Variables in Python

Understanding Variables in Python

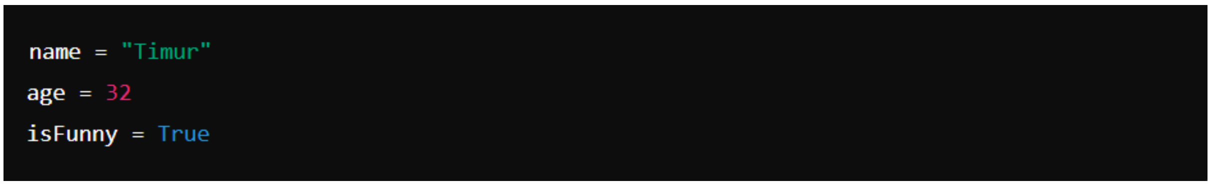
Variables are fundamental to any programming language, and Python is no exception. A variable in Python is essentially a reserved memory location (think of it as the brain of your computer) to store values. It allows Python to remember information about your program.

For example, let's imagine you are creating an application in Python. It is common to see the user log in and be welcomed with a nice message using their username, such as: 'Welcome back, John'. The username displayed on the application changes depending on who is currently connected. That's an example of a variable. The application remembers the username of the person through a variable.

Creating Variables

In Python, creating a variable is straightforward. You simply assign a value to a variable name using the = operator.

Example:



```
name = "Timur"
age = 32
isFunny = True
```

In the above example, three variables have been created: `name`, `age`, and `isFunny`. Each of these variables has a value associated with them. `name` has the value "Timur", so if we need to ask the application the user's name, the answer will be "Timur". Of course, in this specific instance, it will always be "Timur". But as the name variable implies, it can change over time and doesn't have to be "Timur" forever. When the user logs out and someone else logs in, the variable `name` might get a new value such as "Sarah". From that point on, the variable `name` will be "Sarah".

Using Variables

Once you've created variables, you can use them in your programs in various ways. For example, you can perform arithmetic operations, print (display) their values, or use them in functions (see the next chapter).

Example:

```python
grade_1 = 17
grade_2 = 14
average_grade = (grade_1 + grade_2) / 2
print(average_grade)
```

In the above example, three variables have been created again. The first two contain the grades of a user (in this case, 17 and 14 respectively). Then, the program computes the average grade of the user using some math concepts (Python is pretty good at doing math!). Finally, the average grade of the user will be displayed using a function called `print` (more on this in the next chapter) for the user to see on their screen. Again, being variables, the values of `grade_1` and `grade_2` can vary over time and produce different outcomes depending on their values. This allows powerful programs to be created.

Variable Naming Rules

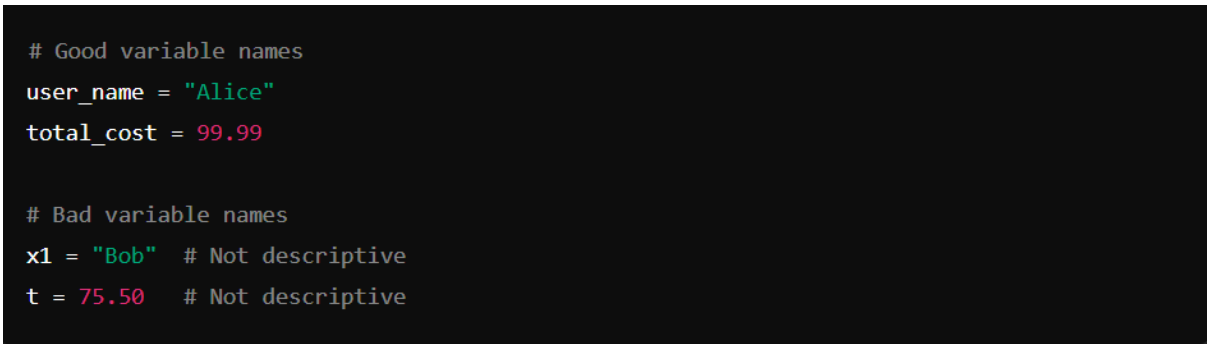
When naming variables in Python, there are a few rules to keep in mind:

1. Variable names must start with a letter (a-z, A-Z) or an underscore (_).
2. The remainder of your variable name can include letters, numbers, or underscores.
3. Variable names are case-sensitive (e.g., `name`, `Name`, and `NAME` are three different variables).

Best Practices for Variable Names

- Use meaningful names that describe the data the variable holds (e.g., `total_price`, `user_age`).
- Use lowercase letters and underscores to separate words (this is known as snake_case).

Examples of Good and Bad Variable Names:



```python
# Good variable names
user_name = "Alice"
total_cost = 99.99

# Bad variable names
x1 = "Bob"   # Not descriptive
t = 75.50    # Not descriptive
```

Conclusion

Variables are a basic yet powerful concept in Python programming. By understanding how to create and use them effectively, you can write more readable and maintainable code. Remember to follow the naming conventions and best practices to ensure your code is clean and understandable.