THE PROBABILITY BOOTCAMP

Python Basics

with Project Showcase

Author:

Md. Farhan Ishraq

GitHub: github.com/farhanishraq17

LinkedIn: linkedin.com/in/farhan-ishraq-04643928a

Email: farhanishraq777@gmail.com

Date: October 6, 2025

Contents

1 Introduction

Python is a readable, expressive, and versatile high-level programming language used widely for teaching, scripting, automation, data analysis, web services, and more. This report covers core Python concepts with many examples, and ends with a Project Showcase of 15 small programs that demonstrate practical applications of those concepts.

2 Variables and Data Types

Variables are symbolic names bound to objects. Python is dynamically typed.

```
# 1. Integer
a = 10

# 2. Float
b = 3.1415

# 3. String
s = "IUT"

# 4. Boolean
flag = True

# 5. Multiple assignment
x, y, z = 1, 2, 3

# 6. Swapping
x, y = y, x

# 7. Type casting
num = int("42")
flt = float("3.14")

# 8. Complex
c = 2 + 3j

# 9. Byte string
bs = b'hello'

# 10. None
n = None
```

Listing 1: Examples: Variables and Data Types

3 Strings and String Manipulation

Strings are immutable sequences of Unicode characters.

```
# 1. Basic string
s = "Python"
```

```

# 2. Concatenation
s2 = s + " " + "3"

# 3. Repetition
print("ha" * 3)

# 4. Indexing
print(s[0], s[-1])

# 5. Slicing
print(s[1:4])

# 6. Length
print(len(s))

# 7. Format strings (f-strings)
name = "Farhan"
g = f"Hello, {name}"

# 8. split and join
words = "a b c".split()
joined = "-".join(words)

# 9. find and replace
t = "IUT is great"
print(t.find("IUT"))
print(t.replace("great", "awesome"))

# 10. raw strings and escapes
path = r"C:\Users\Farhan"
```

Listing 2: 10 String Examples

4 Operators

Categories: arithmetic, comparison, logical, bitwise, membership, identity, assignment.

```
a, b = 7, 2

# 1. +
print(a + b)

# 2. -
print(a - b)

# 3. *
print(a * b)

# 4. /
print(a / b)

```

```python
# 5. //
print(a // b)

# 6. **
print(a ** b)

# 7. %
print(a % b)

# 8. comparison
print(a >= b)

# 9. membership
print('t' in "python")

# 10. bitwise
print(a & b, a | b, a ^ b)
```

Listing 3: 10 Operator Examples

5 Conditional Statements

```python
# 1. Simple
x = 10
if x > 0:
    print("positive")

# 2. if-else
if x % 2 == 0:
    print("even")
else:
    print("odd")

# 3. if-elif-else ladder
score = 78
if score >= 90:
    grade = "A+"
elif score >= 80:
    grade = "A"
elif score >= 70:
    grade = "B"
else:
    grade = "C"

# 4. nested condition
if x > 0:
    if x < 100:
        print("0 < x < 100")

# 5. ternary
status = "pass" if score >= 50 else "fail"

```

```python
# 6. chained comparison
if 0 < x < 100:
    print("within range")

# 7. boolean short-circuit
a = None
res = a or "default"

# 8. membership check
if "apple" in ["apple", "banana"]:
    print("found")

# 9. identity
if a is None:
    print("a is None")

# 10. multiple condition
if x % 2 == 0 and x > 0:
    print("positive even")
```

Listing 4: 10 Examples of If/Elif/Else usage

6 Loops

```python
# 1. for range
for i in range(5):
    print(i)

# 2. for over list
nums = [2,3,5]
for n in nums:
    print(n)

# 3. while
i = 0
while i < 5:
    print(i)
    i += 1

# 4. break
for i in range(10):
    if i == 3:
        break
    print(i)

# 5. continue
for i in range(5):
    if i % 2 == 0:
        continue
    print(i)

# 6. nested loops
```

```python
for i in range(2):
    for j in range(2):
        print(i, j)

# 7. for-else
for i in range(3):
    print(i)
else:
    print("done")

# 8. enumerate
for idx, v in enumerate(["a","b"]):
    print(idx, v)

# 9. zip
for a,b in zip([1,2],[3,4]):
    print(a,b)

# 10. list iteration with index
for i in range(len(nums)):
    print(i, nums[i])
```

Listing 5: 10 Loop Examples

7 Functions

```python
# 1. simple
def hello():
    print("Hello")

# 2. with args
def add(a,b):
    return a + b

# 3. default args
def power(x, p=2):
    return x**p

# 4. variable args
def varargs(*args):
    return sum(args)

# 5. keyword args
def kw(**kwargs):
    return kwargs

# 6. returning multiple
def min_max(a,b):
    return min(a,b), max(a,b)

# 7. lambda
sq = lambda x: x*x
```

```python

# 8. higher-order
def apply(f, x):
    return f(x)

# 9. docstring
def f():
    """Example docstring"""
    pass

# 10. recursion
def fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)
```

Listing 6: 10 Function Examples

8 List/Dict/Set Comprehensions and Generators

```python
# 1. list comprehension
squares = [x*x for x in range(10)]

# 2. list with condition
evens = [x for x in range(20) if x%2==0]

# 3. nested comprehension
pairs = [(i,j) for i in range(3) for j in range(2)]

# 4. dict comprehension
d = {i: i*i for i in range(5)}

# 5. set comprehension
s = {x for x in range(10) if x%3==0}

# 6. generator expression
g = (x*x for x in range(5))
print(next(g))

# 7. use with sum
total = sum(x for x in range(100) if x%2)

# 8. conditional mapping
mapped = [x if x%2==0 else -x for x in range(6)]

# 9. flatten nested list
nested = [[1,2],[3,4]]
flat = [y for x in nested for y in x]

# 10. enumerate in comprehension
enumd = [(i,v) for i,v in enumerate(["a","b"])]
```

Listing 7: 10 Comprehension Examples

9 Collections (List, Tuple, Dict, Set, Collections module note)

```
# 1. list operations
lst = [1,2,3]
lst.append(4)
lst.pop()

# 2. slice assignment
lst[1:3] = [9,9]

# 3. tuple immutability
t = (1,2)
# t[0] = 5 -> TypeError

# 4. dict operations
d = {'a':1}
d['b'] = 2
print(d.get('z', 0))

# 5. dict iteration
for k,v in d.items():
    print(k,v)

# 6. set operations
a = {1,2,3}
b = {2,3,4}
print(a & b, a | b, a - b)

# 7. collections.Counter (note)
# from collections import Counter
# Counter(['a','b','a'])

# 8. OrderedDict & defaultdict note (py3.7+ dict keeps insertion order)

# 9. deque for fast pops from left
# from collections import deque
# q = deque([1,2,3]); q.popleft()

# 10. namedtuple note
# from collections import namedtuple
```

Listing 8: 10 Collections Examples

10 Input / Output and File Handling

```python
# 1. print formatting
print("Name:", "Farhan", "Age:", 21)

# 2. input (example)
# name = input("Enter name: ")

# 3. write file
with open("example.txt", "w") as f:
    f.write("Hello\n")

# 4. read file
with open("example.txt", "r") as f:
    data = f.read()

# 5. readlines
with open("example.txt") as f:
    lines = f.readlines()

# 6. append
with open("example.txt", "a") as f:
    f.write("More\n")

# 7. csv writing (basic)
import csv
with open("out.csv","w",newline="") as f:
    w = csv.writer(f); w.writerow(["a","b"])

# 8. json read/write
import json
obj = {"name":"Farhan"}
with open("data.json","w") as f:
    json.dump(obj,f)

# 9. binary file
with open("bin.dat","wb") as f:
    f.write(b'\x00\x01')

# 10. context manager ensures file closed
```

Listing 9: 10 File I/O Examples

11 Exception Handling

```python
# 1. try-except
try:
    x = 1/0
except ZeroDivisionError:
    print("div by zero")

# 2. multiple except
try:
```

```
    int("a")
except ValueError:
    print("bad int")
except Exception:
    print("other error")

# 3. else clause
try:
    x = int("5")
except:
    pass
else:
    print("converted")

# 4. finally
try:
    pass
finally:
    print("always runs")

# 5. raising exceptions
def f(x):
    if x < 0:
        raise ValueError("negative")

# 6. custom exception
class MyError(Exception):
    pass

# 7. exception chaining
try:
    pass
except Exception as e:
    raise MyError from e

# 8. catching multiple types
try:
    pass
except (TypeError, ValueError):
    pass

# 9. logging exceptions note (use logging module)
# 10. assert for debugging
assert 2+2==4
```

Listing 10: 10 Exception Handling Examples

12 Object-Oriented Programming

```
# 1. simple class
class Person:
    def __init__(self, name):
```

```
        self.name = name

# 2. method
    def greet(self):
        return f"Hello {self.name}"

# 3. inheritance
class Student(Person):
    def __init__(self,name,roll):
        super().__init__(name)
        self.roll = roll

# 4. overriding
    def greet(self):
        return f"Student {self.name}"

# 5. class variable
class C:
    counter = 0

# 6. __str__ and __repr__
class P:
    def __repr__(self):
        return "<P>"

# 7. property decorator
class Celsius:
    def __init__(self,temp=0):
        self._temp = temp
    @property
    def temp(self):
        return self._temp

# 8. staticmethod / classmethod
class X:
    @staticmethod
    def stat(): pass
    @classmethod
    def cls(cls): pass

# 9. dataclass note (from dataclasses import dataclass)

# 10. composition example: class A has B as attribute
```

Listing 11: 10 OOP Examples

13 Modules, Packages and Useful Standard Libraries

```
# 1. import module
import math
print(math.sqrt(16))

```

```
# 2. from import
from math import pi

# 3. alias
import numpy as np  # optional note

# 4. package structure note:
# mypkg/
#   __init__.py
#   mod.py

# 5. use of os
import os
print(os.listdir("."))

# 6. subprocess note
# import subprocess

# 7. use of datetime
import datetime
print(datetime.date.today())

# 8. use of random
import random
print(random.choice([1,2,3]))

# 9. helpful modules: itertools, functools, collections, json, csv, re

# 10. pip install third-party packages for extra functionality
```

Listing 12: Examples and Notes

14 Basic Plotting Note

This report only briefly mentions plotting. For plots, use `matplotlib.pyplot`. Example (not included as figure to keep PDF self-contained):

```
import matplotlib.pyplot as plt
x = [0,1,2,3]
y = [0,1,4,9]
plt.plot(x,y)
plt.title("Quadratic")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Listing 13: Simple Plot Example

15 Project Showcase

Below are 15 small but complete Python programs, each demonstrating application of the concepts covered. Each program is a standalone script that you can copy into a ‘.py‘ file and run.

15.1 a. Simple Calculator (CLI)

```python
# calculator.py
def add(a,b): return a+b
def sub(a,b): return a-b
def mul(a,b): return a*b
def div(a,b):
    if b==0: raise ValueError("Division by zero")
    return a/b

if __name__ == "__main__":
    a = float(input("a: "))
    b = float(input("b: "))
    op = input("op (+ - * /): ")
    if op == '+': print(add(a,b))
    elif op == '-': print(sub(a,b))
    elif op == '*': print(mul(a,b))
    elif op == '/': print(div(a,b))
    else: print("Unknown op")
```

Listing 14: Calculator CLI

15.2 b. Number Guessing Game

```python
# guess.py
import random
num = random.randint(1,100)
tries = 0
while True:
    tries += 1
    guess = int(input("Guess (1-100): "))
    if guess == num:
        print("Correct in", tries, "tries")
        break
    elif guess < num:
        print("Higher")
    else:
        print("Lower")
```

Listing 15: Number Guessing Game

15.3 c. Tic-Tac-Toe (2-player, terminal)

```python
# tictactoe.py
board = [" "]*9
def printb():
    for i in range(3):
        print(board[3*i:3*i+3])
def win(p):
    wins = [(0,1,2),(3,4,5),(6,7,8),(0,3,6),
            (1,4,7),(2,5,8),(0,4,8),(2,4,6)]
    return any(all(board[i]==p for i in w) for w in wins)

player = "X"
for turn in range(9):
    printb()
    pos = int(input(f"{player} move (0-8): "))
    if board[pos] != " ":
        print("Invalid")
        continue
    board[pos] = player
    if win(player):
        printb()
        print(player, "wins")
        break
    player = "O" if player=="X" else "X"
else:
    print("Draw")
```

Listing 16: Tic-Tac-Toe Terminal

15.4 d. Bank Management System (Simple Simulation)

```python
# bank.py
class Account:
    def __init__(self, name, bal=0):
        self.name = name; self.bal = bal
    def deposit(self, amt): self.bal += amt
    def withdraw(self, amt):
        if amt > self.bal: raise ValueError("Insufficient")
        self.bal -= amt
    def __str__(self): return f"{self.name}: {self.bal}"

if __name__=="__main__":
    a = Account("Farhan", 1000)
    a.deposit(500)
    try:
        a.withdraw(2000)
    except Exception as e:
        print("Error:", e)
    print(a)
```

Listing 17: Bank Management Simulation

15.5 e. To-Do List Manager (file-backed)

```python
# todo.py
import json, os
F="todo.json"
def load():
    if os.path.exists(F): return json.load(open(F))
    return []
def save(lst): json.dump(lst, open(F,"w"))
def add(task):
    lst=load(); lst.append(task); save(lst)
def list_tasks():
    for i,t in enumerate(load(),1): print(i,t)

if __name__=="__main__":
    add(input("Task: "))
    list_tasks()
```

Listing 18: To-Do List Manager

15.6 f. Dice Rolling Simulator

```python
# dice.py
import random
def roll(n=1,sides=6):
    return [random.randint(1,sides) for _ in range(n)]
if __name__=="__main__":
    print(roll(5))
```

Listing 19: Dice Simulator

15.7 g. Student Grade Analyzer

```python
# grades.py
grades = [85, 90, 72, 66, 95]
print("Average:", sum(grades)/len(grades))
print("Max:", max(grades))
print("Min:", min(grades))
print("Sorted:", sorted(grades, reverse=True))
```

Listing 20: Grade Analyzer

15.8 h. Password Generator

```python
# passgen.py
import random, string
def gen(n=12):
    chars = string.ascii_letters + string.digits + "!@#$%"
    return "".join(random.choice(chars) for _ in range(n))
```

```
6
7 if __name__=="__main__":
8     print(gen(16))
```

Listing 21: Password Generator

15.9 i. Simple ATM Simulation

```
1  # atm.py
2  accounts = {"123": {"pin":"0000","bal":500}}
3  def atm():
4      acc = input("Account: ")
5      if acc not in accounts: print("No account"); return
6      pin = input("PIN: ")
7      if pin != accounts[acc]["pin"]: print("Wrong PIN"); return
8      while True:
9          cmd = input("cmd (bal, dep, wd, exit): ")
10         if cmd == "bal": print(accounts[acc]["bal"])
11         elif cmd == "dep":
12             amt = float(input("amt: ")); accounts[acc]["bal"] += amt
13         elif cmd == "wd":
14             amt = float(input("amt: "))
15             if amt > accounts[acc]["bal"]: print("Insufficient")
16             else: accounts[acc]["bal"] -= amt
17         else: break
18
19 if __name__=="__main__":
20     atm()
```

Listing 22: ATM Simulation

15.10 j. Weather Mock Forecaster

```
1 # weather.py
2 import random
3 cities = ["Dhaka","Gazipur","Chattogram","Sylhet"]
4 for c in cities:
5     temp = random.uniform(15,35)
6     cond = random.choice(["Sunny","Cloudy","Rainy"])
7     print(f"{c}: {temp:.1f} C, {cond}")
```

Listing 23: Weather Mock Forecast

15.11 k. Quiz Game (Multiple Choice)

```
1 # quiz.py
2 qs = [
3     ("Capital of Bangladesh?","Dhaka"),
4     ("2+2?","4"),
```

```
]
score=0
for q,a in qs:
    ans = input(q+" ")
    if ans.strip().lower() == a.lower():
        print("OK"); score += 1
    else:
        print("Wrong, answer:", a)
print("Score:", score, "/", len(qs))
```

Listing 24: Quiz Game

15.12 l. Rock-Paper-Scissors

```
# rps.py
import random
opts = ["rock","paper","scissors"]
you = input("rock/paper/scissors: ").lower()
comp = random.choice(opts)
print("Computer:", comp)
if you == comp: print("Draw")
elif (you=="rock" and comp=="scissors") or \
     (you=="paper" and comp=="rock") or \
     (you=="scissors" and comp=="paper"):
    print("You win")
else:
    print("You lose")
```

Listing 25: Rock-Paper-Scissors

15.13 m. Simple Chatbot (rule-based)

```
# bot.py
while True:
    msg = input("You: ").lower()
    if msg in ("bye","exit"): print("Bot: Bye"); break
    if "hello" in msg: print("Bot: Hello!")
    elif "name" in msg: print("Bot: I am Bot")
    else: print("Bot: I don't understand")
```

Listing 26: Simple Chatbot

15.14 n. Thermometer Simulation (text output)

```
# thermo.py
import random, time
for _ in range(10):
    temp = random.uniform(15,40)
    bars = int((temp-10)/2)
```

```
    print(f"{temp:.1f} C |" + "#"*bars)
    time.sleep(0.5)
```

Listing 27: Thermometer Simulation

15.15 o. Simple File Encryption (XOR) for small files

```
# xorcrypt.py
def xor_bytes(data, key=0xAA):
    return bytes([b ^ key for b in data])

if __name__=="__main__":
    fn = input("file: ")
    with open(fn, "rb") as f:
        data = f.read()
    enc = xor_bytes(data)
    with open(fn + ".enc", "wb") as g:
        g.write(enc)
    print("Written", fn + ".enc")
```

Listing 28: Simple XOR File Encrypt/Decrypt

16 Conclusion

This document provided a practical and example-rich introduction to Python fundamentals and presented 15 runnable small projects. They are intended to be extended: add data persistence, GUI, network features, or database backends to build them into larger systems.

References

[1] Python Software Foundation. *Python Documentation*. Available at: https://docs.python.org/3/

[2] Al Sweigart. *Automate the Boring Stuff with Python*.

[3] Real Python. *Beginner's Guide to Python*. Available at: https://realpython.com/