perplexity

# Python Variables: Detailed Notes for Lifelong Understanding

## 1. What is a Variable? (Theory)

- **A variable** is simply a name that refers to a value stored in the computer's memory.
- Think of a variable as a labeled box where you can store anything (a number, a word, etc.) and you can change what's in it at any time.
- In Python, you **don't need to declare the data type** of a variable before using it, making Python dynamic and flexible.
- Variables in Python **point to objects** in memory; they are not boxes, but name tags you attach to objects.
- **Variables make your code readable**, maintainable, and reusable.

## 2. Rules for Naming Variables (Theory)

- The **name must start** with a letter (a-z, A-Z) or an underscore (_) but never with a number.
- After the first character, it can include letters, numbers, and underscores.
- Variable names are **case-sensitive** (`data` and `Data` are different).
- Avoid using Python **keywords** (like `for`, `while`, `if`, etc.) as variable names.

## 3. Variable Assignment and Data Types (Theory)

- Assign values using the `=` operator (assignment).
- The **data type** of a variable is automatically inferred from the value.
- You can change the type later just by assigning a new value.

## 4. Best Practices for Variables (Theory + Tips)

- Choose meaningful names: `age`, `username`, `total_price` (instead of `a`, `b`, `c`).
- Follow naming conventions: `snake_case` is standard in Python (`user_age`).
- Use lowercase letters except for constants.

## 5. Coding Examples (Main Section)

Time to see **concepts in action**! Each code example has comments to explain every step and variation.

### A. Basic Variable Declaration

```
# Assigning integer value
x = 10
print("x:", x)  # Output: 10

# Assigning a string value
name = "Alice"
print("name:", name)  # Output: Alice

# Assigning a floating-point number
price = 3.99
print("price:", price)  # Output: 3.99

# Assigning a boolean value
is_logged_in = True
print("is_logged_in:", is_logged_in)  # Output: True

# Assigning multiple variables in one line
a, b, c = 1, 2, 3
print(a, b, c)  # Output: 1 2 3

# Assigning the same value to multiple variables
height = width = length = 0
print(height, width, length)  # Output: 0 0 0
```

### B. Changing Value and Type

```
# Python variables are flexible - type can change easily
num = 5         # num is an int
print(type(num)) # <class 'int'>

num = "Five"    # Now num is a string!
print(type(num)) # <class 'str'>
```

### C. Dynamic Typing

```
# No need to specify data type
value = 10          # int
print(value, type(value))

value = "abc"       # Now, str
print(value, type(value))

value = 3.14        # Now, float
print(value, type(value))
```

### D. Good vs. Bad Variable Names

```
# Good
user_score = 250

# Bad
x = 250    # What does 'x' mean?

print(user_score)
```

### E. Variable Naming Rules (Code Illustration)

```
# Valid names
age = 20
_age = 30
age_2 = 25

# Invalid examples (Uncomment to see the error)
# 2age = 20  # starts with number, invalid
# for = 5    # keyword, invalid
```

### F. String Variables

```
city = "Delhi"
country = 'India'
```

```
# Concatenation
address = city + ", " + country
print(address)  # Output: Delhi, India

# Changing value
city = "Mumbai"
print(city)     # Output: Mumbai
```

### G. Variables and Memory

```
# Variables point to objects in memory
a = 42
b = a   # b points to the same value as a

print("a:", a, "b:", b)  # Output: a: 42 b: 42

# Changing 'a' will not change 'b'
a = 100
print("a:", a, "b:", b)  # Output: a: 100 b: 42
```

### H. User Input and Variables

```
# Taking user input and storing in a variable
user_name = input("Enter your name: ")
print("Hello,", user_name)
```

### I. Using Variables in Expressions

```
num1 = 7
num2 = 3

# Arithmetic using variables
sum = num1 + num2
print("Sum:", sum)

product = num1 * num2
print("Product:", product)
```

## J. Constants in Python

- By convention, constants are variables with **all uppercase names.**
- Python does not force them to be unchangeable, but developers agree not to modify them.

```
PI = 3.14159
GRAVITY = 9.8

print("Pi:", PI)
print("Gravity:", GRAVITY)
```

## K. Swapping Variables

```
# Easy swapping in Python
x = 1
y = 2
print("Before:", x, y)

x, y = y, x  # Swap values
print("After:", x, y)  # Output: 2 1
```

## L. Deleting Variables

```
temp = 99
print(temp)   # Output: 99

del temp     # Deletes the variable

# print(temp) # Uncommenting this will cause an error since temp no longer exists
```

## M. Global vs Local Variables

```
# Global variable
x = 50

def func():
    x = 10  # Local variable (different from global x)
    print("Inside function:", x)
```

```
func()
print("Outside function:", x)  # Remains 50
```

## 6. Quick Recap Table

| Concept | Meaning / Example |
|---|---|
| Variable | `var = 10` |
| Dynamic typing | `num = 5; num = "five"` |
| Multiple assign | `a, b = 1, 2` |
| Swapping | `a, b = b, a` |
| Good names | `user_age`, `total_price` |
| Constants | `PI = 3.14` (convention) |
| Local/Global | Scope inside/outside functions |

## 7. Practice for Mastery

Try these on your own to get fully comfortable:

1. Store your age and your friend's age in variables. Print the older person.
2. Swap the values of two variables using a single line of code.
3. Change the value and type of a variable multiple times in your code. Print its value and type each time.

## 8. Final Pro Tips

- Use descriptive variable names.
- Never use Python's built-in names as your variables (`list`, `str`, etc.).
- Remember: variables are **references to objects**, not the objects themselves.

## Remember:

> **"Variables are the backbone of programming. Master variables, and you command the foundation of all logic in Python!"**