Tutorial I.III - Making Decisions

Programming: Everyday Decision-Making Algorithms

Introduction

Imagine you're searching for a new apartment. You view a flat - should you take it or keep looking for something better? This is another classic optimal stopping problem! Just like in dating or hiring, you need to decide when to stop searching.

In this tutorial, we'll learn how computers make decisions using `if` and `else` statements. These are like the "if this, then that" decisions we make every day!

1 - Understanding If Statements

An `if` statement is like asking a yes/no question and doing something based on the answer. For example:

```
flat_rating = 8  # Rate the flat from 0-10

if flat_rating >= 7:
    print("This is a good apartment!")
```

```
This is a good apartment!
```

The structure is:

1. The word `if`
2. A condition that's either True or False
3. A colon `:`
4. Indented code that runs if the condition is True

> **Tip**
>
> Think of `if` statements like everyday decisions:
>
> - IF it's within budget, view it
> - IF the flat is good, apply for it
> - IF it has enough rooms, add it to the shortlist
>
> Computer `if` statements work exactly the same way! They check a condition and then do something based on that check.

> ! Important
>
> Common Mistakes with If Statements:
>
> 1. Forgetting the colon `:` after the condition
> 2. Incorrect indentation of the code block
> 3. Using `=` (assignment) instead of `==` (comparison)
> 4. Forgetting that the condition must result in `True` or `False`

Exercise 1.1 - Your First Decision

Create an `if` statement that prints "Perfect flat!" if the `flat_rating` is 10.

```
flat_rating = 10
# YOUR CODE BELOW

# Test your answer yourself - the cell should print "Perfect flat!" if
executed correctly
```

2 - Adding Else Statements

But what if we want to do something different when the condition is False? That's where `else` comes in:

```
flat_rating = 4  # Not a great flat!

if flat_rating >= 7:
    print("Apply for this flat!")
else:
    print("Keep searching!")
```

```
Keep searching!
```

The structure is:

1. An `if` statement with its condition
2. Code to run if True
3. The word `else` and a colon `:`
4. Indented code to run if False

> Tip
>
> Think of `if-else` like complete either/or decisions:
>
> - IF it's within budget, schedule viewing, ELSE skip it
> - IF the location is good, consider it, ELSE keep searching
> - IF all criteria are met, apply now, ELSE continue looking
>
> The `else` statement is our backup plan!

Exercise 2.1 - Complete Decision

Write an if-else statement that sets `decision` to "Apply now" if `flat_rating` is at least 7, and "Keep searching" otherwise.

```
flat_rating = 6
decision = ""
# YOUR CODE BELOW
```

```
# Test your answer
assert decision == "Keep searching", "The decision should be 'Keep
searching' as the flat rating is less than 7"
print(f"Decision: {decision} as the flat rating is {flat_rating}")
```

> Tip
>
> Writing Better If-Else Statements:
>
> 1. Keep your conditions simple and readable
> 2. Use meaningful variable names
> 3. Consider what should happen in both cases
> 4. Test both paths to make sure they work

3 - Adding Elif (Else If)

Sometimes we need more than two options. That's where `elif` comes in:

```
flat_rating = 8

if flat_rating >= 9:
    print("Amazing flat - apply immediately!")
elif flat_rating >= 7:
    print("Good flat - consider applying")
elif flat_rating >= 5:
    print("Mediocre flat - keep it as backup")
else:
    print("Poor flat - definitely keep looking")
```

```
Good flat - consider applying
```

The structure adds:

- Multiple `elif` conditions between `if` and `else`
- Each condition is checked in order
- The first True condition runs its code

> Tip
>
> Think of `elif` like multiple-choice decisions:
>
> - IF it's perfect → apply immediately
> - ELIF it's good → schedule second viewing
> - ELIF it's okay → keep as backup
> - ELSE → continue searching
>
> It's like a flowchart where only one path can be taken!

> Warning
>
> Important Elif Rules:
>
> 1. Order matters! Put more specific conditions first
> 2. Only one block will execute
> 3. `elif` must come after `if` and before `else`
> 4. You can have as many `elif` blocks as you need

Exercise 3.1 - Apartment Categories

Create a variable `flat_category` that is:

- "Luxury" if rating is 9 or 10
- "Premium" if rating is 7 or 8
- "Standard" if rating is 5 or 6
- "Basic" for anything lower

```
flat_rating = 8
flat_category = ""
# YOUR CODE BELOW
```

```
# Test your answer
assert flat_category == "Premium", "The flat category should be 'Premium'
as the flat rating is 8"
print(f"Flat Category: {flat_category} as the flat rating is
{flat_rating}")
```

4 - Complex Decisions

In real life, we often need to check multiple conditions at once. We can combine conditions using `and` and `or`:

```
flat_rating = 8
weeks_searching = 3
max_search_time = 4

if (flat_rating >= 7) and (weeks_searching < max_search_time):
    print("Take this flat - it's good enough and we still have time!")
elif (flat_rating >= 9) or (weeks_searching >= max_search_time):
    print("Take this flat - either it's perfect or we're out of time!")
else:
    print("Keep looking!")
```

```
Take this flat - it's good enough and we still have time!
```

> Tip
>
> Tips for Complex Conditions:
>
> - Use parentheses to make your logic clear
> - Break very complex conditions into smaller parts
> - Test edge cases to make sure your logic works

Exercise 4.1 - Real World Flat Hunting

Create a variable `should_apply` that is True if:

- The flat rating is at least 8, OR
- The flat rating is at least 6 AND we've been searching for 3 weeks or more

```
flat_rating = 6
weeks_searching = 4
should_apply = False
# YOUR CODE BELOW
```

```
# Test your answer
assert should_apply == True, "The flat rating is 6 and we've been searching
for 4 weeks, so we should apply"
print(f"Should we apply? {should_apply} as the flat rating is {flat_rating}
and we've been searching for {weeks_searching} weeks")
```

5 - The 37% Rule with Decisions

Remember the optimal stopping rule? Let's apply it to apartment hunting:

```
flats_seen = 6
total_viewings = 15
current_rating = 8
best_rating_so_far = 7

# Calculate if we've passed 37% of viewings
threshold_passed = flats_seen >= (total_viewings * 0.37)

if not threshold_passed:
    print("Keep looking - still in observation phase")
elif current_rating > best_rating_so_far:
    print("Apply for this flat!")
else:
    print("Keep looking - waiting for better than our best")
```

```
Apply for this flat!
```

Exercise 5.1 - Implement the Rule

Adjust the code below to change the variable `make_application` to True if:

- We've seen at least 37% of available flats (use 15 total flats), AND
- The current flat is better than the best we've seen so far

```
flats_seen = 6  # We've seen 6 flats
total_viewings = 15  # We plan to view 15 flats in total
current_rating = 9
best_rating_so_far = 8
make_application = False
# YOUR CODE BELOW
```

```
# Test your answer
assert make_application == True, "The flat rating is 9 and we've been
searching for 4 weeks, so we should apply"
print(f"Should we apply for this flat? {make_application}")
```

Conclusion

Great work! You've learned how to make decisions in Python using:

- Simple if statements
- if-else statements
- elif for multiple conditions
- Complex decisions with and/or
- Applying these to real-world stopping problems

Remember, just like in apartment hunting, these tools help us make better decisions in any situation where we need to decide whether to take what we have or keep looking for something better!

Solutions

You will likely find solutions to most exercises online. However, we strongly encourage you to work on these exercises independently without searching explicitly for the exact answers to the exercises. Understanding someone else's solution is very different from developing your own. Use the lecture notes and try to solve the exercises on your own. This approach will significantly enhance your learning and problem-solving skills.

Remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities. If you encounter difficulties, review the lecture materials, experiment with different approaches, and don't hesitate to ask for clarification during class discussions.

Later, you will find the solutions to these exercises online in the associated GitHub repository, but we will also quickly go over them next week. To access the solutions, click on the Github button on the lower right and search for the folder with today's lecture and tutorial. Alternatively, you can ask ChatGPT or Claude to explain them to you. But please remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities.

That's it for part I! Next week, we'll apply these skills exploration and exploitation!