შეხვედრა 6: ფუნქციები: ვაქციოთ ჩვენი კოდი მრავალჯერადად გამოყენებად ინსტრუმენტად

წინა შეხვედრებზე ჩვენ შევქმენით ჩვენი ბოტის „ტვინი“ ლექსიკონის დახმარებით და ვისწავლეთ, როგორ გავხადოთ ის პროგნოზირებადი. დღეს კი ვისწავლით, როგორ მოწესრიგოთ ჩვენი კოდი ფუნქციების (function) დახმარებით. ფუნქცია არის მძლავრი ინსტრუმენტი, ის პროგრამულ კოდს ერთგავრ ბლოკებად ყოფს. ბლოკებად, რომლებიც მრავალჯერადად შეგვიძლია გამოვიყენოთ. ამ გაკვეთილის ბოლოს შენ შეძლებ, შენი ბოტის ლოგიკა ერთგვარ ინსტრუმენტად გადააქციო, რომელსაც მარტივად გამოიძახებ და გამოიყენებ პროგრამის ნებისმიერ ნაწილში. ამის შემდეგ, შენი ბოტის კოდი უფრო მოწესრიგებული, ადვილად წასაკითხი და ადვილად შესაცვლელი გახდება.

1. რა არის ფუნქცია?

ფუნქცია არის კოდის ბლოკი, რომელსაც კონკრეტული სახელი აქვს და რომელიც ერთ კონკრეტულ ამოცანას ასრულებს. ის კოდს უფრო ორგანიზებულს ხდის და გვეხმარება, ერთი და იგივე შინაარსის მქონე კოდი არ ვწეროთ რამდენჯერმე.

1.1. ფუნქცია, როგორც კოდის ბლოკი, რომელიც კონკრეტულ ამოცანას ასრულებს

ფუნქცია არის პროგრამირების ერთ-ერთი ძირითადი და უმნიშვნელოვანესი კონცეფცია. ის ჰგავს ერთგვარ „მინი-პროგრამას“ ან „კოდის ბლოკს“, რომელიც კონკრეტულ მოქმედებას ასრულებს. წარმოიდგინე, რომ ნამცხვრის გამოცხობა გინდა და რეცეპტში წერია: „მოამზადე ცომი“. ეს ერთი ფრაზა მოიცავს რამდენიმე ნაბიჯს (მაგ. ჩაყარე ფქვილი, დაამატე კვერცხი, ა.შ.). იმის ნაცვლად რომ რეცეპტში ყველა ნაბიჯი დეტალურად იყოს აღწერილი, საკმარისია ერთი კონკრეტული ფრაზა: „მოამზადე ცომი“. ფუნქციაც ზუსტად ამ პრინციპით მუშაობს. მას კონკრეტული სახელი აქვს და როცა ამ სახელს ვიყენებთ, პროგრამა ასრულებს კოდის ბლოკს, რომელიც ფუნქციის შიგნით არის მოცემული. ეს გვეხმარება, რომ პროგრამის კოდი იყოს მოწესრიგებული, ადვილად წასაკითხი და მრავალჯერადად გამოყენებადი.

1.2. ანალოგია: ციფრული ფოტოაპარატი

წარმოიდგინე, რომ ფოტოს გადაღება გინდა. შენ არ ფიქრობ იმაზე, თუ როგორ ასწორებს შენი ციფრული ფოტოაპარატი ფოკუსს. როგორ აფიქსირებს ციფრული სენსორი შემომავალ შუქს. როგორ გარდაქმნის შუქს ციფრულ სიგნალად და როგორ ამუშავებს პროცესორი ამ მონაცემებს იმისთვის, რომ ფოტო მიიღოს. შენ უბრალოდ აჭერ ერთ ღილაკს („გადაიღე ფოტო"). ეს ღილაკი ჩათვალე რომ არის. ფუნქცია. ეს „ფუნქცია" ყოველთვის ერთსა და იმავე რთულ პროცესს ასრულებს და შედეგად ყოველთვის ფოტოს იღებ.

1.3. def საკვანძო სიტყვა ფუნქციის შესაქმნელად

Python-ში ფუნქციის შესაქმნელად ვიყენებთ საკვანძო სიტყვას def (ინგლისური define-ის შემოკლება).

```
# ფუნქციის შექმნა, რომელიც აბრუნებს მისალმებას
def greet_user():
    print("გამარჯობა! კეთილი იყოს თქვენი მობრძანება.")
```

მნიშვნელოვანია: ფუნქციის პროგრამული კოდი, რომელიც მან უნდა შეასრულოს, აუცილებლად უნდა იყოს ოდნავ მარჯვნივ ჩაწეული (**indentation**). ეს არის Python-ის სინტაქსური წესი, რომელიც კოდს უფრო მოწესრიგებულს ხდის და ეხმარება პროგრამას გაიგოს, რომელი სტრიქონები ეკუთვნის ფუნქციას. წარმოიდგინე, რომ ხარ ინჟინერი, რომელმაც ხიდი უნდა ააშენოს. შენ შეიმუშავებ პროექტს და ამ პროექტის შესაბამისად მშენებლებს აძლევ დავალებას: „პირველ რიგში, ავაშენოთ ბურჯები, შემდეგ ბურჯებზე მოვათავსოთ ხიდის კონსტრუქცია“. ამ შემთხვევაში, ბურჯების ასაშენებლად საჭირო ყველა ნაბიჯი უნდა მოთავსდეს ბურჯების კოდის ბლოკში, რაც მკაფიოდ გამოყოფს მას სხვა ნაბიჯებისგან. indentation-იც ზუსტად ამ ფუნქციას ასრულებს Python-ში.

2. ფუნქციის პარამეტრები და დაბრუნებული მნიშვნელობა

2.1. პარამეტრი (არგუმენტი): ინფორმაცია, რომელსაც ფუნქციას გადავცემთ

ფუნქციას შეუძლია მიიღოს ინფორმაცია, რომელსაც დაამუშავებს. ამ ინფორმაციას **პარამეტრი** ან **არგუმენტი** ჰქვია.

```
# ფუნქცია, რომელიც იღებს მომხმარებლის სახელს
def greet_user_with_name(name):
    print(f"გამარჯობა, {name}!")

# ფუნქციის გამოძახება
greet_user_with_name("ლუკა")
```

ამ მაგალითში, name არის ფუნქციის პარამეტრი. ფუნქციის გამოძახებისას, name-ის მნიშვნელობად მიენიჭება "ლუკა".

2.2. return საკვანძო სიტყვა: შედეგის დაბრუნება ფუნქციიდან

ხშირად გვჭირდება, რომ ფუნქციამ გამოთვალოს რამე და დააბრუნოს შედეგი, რათა შემდეგ ეს შედეგი სხვაგან გამოვიყენოთ. ამისთვის ვიყენებთ return საკვანძო სიტყვას.

```
# ფუნქცია, რომელიც აბრუნებს ორი რიცხვის ჯამს
def add_numbers(a, b):
    return a + b

# ფუნქციის გამოძახება და შედეგის შენახვა
sum_result = add_numbers(5, 3)
```

```
print(sum_result) # შედეგი: 8
```

3. ჩატბოტის ლოგიკის ფუნქციაში მოქცევა – მოწესრიგებული და მრავალჯერადი გამოყენება

ახლა, როცა უკვე გავეცანით ფუნქციების შესახებ ინფორმაციას, დროა, ჩვენი ჩატბოტის კოდი უფრო მოწესრიგებული გავხადოთ. წინა შეხვედრაზე ჩვენ შევქმენით ბოტის ლოგიკა, რომელიც მომხმარებლის შეტყობინებას იღებდა და შესაბამის პასუხს აბრუნებდა. თუმცა, ეს კოდი პროგრამის ერთ ნაწილში იყო მოთავსებული და მისი ხელახლა გამოყენება ან შეცვლა არც ისე მარტივი იქნებოდა. ახლა ჩვენ ამ ფუნქციონალს მოვათავსებთ ერთ დიდ „ყუთში“, ანუ ფუნქციაში. ეს საშუალებას მოგვცემს, რომ ეს „ყუთი“ და მისი შიგთავსი, ანუ ბოტის ლოგიკა, გამოვიყენოთ პროგრამის ნებისმიერ ნაწილში, ახალი, დამატებითი კოდის დაწერის გარეშე.

3.1. შევქმნათ ფუნქცია `get_response(user_input)`, რომელიც მომხმარებლის ტექსტს მიიღებს

```
# ფუნქცია, რომელიც მომხმარებლის შეტყობინებას მიიღებს
def get_response(user_input):
    # აქ დავწერთ ლოგიკას
    pass # pass გამოიყენება, როცა ფუნქცია ცარიელია
```

3.2. ფუნქციის შიგნით განვათავსოთ ლექსიკონთან და if-else ლოგიკასთან დაკავშირებული კოდი

```
def get_response(user_input):
    responses = {
        "გამარჯობა": "გამარჯობა!",
        "როგორ ხარ": "მადლობა, კარგად.",
        "ნახვამდის": "ნახვამდის!"
    }

    # პასუხის პოვნა ლექსიკონში
    response = responses.get(user_input, "ვერ გავიგე. სცადე თავიდან.")
    return response
```

მნიშვნელოვანია: ფუნქციის შიგნით არსებული ცვლადები (`responses`, `response`) მხოლოდ ფუნქციის შიგნით არსებობს. ამ შეზღუდვას **ცვლადების ფარგლები (scope)** ჰქვია. წარმოიდგინე, რომ ფუნქცია არის შენი პირადი ოთახი. შენი ოთახის შიგნით არსებული ნივთები

(ცვლადები) მხოლოდ შენ გეკუთვნის და სხვებს არ შეუძლიათ მათი გამოყენება ან შეცვლა. მეგობრის ოთახში (სხვა ფუნქციაში) შეიძლება მსგავსი ნივთები (ცვლადები) იყოს. შენი და შენი მეგობრის ნივთები, მიუხედავად მათი მსგავსებისა, სხვადასხვა ოთახშია განლაგებული, ამიტომ ისინი ერთმანეთში არ აგერევათ. ეს პრინციპი პროგრამას შეცდომებისგან და უწესრიგობისგან იცავს.

3.3. ფუნქციამ უნდა დააბრუნოს (return) ბოტის საბოლოო პასუხი

ზემოთ მოყვანილ მაგალითში, return ბრძანება აბრუნებს ბოტის საბოლოო პასუხს, რომლის შემდეგ გამოყენებაც შეგვიძლია. მაგალითად პასუხი შეგვიძლია ეკრანზე გამოვსახოთ.

```
# მომხმარებლისგან ინფორმაციის მიღება
user_message = input("დაწერე რამე: ")

# ფუნქციის გამოძახება და შედეგის შენახვა
bot_reply = get_response(user_message)

# შედეგის დაბეჭდვა
print(bot_reply)
```

დავალება 6: ფუნქციონალური ბოტი

წინა დავალების კოდი გადააკეთე ისე, რომ ბოტის პასუხის გენერირების ლოგიკა მოექცეს ფუნქციაში get_bot_response(message). პროგრამამ უნდა გამოიძახოს ეს ფუნქცია და დაბეჭდოს მის მიერ დაბრუნებული პასუხი.

დავალება 6.1: შეცდომის პოვნა და გასწორება

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

```
def get_bot_response(message):
    responses = {
        "გამარჯობა": "გამარჯობა!",
        "როგორ ხარ": "მადლობა, კარგად.",
        "ნახვამდის": "ნახვამდის!"
    }
    return responses.get(message, "ვერ გავიგე.")

user_message = input("დაწერე რამე: ")
print(get_bot_response)
```

დავალება 6.2: კოდის დასრულება

მოცემულ პროგრამულ კოდს აკლია ერთი ან რამდენიმე სტრიქონი. დაამატე მხოლოდ ის, რაც აუცილებელია, რომ პროგრამამ გამართულად იმუშაოს.

```
def get_bot_response(message):
    responses = {
        "გამარჯობა": "გამარჯობა!",
        "როგორ ხარ": "მადლობა, კარგად.",
        "ნახვამდის": "ნახვამდის!"
    }
    return responses.get(message, "ვერ გავიგე.")

user_message = input("დაწერე რამე: ")
# გამოძახე ფუნქცია აქ
```

დავალება 6.3: შეიმუშავე პროგრამული კოდი
დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც მოიცავს ფუნქციას get_bot_response, მომხმარებლისგან იღებს შეტყობინებას და ბეჭდავს ფუნქციის მიერ დაბრუნებულ პასუხს.

```
# დაწერე შენი კოდი აქ:
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
# ამ კოდის საშუალებით შეგიძლია გადაამოწმო შენი ნამუშევარი

# ისეთი ფუნქციის შექმნა, რომელიც ბოტის პასუხს აბრუნებს
def get_bot_response(message):
    # ლექსიკონი, რომელიც ინახავს მომხმარებლის შეტყობინებებს და ბოტის პასუხებს
    responses = {
        "გამარჯობა": "გამარჯობა!",
        "როგორ ხარ": "მადლობა, კარგად.",
        "ნახვამდის": "ნახვამდის!"
    }

    # პასუხის მიღება ლექსიკონიდან; თუ ვერ მოიძებნა, დაბრუნდება ნაგულისხმევი ტექსტი
    return responses.get(message, "ვერ გავიგე, სცადე თავიდან.")
```

```
# მომხმარებლისგან შეტყობინების მიღება
user_message = input("დაწერე რამე: ")

# ფუნქციის გამოძახება და დაბრუნებული პასუხის დაბეჭდვა
print(get_bot_response(user_message))
```