Python – Variables (Complete Notes)

1. Variable

- A variable is a reserved memory location to store values.

2. Purpose of a Variable

- Represents data (collection of facts).
- Data can be:
    - Alphabets
    - Numbers
    - Alphanumeric
    - Symbols

3. Properties of a Variable

Each variable has:

- Name
- Type
- Value

4. Naming Conventions for Variables

- Start name with lowercase letter.
- If multiple words: separate with underscore (`snake_case`).
- Example: `student_name`.

5. Creating a Variable

- Syntax:
  `variable_name = value`
- Example:
- `python`

```
age = 16
print(age)  # Output: 16
```

-

6. Creating Multiple Variables in a Single Line

- Possible in Python.
- Rule: Equal number of variables (left side) and values (right side).
- Example:
- `python`

```
a, b, c = 1, 2, 3
```

-

7. Assigning a Single Value to Multiple Variables

- Assign one value to many variables at once.
- Example:
- `python`

```
a = b = c = 3
```

-

8. Variable Re-initialization

- Variables can be updated/re-assigned.
- New value replaces the old one.
- Example:
- `python`

```
sal = 10000
sal = 12000
```

-

9. Important Notes

- Python has dynamic data typing:
  - No need to declare data type explicitly.
  - Data type is determined automatically at runtime.

Session-6 Python – Data Types

1. What is a Data Type in Python

- Data type → represents the type of data stored in a variable/memory.
- Examples:
    - `emp_id = 1` → integer
    - `name = "Daniel"` → string
    - `salary = 10000.56` → float
- **Check type:** `type(variable)`

---

2. `type(p)` Function

- Predefined function to check **variable's data type**.
- Syntax:

  python

```
type(variable)
```

-
- Example:

  python

```
a = 1
print(type(a))  # <class 'int'>
```

-

---

3. Types of Data Types

Two Main Categories

1. **Built-in Data Types** → provided by Python.
2. **User-defined Data Types** → created by programmer (e.g., classes, modules, arrays).

3.1 Built-in Data Types

- **Numeric types**

    1. **int**
        - Stores integers (no decimal).
        - No size limit in Python.
    2. python

```python
x = 20
print(type(x))  # <class 'int'>
```

3.
4. **float**
    - Stores numbers with decimals.
5. python

```python
salary = 10000.56
print(type(salary))  # <class 'float'>
```

6.
7. **complex**
    - Stores complex numbers (not shown in PDF examples but exists in Python).

- **Boolean (`bool`)**
    - Values: `True` / `False`
    - Internally: `True` → 1, `False` → 0
- python

```
a = True
b = False
```

-
- **None type**
    - Represents absence of value.
    - Example:

      python

```
x = None
print(type(x))  # <class 'NoneType'>
```

    -
    - Functions/methods may return `None`.

- **Sequences**
    - Can store a collection of values.
    - Types:
        1. **String (`str`)** → characters in `'single'`, `"double"`, or `'''triple'''` quotes.

           python

```
name = "Daniel"
```

2.
3. **List (`list`)** → mutable, ordered, `[ ]` brackets.

   python

```
values = [10, 20, 30]
```

4.
5. **Tuple (`tuple`)** → immutable, ordered, `( )` brackets.

   python

```
values = (10, 20, 30)
```

6.
7. **Set (`set`)** → mutable, unordered, `{ }` braces (no duplicates).

   python

```
values = {10, 20, 30}
```

8.
9. **Dictionary (`dict`)** → `{key: value}` pairs.

   python

```
details = {1: "A", 2: "B"}
```

10.
11. **Range (`range`)**
    - Represents sequence of numbers:
      - `range(end)` → 0 to end-1
      - `range(start, end)` → start to end-1
      - Can be iterated using `for` loop.
12. python

```
for i in range(5):
    print(i)  # 0 to 4
```

13.

3.2 User Defined Data Types

- Created by the programmer.
- Examples: **class**, **module**, **array**.
- Defined using `class` keyword.
- Will be discussed in **OOP** chapter.

🔹 Quick Revision Table

Data Type	**Syntax Example**	**Mutable?**	**Ordered?**	**Duplicates ?**	**Notes**
int	`x = 10`	No	-	-	no decimal
float	`x = 3.14`	No	-	-	decimal values
bool	`x = True`	No	-	-	True/False
NoneType	`x = None`	No	-	-	no value
str	`"Hello"`	No	Yes	Yes	sequence of chars
list	`[1,`	Yes	Yes	Yes	mutable
tuple	`(1, 2, 3)`	No	Yes	Yes	immutable
set	`{1, 2, 3}`	Yes	No	No	unique
dict	`{1: 'A', 2: 'B'}`	Yes	Yes(3.7+)	Keys unique	key:value
range	`range(5)`	No	Yes	Yes	generates numbers
