📘 Lecture 04: Variables & Data Types in Python

✅ 1. What Are Variables?

- A **variable** is a *named container* used to **store data temporarily** while a program runs.
- The **value of a variable can change**, so it's called "variable."

🧠 Example (Conceptual):

A **glass** is a container (variable), and **water** is the data inside it.

```
python
CopyEdit
fruit = "apple"
a = 10
```

📘 2. Rules for Naming Variables

Rule	Example	Notes
✅ Must start with letter/underscore	`_name`, `username`	
❌ Cannot start with number	`1user` ❌, `user1` ✅	
✅ Can contain letters, numbers, underscore	`user_name`, `age1`	
✅ Case-sensitive	`Name ≠ name`	
❌ Don't use reserved words	`print = 5` ❌	

📘 3. Naming Conventions

Style	Example
Snake Case	`first_name` ✅
Camel Case	`firstName` ✅

Style	Example
Pascal Case	`FirstName` ✅

□ 4. Initializing Variables

✍ Syntax:

```
python
CopyEdit
variable_name = value
```

📌 Examples:

```
python
CopyEdit
fruit = "apple"      # string
a = 10               # integer
pi = 3.14            # float
is_active = True     # boolean
```

□ 5. Data Types in Python

Type	Description	Example
`int`	Whole number	`10`, `-5`
`float`	Decimal number	`3.14`, `-0.01`
`str`	Text enclosed in quotes	`"hello"`, `'123'`
`bool`	True or False	`True`, `False`

💻 6. Examples:

```
python
CopyEdit
country = "Pakistan"
age = 25
price = 99.99
is_active = True
python
CopyEdit
```

```
FirstName = "Ali"
firstname = "Ahmed"   # both are different!
python
CopyEdit
1user = "Ali"         # ❌ Invalid
print = 10            # ❌ Invalid
```

🎯 7. Best Practices

- ✅ Use **meaningful names**: `student_name` instead of `sn`
- ✅ Be **consistent** with naming style
- ❌ Don’t use **reserved keywords** as variable names

📌 8. Summary

- Variables are containers for storing data
- Follow **naming rules & conventions**
- Understand **basic data types**
- Use clear, descriptive names for readability

🧠 Difficult Terms:

Term	Explanation
Variable	Named location to store data
Reserved Word	Built-in keywords in Python, like `if`, `print`, `for`

□ Lecture 04 (part 2): Python Collections & Virtual Environment Setup

---

📚 Overview:

- Deep dive into **collections**: List, Tuple, Dict, String
- Best practices for using collections
- Create & use **virtual environments** using Anaconda

📂 1. Dictionary

- Stores **key-value pairs**
- Keys must be **unique**

```
python
CopyEdit
person = {
  "name": "John",
  "age": 30,
  "city": "New York"
}
```

✅ Access:

```
python
CopyEdit
print(person["name"])
print(person.get("gender", "Not Found"))  # Avoids crash
```

🆕 Add / Update:

```
python
CopyEdit
person["country"] = "USA"
person["city"] = "London"
```

❌ Remove:

```
python
CopyEdit
del person["age"]
```

🔎 Check Key:

```
python
CopyEdit
if "name" in person:
    print("Key exists")
```

□ 2. Lists

- Ordered, **modifiable**, and allows duplicates

```
python
```

```
CopyEdit
numbers = [10, 20, 30, 40]
```

🔍 Access:

```
python
CopyEdit
print(numbers[0])
print(numbers[-1])
```

➕ Add:

```
python
CopyEdit
numbers.append(50)
numbers.insert(2, 15)
```

❌ Remove:

```
python
CopyEdit
numbers.pop()
```

✏️ Modify:

```
python
CopyEdit
numbers[0] = 99
```

✨ 3. Strings

```
python
CopyEdit
name = "John"
```

🔗 Combine (Concatenate):

```
python
CopyEdit
full_name = first_name + " " + last_name
```

🔁 Repeat:

```
python
CopyEdit
print("ha" * 3)  # hahaha
```

🔑 Index & Slice:

```
python
CopyEdit
print(name[0])       # J
print(name[1:4])     # ohn
```

f-String (✅ Best):

```
python
CopyEdit
age = 25
print(f"My age is {age}")
```

🔐 4. Tuples

- Like lists, but **immutable** (unchangeable)

```
python
CopyEdit
person_tuple = ("Bob", 30, "Engineer")
```

🔍 Access:

```
python
CopyEdit
print(person_tuple[0])
print(person_tuple[1:])
```

❌ Modify:

```
python
CopyEdit
person_tuple[1] = 35  # ❌ Error
```

✂ Unpack:

```
python
CopyEdit
name, age, job = person_tuple
```

🛠 5. Virtual Environment Setup (Anaconda)

✅ Steps:

```
bash
CopyEdit
cd path_to_folder
conda create -n farah python=3.10
conda activate farah
code .
```

1. Open **VS Code**
2. Select the correct **kernel** from the top-right: `Farah`
3. ✅ Allow access if prompted

🧠 Best Practices for Collections

Use	Collection Type
Ordered & Editable	List
Fixed & Secure	Tuple
Key-Value Mapping	Dictionary
Unique Items	Set (not covered today)

📋 Homework & Revision

- 💻 Practice dictionary, list, tuple, and string
- 🔁 Try modifying, accessing, and looping collections
- □ Use them in different real-life examples (e.g., storing student info, shopping cart)
- ⚙️ Keep practicing virtual environments

🎓 Conclusion

- You now understand **collections** and their **real uses**
- You've practiced **virtual environment setup**
- Mastery of these skills will help you **build solid Python projects**