UnCodemy

# Python for Beginners PDF

Python is a beginner-friendly programming language. To start, install Python, run a first print statement, learn variables, numbers, text, lists, conditions, loops and functions, then write small programs. This page teaches those basics in the order a complete beginner needs them, with short examples you can type and run.

## Key takeaways

- Install Python and a code editor, then run your first program in under an hour.
- Learn in this order: variables, types, conditions, loops, lists, functions, files.
- Indentation is part of the syntax, so keep it consistent.
- Type every example instead of copying, and change it to see what happens.
- Small programs each day beat long sessions once a week.

## What is Python for beginners?

Python is a general-purpose programming language known for readable syntax, and it is widely used for scripting, data analysis, automation, web development and machine learning.

## How do I start?

Download Python from the official site, install it and open a terminal or an editor such as IDLE or VS Code. Type `python --version` to confirm it works. Then write and run your first program.

```
print("Hello, Python")
```

## What are variables and data types?

A variable is a name that stores a value. The main types are `int` (whole numbers), `float` (decimals), `str` (text) and `bool` (True or False).

```
age = 21
price = 99.5
name = "Asha"
is_student = True
print(name, age, price, is_student)
print(type(price))
```

Use `input()` to read text from the user and convert it with `int()` or `float()` when you need a number.

## How do I make decisions?

Use `if`, `elif` and `else`. Python uses indentation to show which lines belong to a block.

```
marks = 72
if marks >= 80:
    grade = "A"
elif marks >= 60:
    grade = "B"
else:
    grade = "C"
print(grade)
```

## How do loops work?

A `for` loop repeats over a sequence, and a `while` loop repeats while a condition is true.

```
total = 0
for n in range(1, 6):
    total += n
print(total)
```

`range(1, 6)` gives 1 to 5. Use `break` to stop a loop early and `continue` to skip to the next pass.

## What are lists and dictionaries?

A list holds an ordered collection, and a dictionary stores key-value pairs.

```
marks = [72, 85, 64]
marks.append(90)
print(len(marks), max(marks), sum(marks) / len(marks))

student = {"name": "Asha", "city": "Delhi"}
student["age"] = 21
print(student["name"], list(student.keys()))
```

## How do I write a function?

A function packages a task so you can reuse it.

```
def average(numbers):
    return sum(numbers) / len(numbers)

print(average([72, 85, 64, 90]))
```

Give functions clear names and keep each one focused on a single job.

## How do I work with text?

Strings have helpful methods such as `upper()`, `strip()`, `split()` and `replace()`.

```
text = "  data analytics  "
print(text.strip().title())
print("data analytics".split())
```

## How do I read and write files?

Use `with open(...)` so that the file closes automatically.

```
with open("notes.txt", "w") as f:
    f.write("first line")
with open("notes.txt") as f:
    print(f.read())
```

## What mistakes do beginners make?

Mixing tabs and spaces, forgetting the colon after `if` or `for`, confusing `=` with `==`, using a variable before assigning it and reading error messages from the top instead of the last line. Read the last line of an error first; it names the problem.

## What should I build first?

A number guessing game, a simple calculator, a to-do list stored in a file, a word counter and a marks report. Each uses only the concepts above and teaches you to break a problem into steps.

## How do I read error messages?

Python prints a traceback, a list of the calls that led to the error. Read the last line first: it gives the error type and a short message, such as `NameError: name 'total' is not defined`. Then look at the line number just above it. Most beginner errors are typing mistakes, missing colons, wrong indentation or using a variable before it exists. Fix one error at a time and run again.

## How do I practise every day?

Set a thirty-minute habit. On day one write a program that prints your name and age. On day two use a loop to print a table of five. On day three build a list of marks and compute the average. Each day add one small feature to something you already wrote. This builds confidence faster than watching long videos.

## What should I learn after the basics?

Learn modules and packages, error handling with `try` and `except`, classes and objects, and then the library for your goal: pandas for data, Flask or Django for the web, or Selenium for automation. Reading other people's short programs also improves your style.

## How do I find help?

Search the exact error message, read the official tutorial and ask in a community with a short example that reproduces the problem. Explaining your problem clearly often reveals the answer.

## What are good habits from day one?

Use clear variable names such as `total_marks` instead of `x`, add a comment where the reason is not obvious, keep each function short and run your code often rather than writing fifty lines before testing. Save every practice program in one folder with a date in the file name so you can see your progress.

## Frequently asked questions

### Is Python good for beginners?

Yes. Its readable syntax and large community make it one of the common first programming languages.

### How long does it take to learn Python basics?

Many beginners cover the basics in a few weeks of regular practice, though speed varies with time and background.

### What should I learn first in Python?

Start with variables, data types, conditions, loops, lists, functions and file handling, in that order.

### Do I need to install anything?

You need Python and a code editor. Online notebooks also work if you cannot install software.

### What should I build as a beginner?

Try a calculator, a guessing game, a to-do list saved to a file and a word counter.

## Sources and further reading

- Python documentation
- The Python Tutorial

## About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550