Tutorial I.I - Variables and Types

Programming: Everyday Decision-Making Algorithms

Introduction

Welcome to this interactive Python tutorial on variables, types, and algorithmic thinking! We'll explore these concepts through the lens of optimization problems, specifically using the famous "Secretary Problem" as our running example.

The Secretary Problem asks: "If you need to hire the best candidate, and you can only interview candidates one at a time (with no going back), when should you stop and make an offer?" This is a perfect example of optimal stopping and will help us understand how variables and data types play crucial roles in algorithmic thinking. To make the problem more interesting, we will use the Secretary Problem to help us make better dating decisions! The question is thus: **When should you stop dating and commit to someone?**

Known academically as the "Optimal Stopping Problem" or "The Marriage Problem", it asks: If you want to find the best possible partner, and you can only date one person at a time (no going back to previous dates!), when should you stop dating and propose?

Let's learn how to use Python to help us in order to make better dating decisions!

1 - Variables

Just like you keep track of important things in dating (name, phone number, red flags), in Python we use variables to store information:

```
date_name = "Alexander" # Current date's name
red_flags = 0 # Count of red flags noticed
butterflies_felt = True # Do I feel butterflies?
compatibility_score = 8.5 # On a scale of 1-10, how compatible are we?
print("We have defined a some variables based on tonight's date.")
```

```
We have defined a some variables based on tonight's date.
```

> **Tip**
>
> Variables in Python are like containers that store information. When we write `date_name = "Alexander"`, we're:
>
> 1. Creating a container called `date_name`
> 2. Putting the value "Alexander" inside it
> 3. Telling Python this is a text value (string)

> **Note**
>
> Notice how we use meaningful variable names that describe what they contain:
>
> - `date_name` is better than just `name`
> - `red_flags` is clearer than just `flags`
> - `butterflies_felt` clearly indicates a yes/no feeling

Exercise 1.1 - First Date Setup

Create a variable `currently_dating` and set it to `False` (because we're starting our dating journey).

```
# YOUR CODE BELOW
```

```
# Test your answer
assert currently_dating == False, "Let's start fresh - set currently_dating to False!"
print("Ready to start dating! We have defined the variable `currently_dating` as
 ↪  boolean.")
```

Exercise 1.2 - Date Name

Create a variable `date_name` with your tonight's date's name (let's say `"Mika"`).

```
# YOUR CODE BELOW
```

```
# Test your answer
assert date_name == "Mika", "Let's use 'Mika' as our tonight's date's name"
print(f"We have defined a string variable for our tonight's date. Now, we are looking for
 ↪  {date_name}!")
```

2 - Types

Just like dating involves different types of information, Python has different types for different kinds of data:

- Integers (`int`): Counting things (number of dates, red flags)
- Floats (`float`): Ratings and scores (compatibility score)
- Booleans (`bool`): Yes/no information (are they single?)
- Strings (`str`): Text (names, favorite food)

> **!** Important
>
> Why do types matter? Because:
> - They determine what operations we can perform (you can't divide names!)
> - They affect how much memory the computer needs
> - They help prevent errors in our code

```
dates_gone_on = 7  # int: keeping count
chemistry_level = 9.5  # float: rating of the butterflies
is_single = True  # bool: availability
favorite_food = "Sushi"  # str: important info!
print(f"We have gone on {dates_gone_on} dates, a chemistry level of {chemistry_level} in
 ↪  the last date, and we love {favorite_food}!")
```

```
We have gone on 7 dates, a chemistry level of 9.5 in the last date, and we love Sushi!
```

> Tip
>
> You can check a variable's type using the `type()` function.

Exercise 2.1 - Dating Score

Create a float variable `minimum_rating` and set it to `7.0` (our standards are high!). Of course, we don't objectify people and rate their appearance, thus we rate the compatibility!

```
# YOUR CODE BELOW
```

```
# Test your answer
assert minimum_rating == 7.0 and isinstance(minimum_rating, float), "Set your standards
 ↪  (minimum_rating) to 7.0!"
print(f"We won't settle for less than {minimum_rating}/10, where the minimum rating is
 ↪  defined as a float!")
```

Exercise 2.2 - Dating History

Create an integer variable `past_relationships` and set it to `0`.

```
# YOUR CODE BELOW
```

```
# Test your answer
assert past_relationships == 0 and isinstance(past_relationships, int), "Let's start
 ↪  fresh with 0 past relationships"
print("Clean slate, although likely not true! We have defined the variable
 ↪  `past_relationships` as an integer and are ready to start dating!")
```

3 - Converting Data Types

Python provides several functions to convert between different types:

- `int()`: Converts to integer
  - `int("37")` → `37`
  - `int(37.8)` → `37` (truncates decimal)
- `float()`: Converts to float
  - `float("37.5")` → `37.5`
  - `float(37)` → `37.0`
- `str()`: Converts to string
  - `str(37)` → `"37"`
  - `str(37.5)` → `"37.5"`
- `bool()`: Converts to boolean
  - `bool(1)` → `True`
  - `bool(0)` → `False`
  - `bool("")` → `False`
  - `bool("hello")` → `True`

In optimal stopping for dating, the math says you should date and reject the first 37% of your dating pool, then commit to the next person who's better than everyone you've met before! Let's track this:

```
dating_pool = 100 # The number of people we will date
exploration_phase = int(dating_pool * 0.37)  # The 37% rule
print(f"Date (and reject) the first {exploration_phase} people")
```

```
Date (and reject) the first 37 people
```

Exercise 3.1 - Decision Point

Convert the string `"37"` into an integer variable named `stopping_point`.

```
stopping_point = "37"
# YOUR CODE BELOW
```

```
# Test your answer
assert isinstance(stopping_point, int) and stopping_point == 37, "Convert '37' to an
 ↪  integer"
print(f"After person {stopping_point}, we'll get serious about finding the best possible
 ↪  partner.")
```

> **Tip**
>
> When converting strings to numbers: - Make sure the string contains only numeric characters - Use `int()` for whole numbers - Use `float()` if you need decimal points

4 - String Formatting

Let's create a nice dating profile using string formatting:

```
name = "Casey"
age = 25
bio = f"{name}, {age} - Looking for someone special!"
print(bio)
```

```
Casey, 25 - Looking for someone special!
```

Exercise 4.1 - Profile Message

We have already defined the variables `name` and `minimum_rating`. Now we want to use them in the context of a formatted string. You have already seen how to do this in the previous cells, but not explicitly. String formatting is a powerful tool in Python that allows you to insert variables into strings. We do so by using curly braces `{}` around the variable names and with the `f` prefix before the string. For example:

```
f"Hello, {name}! Nice to meet you."
```

```
'Hello, Casey! Nice to meet you.'
```

Here, `{name}` will insert the value of `name` into the string if the variable `name` is defined.

Create a formatted string `profile_message` using the variables `name` and `minimum_rating` that says "Looking for {name} (must be at least {minimum_rating}/10)".

```
# YOUR CODE BELOW
```

```
# Test your answer
assert profile_message == "Looking for Casey (must be at least 7.0/10)", "Format your
 ↪  dating profile message correctly!"
print("Your dating profile is ready!")
```

Conclusion

Congratulations! You've learned about Python variables and types through the lens of dating optimization!

Remember:

- Variables help track your dating journey
- Different types handle different kinds of dating data
- The 37% rule suggests when to get serious
- String formatting helps create the perfect dating profile

Dating tip: While this mathematical approach is a good starting point, remember that love isn't always logical! We haven't considered the emotional aspects of dating yet or the fact that we might not be the best partner for each other.

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.

Continue to the next tutorial to learn about comparisons and conditionals in the context of the optimal stopping problem!