PYTHON

Beginner to Advanced Notes

12 modules · tested code with real output · 73-topic checklist
5 classic mistakes with exact errors · roadmap



WWW.TECHPROFREE.COM

TechProFree Edition · 100% free, made by us

The Roadmap — Beginner → Intermediate → Advanced

Stage	Modules	Time (1 hr/day)	By the end you can...
Beginner	1–6: setup, variables, conditions, loops, strings, data structures	3–4 weeks	Build calculators, quizzes, number games
Intermediate	7–9: functions, files, error handling, modules	3–4 weeks	Build tools that save data and survive bad input
Advanced-ready	10–12: OOP, pip, comprehensions, generators, decorators	4–6 weeks	Read framework code, start data science or web

The only rule that matters: type every example yourself, break it on purpose, fix it. Ten typed-and-broken programs teach more than a hundred watched tutorials.

Complete Topics Checklist — 73 Topics

BEGINNER (25)
[ ] Installing Python & PATH
[ ] Running .py / IDLE / VS Code
[ ] print() & comments
[ ] Variables & naming rules
[ ] Types: int, float, str, bool
[ ] Type conversion
[ ] input() — returns str!
[ ] f-strings & formatting
[ ] Arithmetic operators
[ ] Comparison operators
[ ] Logical operators
[ ] if / elif / else
[ ] Nested conditions
[ ] for loops & range()
[ ] while loops
[ ] break & continue
[ ] String methods
[ ] String slicing
[ ] Lists: append, remove
[ ] List slicing
[ ] Tuples & unpacking
[ ] Sets & uniqueness
[ ] Dictionaries & get()
[ ] Looping collections
[ ] len, min, max, sum

INTERMEDIATE (24)
[ ] Functions: def, return
[ ] Defaults & keyword args
[ ] *args & **kwargs
[ ] Variable scope
[ ] List comprehensions
[ ] Dict/set comprehensions
[ ] lambda functions
[ ] map, filter, sorted(key)
[ ] File modes w/r/a
[ ] with statement
[ ] CSV files
[ ] try/except/else/finally
[ ] Raising exceptions
[ ] Common exceptions
[ ] Modules & imports
[ ] Standard library tour
[ ] pip & packages
[ ] Virtual environments
[ ] requests basics
[ ] JSON loads & dumps
[ ] Date parsing (strptime)
[ ] enumerate & zip
[ ] Ternary expressions
[ ] Docstrings & help()

ADVANCED (24)
[ ] Classes & objects
[ ] __init__ & self
[ ] Instance vs class attrs
[ ] Method overriding
[ ] Inheritance & super()
[ ] Polymorphism
[ ] Dunder methods
[ ] Generators & yield
[ ] Iterators
[ ] Decorators
[ ] Closures
[ ] Type hints
[ ] Regular expressions
[ ] pathlib
[ ] logging
[ ] Testing (pytest)
[ ] Large files efficiently
[ ] timedelta math
[ ] Counter & defaultdict
[ ] itertools essentials
[ ] SQLite basics
[ ] Project structure
[ ] Reading documentation
[ ] Next: pandas/Flask

The 12 Modules — Notes, Code & Real Output

Every code block below was executed before this PDF was made; the output shown is the real output. Reading is 20%, typing is 80%.

Module 1: Setup & Your First Program

Install Python from python.org (tick "Add to PATH" on Windows), open any editor and run your first line. If this prints, your setup works.

CODE

```
print("Hello, TechProFree!")
print("Python version check done - you are ready.")
```

OUTPUT (REAL)

```
Hello, TechProFree!
Python version check done - you are ready.
```

Practice: Change the message, run again — you just edited a program.

Module 2: Variables, Data Types & Input

Variables are labeled boxes; Python infers the type. f-strings drop variables into text. Tattoo this: input() ALWAYS returns a string.

CODE

```
name = "Ahmed"
age = 21
height = 5.6
is_student = True

print(type(name), type(age), type(height), type(is_student))
print(f"{name} is {age} years old.")
```

OUTPUT (REAL)

```
<class 'str'> <class 'int'> <class 'float'> <class 'bool'>
Ahmed is 21 years old.
```

Practice: Take a name and birth year with input(), print the age (int() first).

Module 3: Operators & If/Else

if/elif/else runs the first true branch. // floor-divides, % gives remainder, ** is power.

CODE

```
marks = 78

if marks >= 80:
    grade = "A"
elif marks >= 70:
    grade = "B"
elif marks >= 60:
    grade = "C"
else:
    grade = "F"

print("Marks:", marks, "| Grade:", grade)
print(10 // 3, 10 % 3, 2 ** 5)
```

OUTPUT (REAL)

```
Marks: 78 | Grade: B
3 1 32
```

Practice: Write a grading program for your marks scheme; add an A+ branch.

Module 4: Loops — for & while

for i in range(1,6) counts 1–5 (range stops BEFORE the end). while repeats until false.

CODE

```
for i in range(1, 6):
    print(f"5 x {i} = {5*i}")

count = 3
while count > 0:
    print("Countdown:", count)
    count -= 1
print("Done!")
```

OUTPUT (REAL)

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
Countdown: 3
Countdown: 2
Countdown: 1
Done!
```

Practice: Print the 7-times table, then a countdown that ends with "Liftoff!"

Module 5: Strings in Depth

Change case, replace, measure, slice [start:end], membership with in. Strings are everywhere.

CODE

```
text = "python is easy"

print(text.upper())
print(text.title())
print(text.replace("easy", "powerful"))
print(len(text), "characters")
print(text[0:6])
print("easy" in text)
```

OUTPUT (REAL)

```
PYTHON IS EASY
Python Is Easy
python is powerful
14 characters
python
True
```

Practice: Print a sentence reversed (hint: slicing step — [::-1]).

Module 6: Lists, Tuples, Sets & Dicts

list = ordered+changeable, tuple = fixed, set = no duplicates, dict = key-value. Dicts run the modern world.

CODE

```
fruits = ["apple", "mango", "banana"]
point = (3, 5)
unique = {1, 2, 2, 3, 3}
student = {"name": "Sara", "marks": 88}

fruits.append("orange")
print(fruits)
print(point[0], point[1])
print(unique)
print(student["name"], "scored", student["marks"])
```

OUTPUT (REAL)

```
['apple', 'mango', 'banana', 'orange']
3 5
{1, 2, 3}
Sara scored 88
```

Practice: Build a dict about yourself (name, city, skills list) and print each part.

Module 7: Functions

def once, reuse forever. Parameters, return, and defaults (lang="en") make them flexible.

CODE

```
def greet(name, lang="en"):
    if lang == "ur":
        return f"Salaam, {name}!"
    return f"Hello, {name}!"

def area(length, width):
    return length * width

print(greet("Ali"))
print(greet("Ali", lang="ur"))
print("Area:", area(5, 3))
```

OUTPUT (REAL)

```
Hello, Ali!
Salaam, Ali!
Area: 15
```

Practice: Write bmi(weight_kg, height_m) returning BMI rounded to 1 decimal.

Module 8: File Handling

with open() reads/writes AND auto-closes even on errors. "w" overwrites, "a" appends, "r" reads.

CODE

```
with open("notes.txt", "w") as f:
    f.write("Line 1: Python file handling\n")
    f.write("Line 2: with open() auto-closes the file\n")

with open("notes.txt", "r") as f:
    for line in f:
        print(line.strip())
```

OUTPUT (REAL)

```
Line 1: Python file handling
Line 2: with open() auto-closes the file
```

Practice: Save 3 goals to goals.txt, read them back numbered.

Module 9: Error Handling — try/except

Catch the SPECIFIC error you expect; finally always runs. This is how programs survive users.

CODE

```
def safe_divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Cannot divide by zero!"
    finally:
        print(f"  (attempted {a} / {b})")

print(safe_divide(10, 2))
print(safe_divide(5, 0))
```

OUTPUT (REAL)

```
(attempted 10 / 2)
5.0
  (attempted 5 / 0)
Cannot divide by zero!
```

Practice: Wrap int(input()) in try/except; loop until valid.

Module 10: OOP — Classes & Inheritance

Classes bundle data (__init__) with behavior (methods). Cat(Animal) inherits and overrides — that's polymorphism.

CODE

```
class Animal:
    def __init__(self, name):
        self.name = name
    def speak(self):
        return f"{self.name} makes a sound"

class Cat(Animal):
    def speak(self):
        return f"{self.name} says Meow"

pets = [Animal("Rex"), Cat("Milo")]
for p in pets:
    print(p.speak())
```

OUTPUT (REAL)

```
Rex makes a sound
Milo says Meow
```

Practice: Add a Dog class saying Woof; give Animal an age shown in speak().

Module 11: Modules & pip

math, random, datetime: one import away. pip install opens 500,000+ packages.

CODE

```
import math
import random
from datetime import date

print("Square root of 144:", math.sqrt(144))
print("Random pick:", random.choice(["red", "green", "blue"]))
print("Today is:", date.today().strftime("%d %B %Y"))
```

OUTPUT (REAL)

```
Square root of 144: 12.0
Random pick: green
Today is: 15 November 2026
```

Practice: Build a two-dice roller with random; print the total.

Module 12: Comprehensions, Lambda, Generators, Decorators

One-line list building, throwaway sort keys, lazy sequences with yield, and @decorators wrapping functions.

CODE

```
squares = [n**2 for n in range(1, 6)]
print("Squares:", squares)

students = [("Ali", 85), ("Sara", 92), ("Bilal", 78)]
top = sorted(students, key=lambda s: s[1], reverse=True)
print("Topper:", top[0])

def countdown(n):
    while n > 0:
        yield n
        n -= 1
print("Generator:", list(countdown(5)))

def shout(func):
    def wrapper():
        return func().upper() + "!"
    return wrapper

@shout
def hello():
    return "hello from techprofree"
print(hello())
```

OUTPUT (REAL)

```
Squares: [1, 4, 9, 16, 25]
Topper: ('Sara', 92)
Generator: [5, 4, 3, 2, 1]
HELLO FROM TECHPROFREE!
```

Practice: Comprehension for evens 1–20; sort (product, price) tuples by price with lambda.

5 Classic Mistakes — With the Exact Errors

1. The missing indent

CODE

```
if True:
print("hi")
```

REAL RESULT

```
    print("hi")
    ^
IndentationError: expected an indented block after 'if' statement on line 1
```

Everything inside if/loops/functions must be indented (4 spaces).

2. = vs ==

CODE

```
x = 5
if x = 5:
    print("yes")
```

REAL RESULT

```
    if x = 5:
       ^^^^^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

One = assigns, two == compares. Python now suggests the fix.

3. input() returns a string

CODE

```
age = "21"
print(age + 5)
```

REAL RESULT

```
Traceback (most recent call last):
  File "<string>", line 2, in <module>
TypeError: can only concatenate str (not "int") to str
```

Convert before math: age = int(input("Age: "))

4. Mutable default argument

CODE

```
def add_item(item, bag=[]):
    bag.append(item)
    return bag

print(add_item("pen"))
print(add_item("book"))
```

REAL RESULT

```
['pen']
['pen', 'book']
```

The default list is created ONCE and shared. Fix: default to None, create inside.

5. Off-by-one with range()

CODE

```
print(list(range(5)))
print(list(range(1, 6)))
```

REAL RESULT

```
[0, 1, 2, 3, 4]
[1, 2, 3, 4, 5]
```

range(n) starts at 0, stops BEFORE n. Want 1–5? range(1, 6).

Legally Free Python Books (Official Links)

Book	Level	Link
Python for Everybody (PY4E)	Beginner	py4e.com/book
Automate the Boring Stuff	Beg–Inter	automatetheboringstuff.com
Think Python (2e)	Beg–Inter	greenteapress.com/wp/think-python-2e
A Byte of Python	Beginner	python.swaroopch.com
Python Notes for Professionals	Intermediate	goalkicker.com/PythonBook
Dive Into Python 3	Intermediate	diveintopython3.net
Official Python Tutorial	All levels	docs.python.org/3/tutorial

We don't host or link pirated copies of paid books. Everything above is officially free from its author or publisher. This PDF itself is original TechProFree content — share it freely.

Type it. Break it. Fix it. — that's the whole secret.

Next: pick your first project → techprofree.com/python-projects-beginner-to-advanced