ და ჩვენს პროგრამაში "შემოვიტანოთ" import ბრძანებით.

```
# random ბიბლიოთეკის შემოტანა
import random
```

3.2. random.choice() ფუნქცია სიიდან შემთხვევითი ელემენტის ამოსარჩევად

როდესაც ბიბლიოთეკა შემოტანილია, შეგვიძლია გამოვიყენოთ მასში არსებული ფუნქციები. random.choice() ფუნქცია იღებს სიას და აბრუნებს მის ერთ, შემთხვევით ელემენტს. ეს ფუნქცია არის ის ინსტრუმენტი, რომელსაც გამოვიყენებთ, რათა ჩვენმა ბოტმა მრავალფეროვანი პასუხები დაგვიბრუნოს.

```
# მისალმებების სიის შექმნა
greetings = ["გამარჯობა!", "სალამი!", "მოგესალმები!"]

# შემთხვევითი მისალმების არჩევა
random_greeting = random.choice(greetings)

# შემთხვევითი მისალმების დაბეჭდვა
print(random_greeting)
```

ამ კოდის გაშვების შემდეგ, ყოველ ჯერზე სხვადასხვა მისალმება დაიბეჭდება.

დავალება 4: "არაპროგნოზირებადი ბოტი"

შექმენი მისალმებების სია (მაგ. ["გამარჯობა!", "მოგესალმები!", "სალამი!"]). დაწერე კოდი, რომელიც random.choice()-ის გამოყენებით, პროგრამის ყოველი გაშვებისას შემთხვევით მისალმებას დაბეჭდავს.

დავალება 4.1: შეცდომის პოვნა და გასწორება

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

```
# შეცდომით დაწერილი კოდი
import random
```

```
greetings = ["გამარჯობა", "სალამი", "მოგესალმები"]
 random_greeting = random.choice(greetings)

print(random_greeting)
```

დავალება 4.2: კოდის დასრულება
მოცემულ პროგრამულ კოდს აკლია ერთი ან რამდენიმე სტრიქონი. დაამატე მხოლოდ ის, რაც აუცილებელია, რომ პროგრამამ გამართულად იმუშაოს.

```
import random

greetings = ["გამარჯობა!", "სალამი!", "მოგესალმები!"]

# აირჩიე შემთხვევითი მისალმება აქ

# დაბეჭდე შედეგი
print(random_greeting)
```

დავალება 4.3: კოდის დაწერა ნულიდან
დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც შექმნის მისალმებების სიას და დაბეჭდავს შემთხვევით ერთ-ერთ მათგანს.

```
# აქ დაწერე შენი კოდი:
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
# ამ კოდის საშუალებით შეგიძლია გადაამოწმო შენი ნამუშევარი

# random ბიბლიოთეკის შემოტანა
import random

# მისალმებების სიის შექმნა
greetings = ["გამარჯობა!", "სალამი!", "მოგესალმები!"]

# შემთხვევითი მისალმების არჩევა
random_greeting = random.choice(greetings)

# შემთხვევითი მისალმების დაბეჭდვა
print(random_greeting)
```