Python Fundamentals – Internship Learning Notes (Revised)

Author: Drushti Singh
Context: Internship Learning (Foundation Revision)
Objective: Strengthen Python basics through hands-on practice, error analysis, and clean coding principles.

---

1. Learning Objective

The goal of this task was to revise Python fundamentals and understand **why errors occur**, not just how to fix them. Each exercise focused on writing code, predicting outcomes, debugging errors, and summarizing learnings.

---

2. Key Concepts Covered (Summary)

- Python Variable Naming Rules
- Case Sensitivity in Python
- Keywords and Reserved Words
- Python Data Types and Dynamic Typing
- Type Conversion (Casting)
- Basic Operators and Expressions

---

3. Python Variables – Core Rules

Rules Learned

- Variable names must start with a **letter or underscore (_)**
- They **cannot start with a number**
- **Keywords** cannot be used as variable names
- Python is **case-sensitive** (`name` ≠ `Name`)

Example

```
student_name = "Drushti"
print(student_name)
```

Output:

```
Drushti
```

Learning: Clean and valid variable names prevent syntax errors.

---

4. Common Errors and Debugging

Errors Observed

- Using spaces in variable names
- Using incorrect variable case
- Mixing incompatible data types (e.g., string + integer)

Example Error & Fix

```
text = "Total: "
num = 10
print(text + str(num))
```

Learning: Type conversion is required when combining different data types.

---

5. Python Data Types

Data Types Practiced

- `int` – Integer values
- `float` – Decimal values
- `str` – Text data
- `bool` – True / False values

Key Insight

Python supports **dynamic typing**, meaning the same variable can store different data types at different times.

---

6. Operators and Expressions (Overview)

- Arithmetic operators: `+ - * / // % **`
- Comparison operators: `== != > < >= <=`
- Logical operators: `and or not`

Learning: Operator precedence affects the final output of expressions.

---

7. Final Outcome

- Revised Python fundamentals with practical examples
- Improved understanding of error messages

- Built habit of predicting output before execution
- Strengthened debugging and problem-solving skills

---

8. Conclusion

This revision helped reinforce Python basics and improved confidence in writing clean, error-free code. A strong foundation is essential for advanced topics and real-world development.

Note: This document is a revised and personalized version of internship practice tasks, structured to reflect my individual learning and understanding.