Noteskarts

Smart Notes. Bright Future.

COMPUTER APPLICATIONS IN PHARMACY

Unit 1 — Introduction to Python Programming

For D.Pharm & B.Pharm Students

www.noteskarts.com

Get Noteskarts Prime — Full Access

Scan the QR code above with your phone camera to download Noteskarts Prime and unlock complete detailed notes, PYQ papers, MCQ tests & video lectures for D.Pharm, B.Pharm, GPAT & Pharmacy Competitive Exams.

Free Bonus — Noteskarts Puzzle Lab

Revise this exact topic through fun MCQ games — Rapid MCQ, Term Scramble, True or False & more. 100% free, no sign-up needed. Scan and play!

Introduction to Python Programming

Python is a high-level, general-purpose, interpreted programming language created by Guido van Rossum and first released in 1991. It is widely used today in pharmacy informatics, data analysis of clinical/pharmacokinetic data, automation of laboratory calculations, and building simple decision-support tools.

Imp Features of Python

- **Easy to Learn and Read** — syntax resembles plain English; ideal for beginners with no prior coding background.
- **Interpreted Language —** code executes line-by-line, making debugging easier.
- **Free and Open Source** — freely available and community supported.
- **Platform Independent** — the same code runs on Windows, macOS and Linux.
- **Extensive Libraries** — huge collection of built-in and third-party libraries (e.g., NumPy, Pandas for pharmacy data analysis).
- **Dynamically Typed —** no need to declare the data type of a variable explicitly.
- **Supports Multiple Paradigms** — procedural, object-oriented, and functional programming.

Why Pharmacy Students Learn Python

Python is used to analyse pharmacokinetic datasets (Cmax, Tmax, AUC), automate dose calculations, plot dissolution profiles, and manage large drug/ADR databases — skills valued in pharmaceutical research and the pharma industry.

A Brief History

Python was designed by Guido van Rossum at Centrum Wiskunde & Informatica (CWI), Netherlands, and released publicly in 1991. Its name comes from the British comedy show 'Monty Python's Flying Circus', not the snake. Python 3, the current standard version, introduced cleaner syntax and better Unicode support compared to the older Python 2.

Applications of Python in the Pharmacy & Healthcare Sector

- Data Analysis — processing clinical trial data, ADR reports, and pharmacokinetic parameters using Pandas.
- Data Visualization — plotting dissolution profiles, calibration curves, and concentration-time graphs with Matplotlib.
- Automation — automating repetitive dose or dilution calculations, report generation, and inventory checks.
- Machine Learning — drug discovery, QSAR modelling, and predicting drug-likeness of new molecules.
- Hospital & Pharmacy Management Systems — building simple billing, stock, and patient record software.

Installing Python and Setting Up an IDE

What is an IDE?

An Integrated Development Environment (IDE) is software that combines a code editor, debugger, and program-running tools in one interface, making it easier to write, test, and fix code compared to a plain text editor.

Steps to Install Python

- Step 1: Go to the official website — python.org/downloads
- Step 2: Download the latest stable version for your operating system (Windows/macOS/Linux).
- Step 3: Run the installer. IMPORTANT: tick the checkbox 'Add Python to PATH' before clicking Install.
- Step 4: Verify installation by opening Command Prompt/Terminal and typing: python --version
- Step 5: Install pip (usually bundled automatically) to manage libraries.

Writing and Running Your First Program

Once Python and an IDE are installed, code can be saved as a file with a .py extension and executed. In an IDE, this is usually done by clicking a 'Run' button or pressing a shortcut (e.g., F5 in IDLE, the green arrow in PyCharm/VS Code).

```
# my_first_program.py
print("Welcome to Pharmacy Programming")
print("1 tablet =", 500, "mg Paracetamol")
```

Output:

```
Welcome to Pharmacy Programming
1 tablet = 500 mg Paracetamol
```

□ Two Ways to Run Python Code

Script mode: write all code in a .py file and run it at once (used for full programs). Interactive mode (REPL/Jupyter cell): run one line/cell at a time and see results instantly — best for learning and quick data checks.

Jupyter Notebook, PyCharm & VS Code — Comparison

IDE	Nature	Best Used For	Cost
Jupyter Notebook	Browser-based, cell-by-cell execution	Data analysis, learning, visualizing PK data step-by-step	Free, part of Anaconda
PyCharm	Full-featured, professional IDE	Large projects, app/software development	Free (Community) / Paid (Pro)
VS Code	Lightweight, highly customizable editor+IDE	General coding, scripting, extensions for Python	Free
IDLE	Python's built-in basic IDE	Simple scripts, quick testing	Free (comes with Python)

Getting Started with Each IDE

- Jupyter Notebook: install via Anaconda distribution or run 'pip install notebook', then launch with the command jupyter notebook — it opens in your web browser with a cell-based interface.
- PyCharm: download the Community (free) edition from jetbrains.com, install, then create a 'New Project' and select the Python interpreter to begin coding.
- VS Code: download from code.visualstudio.com, then install the official 'Python' extension from the Extensions Marketplace to enable running and debugging .py files.

Advantages of IDEs over Text Editors

Feature	Text Editor	IDE
Syntax Highlighting	Rarely available	Built-in, colour-coded
Error Detection	No real-time checking	Real-time error/warning underline
Code Completion	Not available	Auto-suggest (IntelliSense)
Debugging Tools	Absent	Breakpoints, step-through debugger
Run Code	Needs separate terminal	Run/Stop button built in
Version Control	Manual	Integrated Git support

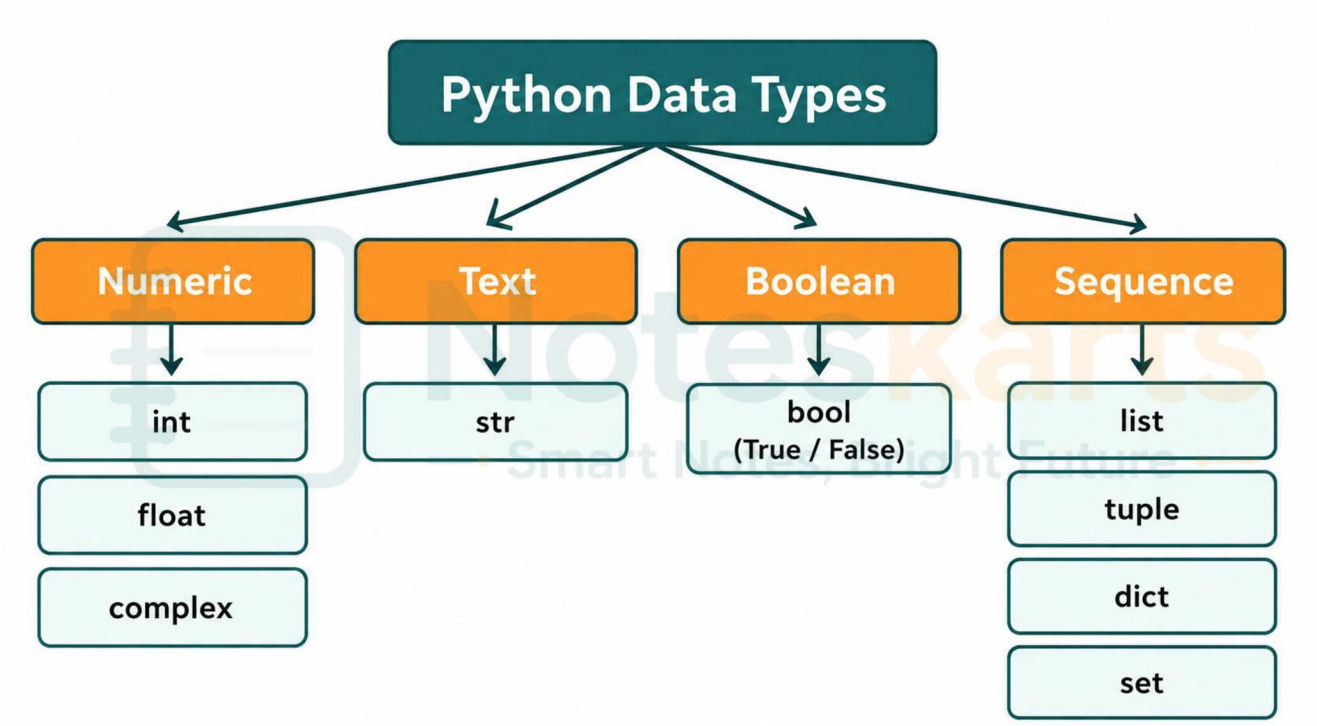
Python Variables and Data Types

Variables

A variable is a named location in memory used to store a value. In Python, variables do not need explicit type declaration — the type is decided automatically based on the assigned value.

Example	Meaning
`drug_name = "Paracetamol"`	String variable
`dose_mg = 500`	Integer variable
`half_life = 2.5`	Float variable
`is_available = True`	Boolean variable

Python Data Types



Data Type	Description	Example
`int`	Whole numbers, positive or negative	dose = 500
`float`	Decimal / real numbers	conc = 12.5
`str`	Text enclosed in quotes	drug = "Aspirin"
`bool`	Logical value True or False	expired = False

Data Type	Description	Example
`complex`	Numbers with real & imaginary part	z = 2 + 3j
`list`	Ordered, changeable collection	drugs = ["A","B"]
`tuple`	Ordered, unchangeable collection	coords = (1, 2)
`dict`	Key–value pairs	d = {"Cmax":45}

□ Common Mistake

Writing dose = "500" stores the number as text (str), not as a usable number. Attempting dose + 10 on a string value causes a TypeError. Always check the data type using type(variable_name) before doing calculations.

Worked Example — Storing a Drug's Details

```
drug_name = "Paracetamol"      # str
dose_mg = 500                  # int
half_life_hr = 2.5             # float
is_prescription_only = False   # bool

print(drug_name, dose_mg, half_life_hr, is_prescription_only)
```

Comments in Python

Comments are notes in the code that Python ignores while running — used to explain logic to readers. A single-line comment starts with #. A multi-line comment/docstring is written between triple quotes ''' ... '''.

Type	Syntax	Example
`Single-line comment`	# text	# calculate total dose
`Multi-line comment`	''' text '''	'''This program\ncalculates dose'''

Type Casting

Type casting (type conversion) means converting one data type into another. Python supports this through built-in functions.

Function	Converts To	Example	Result
`int()`	Integer	int("25")	25
`float()`	Float	float("3.5")	3.5
`str()`	String	str(100)	"100"
`bool()`	Boolean	bool(0)	False

Remember

Implicit casting happens automatically (e.g. int + float = float). Explicit casting is done manually by the programmer using int(), float(), str(), etc.

Basic Operators

Arithmetic Operators

Operator	Meaning	Example (a=10, b=3)	Result
+	Addition	a + b	13
-	Subtraction	a - b	7
*	Multiplication	a * b	30
/	Division (float result)	a / b	3.33
//	Floor Division	a // b	3
%	Modulus (remainder)	a % b	1
**	Exponent (power)	a ** b	1000

Comparison Operators

Operator	Meaning	Example	Result
==	Equal to	5 == 5	True
!=	Not equal to	5 != 3	True
>	Greater than	7 > 4	True
<	Less than	7 < 4	False
>=	Greater than or equal to	5 >= 5	True

Operator	Meaning	Example	Result
<=	Less than or equal to	3 <= 2	False

Logical Operators

Operator	Meaning	Example	Result
and	True if both conditions are true	(5>2) and (3>1)	True
or	True if at least one condition is true	(5>2) or (1>3)	True
not	Reverses the result	not(5>2)	False

Operator Precedence (order of evaluation)

Python follows this order, highest priority first: () Parentheses -> ** Exponent -> * / // % (left to right) -> + − (left to right). Example: 10 + 2 * 3 = 16, NOT 36, because multiplication happens before addition.

Worked Example — Dose per Body Weight

```
weight_kg = 60
dose_per_kg = 7.5
total_dose = weight_kg * dose_per_kg   # arithmetic operator
print("Total dose (mg):", total_dose)

is_safe = total_dose <= 500            # comparison operator
print("Within safe limit:", is_safe)
```

Input and Output Operations

Output — print()

The print() function displays output on the screen.

Code	Output
`print("Hello Pharmacist")`	Hello Pharmacist
`print("Dose:", 500, "mg")`	Dose: 500 mg

Input — input()

The input() function takes data typed by the user. It always returns a string, so numeric input must be type-cast.

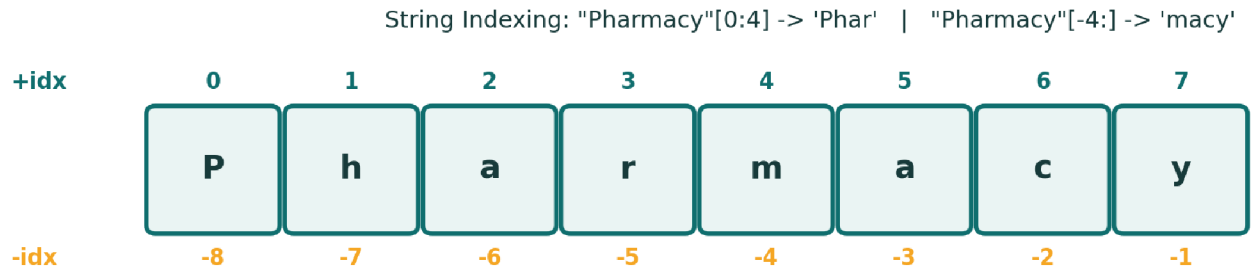
Code	Explanation
`name = input("Enter drug name: ")`	Stores text entered by user as a string
`dose = int(input("Enter dose: "))`	Converts the entered text into an integer

Complete Example — Input, Process, Output

```
name = input("Enter patient name: ")
weight = float(input("Enter weight in kg: "))
dose = weight * 10   # e.g., 10 mg per kg
print(name, "should receive", dose, "mg")
```


Basic String Operations and Manipulation

A string is a sequence of characters enclosed in single (' '), double (" "), or triple quotes. Strings in Python are indexed, allowing access to individual characters and substrings.



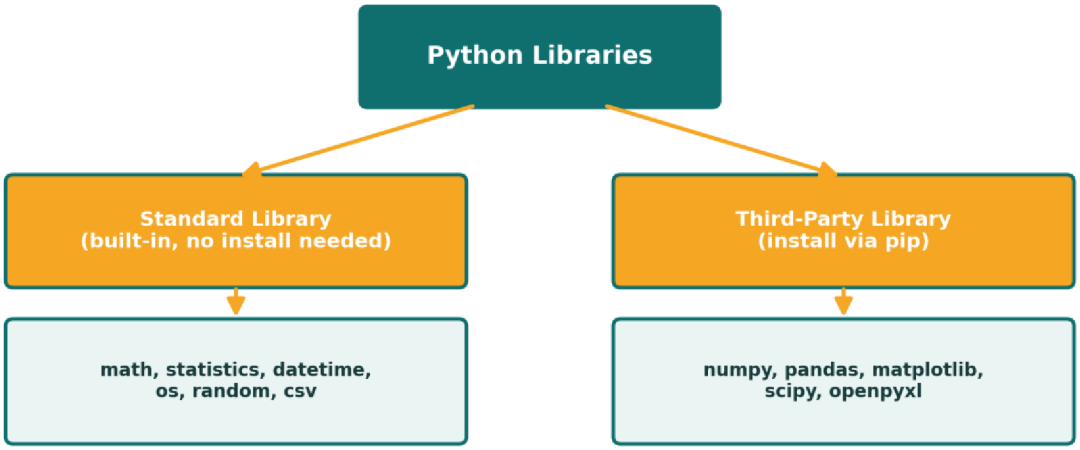
Operation	Example	Result
`Concatenation (+)`	"Para" + "cetamol"	"Paracetamol"
`Repetition (*)`	"Ab" * 3	"AbAbAb"

Operation	Example	Result
`Length – len()`	len("Aspirin")	7
`Uppercase – .upper()`	"aspirin".upper()	"ASPIRIN"
`Lowercase – .lower()`	"ASPIRIN".lower()	"aspirin"
`Strip spaces – .strip()`	" drug ".strip()	"drug"
`Replace – .replace()`	"Tab".replace("T","C")	"Cab"
`Split – .split()`	"a,b,c".split(",")	['a','b','c']
`Find – .find()`	"Pharmacy".find("a")	1
`Slicing – [start:end]`	"Pharmacy"[0:4]	"Phar"

⚠ Common Mistake

Python string indexing starts at 0, not 1. "Aspirin"[1] gives 's', not 'A'. Also, strings are immutable — you cannot change a character directly (word[0]='X' gives an error); instead create a new string using slicing or replace().

Standard Libraries vs Third-Party Libraries



Type	Definition	Examples	Installation
`Standard Library`	Comes bundled with Python installation	math, datetime, random, os, csv	No installation needed — just import
`Third-Party Library`	Created by external developers/community	numpy, pandas, matplotlib, scipy	Must be installed using pip

Installing and Uninstalling Libraries (pip)

pip (Pip Installs Packages) is Python's standard package manager used to install, upgrade, and remove third-party libraries from the command line/terminal.

Command	Purpose
`pip install numpy`	Installs the numpy library
`pip install pandas matplotlib`	Installs multiple libraries at once
`pip uninstall numpy`	Removes the numpy library
`pip list`	Shows all installed libraries
`pip show numpy`	Shows details/version of a library
`pip install --upgrade numpy`	Upgrades a library to the latest version

□ Import Statement

After installation, a library must be imported to use it: import numpy as np — 'np' here is just a short alias used to call the library's functions.

Mini Case Study — Simple Dose Calculator Program

This short program brings together variables, data types, type casting, input/output, operators, and string formatting learned in this unit.

```
# Simple weight-based dose calculator

patient_name = input("Enter patient name: ")
weight_kg = float(input("Enter weight (kg): "))
dose_per_kg = 5          # mg per kg (example value)

total_dose = weight_kg * dose_per_kg
is_within_limit = total_dose <= 250

print("Patient:", patient_name.upper())
print("Calculated dose:", round(total_dose, 2), "mg")
print("Safe to administer:", is_within_limit)
```

Sample Output:

```
Enter patient name: Ramesh
Enter weight (kg): 45
Patient: RAMESH
Calculated dose: 225.0 mg
Safe to administer: True
```

□ Concepts Used

input() & type casting (float), arithmetic operator (*), comparison operator (<=), string method (.upper()), built-in function round(), and print() with multiple values — a complete mini revision of this unit.

Important Points for Examination

- Python is an interpreted, high-level, dynamically typed language created by Guido van Rossum (1991).
- Always tick 'Add Python to PATH' while installing Python on Windows.
- IDE = Editor + Debugger + Run tools combined; gives auto-completion and error checking unlike plain text editors.
- Jupyter Notebook is best for step-by-step data analysis; PyCharm for large projects; VS Code is lightweight and flexible.
- Python has 4 basic numeric/text/logical types: int, float, str, bool — plus collections list, tuple, dict, set.
- Type casting functions: int(), float(), str(), bool().
- input() always returns a string — cast it to int()/float() for numeric use.

- Arithmetic operators: + − * / // % ** ; Comparison: == != > < >= <= ; Logical: and, or, not.
- Strings are indexed from 0 (left) and −1 (right, from the end); slicing syntax is [start:end].
- Standard library needs no installation; third-party libraries are installed using pip install <name>.
- pip is Python's package manager — install, uninstall, list, and upgrade libraries through it.