შეხვედრა 3: გადაწყვეტილებების მიღება პროგრამული კოდით – ვასწავლოთ ბოტს პასუხის გაცემა

წინა შეხვედრებზე ჩვენ ვისწავლეთ, თუ როგორ შევქმნათ ჩვენი ბოტის „მეხსიერება“ ცვლადების სახით და როგორ „ავალაპარაკოთ“ ის print() ფუნქციის გამოყენებით. დღეს კი გადავდივართ ყველაზე საინტერესო ნაწილზე: **ჩვენს პროგრამას ვასწავლით როგორ მიიღოს გადაწყვეტილებები და როგორ მოერგოს მომხმარებლის ქცევას.**

ჩვენი მთავარი ამოცანა იქნება, შევქმნათ პროგრამა, რომელიც მომხმარებლის პასუხის მიხედვით განსხვავებულად მოიქცევა. ეს არის ჩვენი ბოტის პირველი, უმარტივესი ვერსია.

1. მოკლე მიმოხილვა: რა არის ცვლადი?

გახსოვს, წინა შეხვედრაზე ვთქვით, რომ ცვლადი არის პროგრამის ერთგვარი „მეხსიერება“? ის ჰგავს პატარა ყუთს, რომელსაც სახელი აწერია და რომელშიც შეგვიძლია ნებისმიერი ინფორმაცია შევინახოთ, მაგალითად, ტექსტი, რიცხვი, ან რაიმე სხვა. როცა ბოტს მომხმარებლის პასუხს ვეკითხებით, ამ პასუხს სწორედ ასეთ ყუთში ვინახავთ, რათა შემდეგ მიღებულ ინფორმაციასთან მუშაობა შევძლოთ.

2. მომხმარებლისგან ინფორმაციის მიღება

ბოტი რომ მომხმარებელთან დიალოგში შევიდეს, მას უნდა შეეძლოს მისგან მიღებული ინფირმაციის დამახსოვრება. ამისთვის Python-ში არსებობს სპეციალური ფუნქცია.

2.1. input() ფუნქცია: როგორ მივიღოთ მომხმარებლის პასუხი

წარმოიდგინე, რომ მეგობარს ეკითხები: „როგორ ხარ?“. მეგობარი რამდენიმე წამის შემდეგ გიბრუნებს პასუხს: „კარგად ვარ“. input() ფუნქცია შესაძლებლობას გვაძლევს ეს მოქმედებები განვახორციელოთ ჩვენს პროგრამაში. მას ეკრანზე გამოაქვს ჩვენ მიერ დაწერილი შეკითხვა, აჩერებს პროგრამას და ელოდება, სანამ მომხმარებელი პასუხს ჩაწერს და კლავიატურაზე დააჭერს Enter-ის ღილაკს.

2.2. მიღებული ინფორმაციის შენახვა ცვლადში

როგორც კი მომხმარებელი პროგრამას პასუხს გასცემს, პროგრამა ამ პასუხს დაიმახსოვრებს. პასუხის დასამახსოვრებლად, ჩვენ ის უნდა შევინახოთ ცვლადში.

```
# ამ ველში ჩაწერე კოდი, რომელიც მომხმარებელს ჰკითხავს მის სახელს
# და შემდეგ შეინახავს მას ცვლადში `user_name`.
# ბოლოს კი დაბეჭდავს მისალმებას.

user_name = input("როგორ გქვია? ")
print(f"გამარჯობა, {user_name}!")
```

მნიშვნელოვანია: input() ფუნქციით მიღებული ნებისმიერი ინფორმაცია ყოველთვის ტექსტის (string) ტიპისაა, რაც იმასნიშნავს, რომ თუ მომხმარებელი ჩაწერს, მაგალითად რიცხვ 17-ს, Python-ი მას არა რიცხვად, არამედ ტექსტად აღიქვამს.

3. პირობითი ოპერატორები: if-else

ახლა, როცა ვიცით მომხმარებლისგან ინფორმაციის მიღება, დროა ვასწავლოთ ბოტს, იმოქმედოს მიღებული პასუხის შესაბამისად.

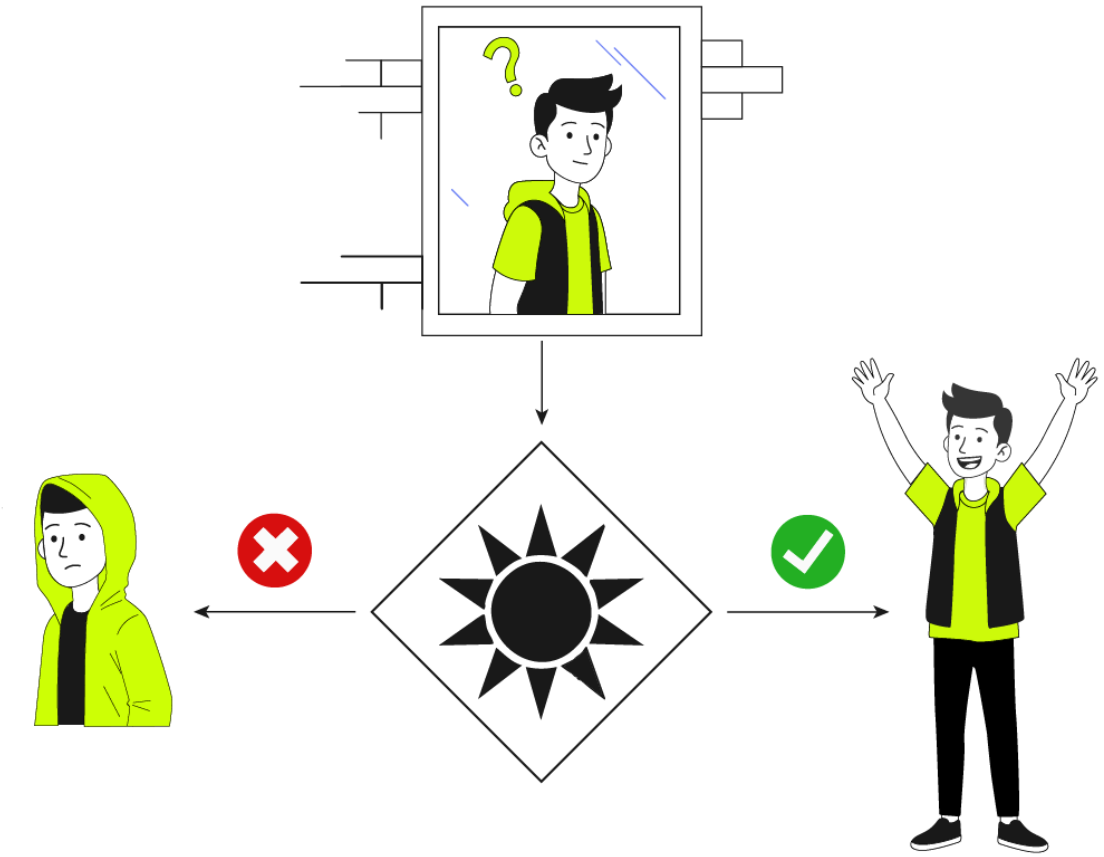
3.1. ანალოგია: გადაწყვეტილების მიღების ხელოვნება

წარმოიდგინე, დილით სახლიდან გასვლის წინ ამინდს ამოწმებ.

- **თუ** მზიანი ამინდია, **მაშინ** მაისურს და შორტს ჩაიცვამ.
- **სხვა შემთხვევაში** (თუ წვიმს ან ღრუბლიანი ამინდია), მოსაცმელს და გრძელ შარვალს ჩაიცვამ.

ეს არის პირობითი ოპერატორების გამოყენების მარტივი მაგალითი. Python-ში ასეთ ლოგიკას if და else ბრძანებებით ვქმნით.

- **if (თუ):** მოწმდება პირობა. თუ პირობა მართალია, სრულდება შესაბამისი კოდი.
- **else (სხვა შემთხვევაში):** თუ if-ის პირობა მცდარია, სრულდება else-ის ბლოკში არსებული კოდი.



3.2. შედარების ოპერატორები (==, !=, <, >): როგორ შევადაროთ მნიშვნელობები

იმისთვის, რომ პირობა შევამოწმოთ, გვჭირდება ოპერატორები, რომლებითაც მნიშვნელობებს ერთმანეთს შევადარებთ. ეს ოპერატორები პასუხად გვიბრუნებენ True-ს (ჭეშმარიტი) ან False-ს (მცდარი), რასაც შემდეგ if-ის კონსტრუქცია იყენებს.

- == (უდრის)
- != (არ უდრის)
- > (მეტია)
- < (ნაკლებია)
- >= (მეტია ან ტოლია)
- <= (ნაკლებია ან ტოლია)

მაგალითად:

```
age = 20
```

```
is_adult = age > 18 # is_adult ცვლადის მნიშვნელობა იქნება True

print(is_adult)
```

3.3. if, elif, else კონსტრუქცია ლოგიკური განშტოებების შესაქმნელად

რა მოხდება, თუ რამდენიმე პირობის შემოწმება გვინდა? ამისთვის elif (ე.ი. „else if“) გამოიყენება.

მაგალითად:

```
score = 85

if score > 90:
      print("შესანიშნავი")
elif score > 70:
      print("ძალიან კარგი")
else:
      print("კარგი")
```

ამ შემთხვევაში, რადგან 85 არ არის 90-ზე მეტი, პროგრამა გადადის შემდეგ პირობაზე (elif) და ბეჭდავს „ძალიან კარგი“.

4. მარტივი დიალოგის აგება

ახლა უკვე გვაქვს ყველა ინსტრუმენტი, რათა მარტივი დიალოგი ავაწყოთ.

4.1. ალგორითმი: კოდის დაწერამდე ალგორითმის შემუშავება

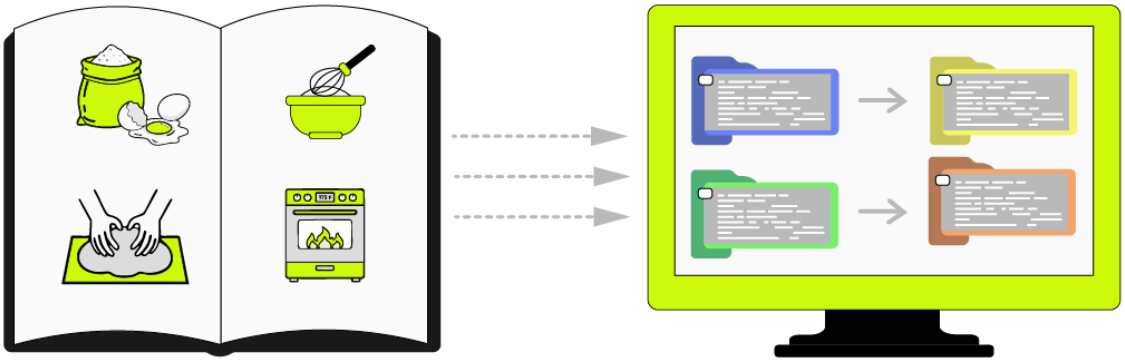
პროგრამირებაში კოდის დაწერამდე მნიშვნელოვანია გეგმის, **ალგორითმის შემუშავება**. ალგორითმის წინასწარ დაგეგმვა გვეხმარება პრობლემის ნათელ ნაბიჯებად დანახვაში, რაც უზრუნველყოფს კოდის სწრაფ და ზუსტ დაწერას. ალგორითმის შემუშავება ამცირებს შეცდომებს და საშუალებას იძლევა ოპტიმალური გადაწყვეტილებები მივიღოთ. ეს მიდგომა განსაკუთრებით მნიშვნელოვანია რთული ამოცანების შესრულებისას და გუნდურ მუშაობაში, სადაც ნათელი გეგმა წარმატების საფუძველია.

რა არის ალგორითმი?

წარმოიდგინე, რომ ნამცხვრის გამოცხობა გინდა. სანამ უშუალოდ მომზადებას დაიწყებ, მიჰყვები ინსტრუქციების ჩამონათვალს (რეცეპტს):

1. მოამზადე ინგრედიენტები.
2. აურიე ფქვილი, შაქარი და კვერცხი.
3. მოზილე ცომი.
4. გამოაცხვე ღუმელში 180 გრადუსზე 30 წუთის განმავლობაში.

ეს ინსტრუქციების თანმიმდევრობაა და მას ალგორითმი ეწოდება. მისი მიზანი კონკრეტული პრობლემის გადაჭრაა (ჩვენ მიერ მოყვანილი მაგალითის შემთხვევაში, ნამცხვრის გამოცხობაა მისი მიზანი). პროგრამირებაშიც ზუსტად ასეა: ალგორითმი არის ნაბიჯ-ნაბიჯ ინსტრუქციების ერთობლიობა, რომელიც კომპიუტერმა უნდა შეასრულოს სასურველი შედეგის მისაღებად.



4.1. როგორ დავწეროთ პროგრამა, რომელიც მომხმარებლის პასუხის მიხედვით განსხვავებულად მოიქცევა

ჩვენი მარტივი დიალოგის შემთხვევაში, ალგორითმი ასე გამოიყურება:

დაიწყე პროგრამა

ნაბიჯი 1: დაუსვი მომხმარებელს კითხვა: "როგორ ხარ?";
ნაბიჯი 2: შეინახე პასუხი ცვლადში "feeling";
ნაბიჯი 3: შეამოწმე პირობა - არის თუ არა "feeling" ტოლი "კარგად"-ის?;
ნაბიჯი 4: თუ პირობა ჭეშმარიტია, დაბეჭდე "მიხარია!";
ნაბიჯი 5: თუ პირობა მცდარია, დაბეჭდე "იმედია, ყველაფერი კარგად იქნება!".

დაასრულე პროგრამა

ამ შემთხვევაში, ეს ალგორითმი, რომელიც ჯერ კიდევ არაა პროგრამული კოდი, გვეხმარება აზროვნებაში და პროგრამული კოდის შემუშავებაში. ასეთ, ადამიანურ ენაზე დაწერილ ალგორითმს, ხშირად **ფსევდოკოდს** უწოდებენ.

პრაქტიკული დავალება 3: "მარტივი დიალოგი"

დაწერე პროგრამა, რომელიც მომხმარებელს ჰკითხავს „როგორ ხარ?“. თუ მომხმარებელი დაწერს „კარგად“, ბოტმა უნდა უპასუხოს „მიხარია!“. ყველა სხვა შემთხვევაში, ბოტმა უნდა უპასუხოს „იმედია, ყველაფერი კარგად იქნება!“.

დავალება 3.1: შეცდომის პოვნა და გასწორება

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

```
feeling = input("როგორ ხარ? ")
if feeling = "კარგად":
      print("მიხარია!")
else:
      print("იმედია, ყველაფერი კარგად იქნება!")
```

დავალება 3.2: კოდის დასრულება
მოცემულ პროგრამულ კოდს აკლია ერთი ან რამდენიმე სტრიქონი. დაამატე მხოლოდ ის, რაც აუცილებელია, რომ პროგრამამ გამართულად იმუშაოს.

```
feeling = input("როგორ ხარ? ")
if feeling == "კარგად":
      print("მიხარია!")
else:
    ...
```

დავალება 3.3: კოდის დაწერა ნულიდან
დაწერე პროგრამული კოდი, შექმენი პროგრამა, რომელიც მომხმარებელს ჰკითხავს „როგორ ხარ?“ და მისი პასუხის მიხედვით დაბეჭდავს შესაბამის კომენტარს.

```
# აქ დაწერე შენი კოდი:
```

სწორი პასუხი (პროგრამული კოდი სრულად):

```
# ამ კოდის საშუალებით შეგიძლია გადაამოწმო შენი ნამუშევარი
feeling = input("როგორ ხარ? ")

if feeling == "კარგად":
      print("მიხარია!")
else:
      print("იმედია, ყველაფერი კარგად იქნება!")
```