Practice Questions from Unit 1 to Unit 6

---

## UNIT I – Basics, Variables, Expressions & I/O

### Q1. Print a Welcome Message (Easy)

Write a program to print Hello, Python!.

**Solution**

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

---

### Q2. Swap Two Numbers (Easy)

Take two integers from user and swap them using a temporary variable.

**Solution**

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))

temp = a
a = b
b = temp

print("After swap: a =", a, "b =", b)
```

---

### Q3. Area of Circle (Easy–Medium)

Read radius from user and print area of circle ($\pi r^2$, use 3.14).

**Solution**

```
r = float(input("Enter radius: "))
area = 3.14 * r * r
print("Area of circle:", area)
```

---

**Q4. Evaluate Expression (Medium)**

Take three numbers a, b, c and compute result = (a + b) ** 2 - 4 * a * c.

**Solution**

```
a = float(input("a: "))

b = float(input("b: "))

c = float(input("c: "))


result = (a + b) ** 2 - 4 * a * c

print("Result:", result)
```

---

**Q5. Check Type of Variable (Easy)**

Take input from user and display its value and type.

**Solution**

```
x = input("Enter something: ")

print("Value:", x)

print("Type:", type(x))
```

---

**Q6. Simple Interest (Medium)**

Input principal, rate, time and print simple interest = PRT/100.

**Solution**

```
p = float(input("Principal: "))

r = float(input("Rate: "))

t = float(input("Time in years: "))


si = p * r * t / 100
```

```
print("Simple Interest:", si)
```

---

**Q7. Celsius to Fahrenheit (Easy)**

Convert temperature from Celsius to Fahrenheit.

**Solution**

```
c = float(input("Temperature in Celsius: "))

f = (9/5) * c + 32

print("Fahrenheit:", f)
```

---

**Q8. Use of Comments (Easy)**

Write a program that adds two numbers and uses at least two comments.

**Solution**

```
# Program to add two numbers

x = int(input("Enter first number: "))

y = int(input("Enter second number: "))


# Add the numbers

s = x + y

print("Sum:", s)
```

---

**Q9. Script vs Interactive (Medium – thinking)**

Write a program that reads your name and age and prints Name: <name>, Age: <age>.
(Explain: This is a script, not interactive shell usage.)

**Solution**

```
name = input("Enter name: ")

age = int(input("Enter age: "))
```

```
print("Name:", name, "Age:", age)
```

(*Teacher note: running as .py file = script mode*.)

---

**Q10. Name Error Demonstration (High – concept)**

Write a program that intentionally creates a NameError and then correct it.

**Solution**

```
# Wrong code causing NameError
# print(total)   # total is not defined


# Corrected code
total = 10
print(total)
```

---

**UNIT II – Conditionals, Loops, Boolean, Random**

**Q1. Even or Odd (Easy)**

Read an integer and print whether it is even or odd.

**Solution**

```
n = int(input("Enter number: "))
if n % 2 == 0:
   print("Even")
else:
   print("Odd")
```

---

**Q2. Maximum of Three Numbers (Easy–Medium)**

Find largest among three numbers using if–elif–else.

**Solution**

```
a = int(input("a: "))
b = int(input("b: "))
c = int(input("c: "))

if a >= b and a >= c:
    print("Largest:", a)
elif b >= c:
    print("Largest:", b)
else:
    print("Largest:", c)
```

---

**Q3. Grade of Student (Medium)**

Input marks (0–100) and print grade:
90–100: A, 80–89: B, 70–79: C, 60–69: D, else: F.

**Solution**

```
m = int(input("Marks: "))

if m >= 90:
    grade = "A"
elif m >= 80:
    grade = "B"
elif m >= 70:
    grade = "C"
elif m >= 60:
    grade = "D"
else:
    grade = "F"

print("Grade:", grade)
```

---

**Q4. Sum of First n Natural Numbers (Easy)**

Use for loop to compute sum of 1…n.

**Solution**

```
n = int(input("n: "))
s = 0
for i in range(1, n+1):
    s += i
print("Sum:", s)
```

---

**Q5. Multiplication Table (Easy–Medium)**

Print multiplication table of a number up to 10.

**Solution**

```
n = int(input("Enter number: "))
for i in range(1, 11):
    print(n, "x", i, "=", n * i)
```

---

**Q6. Count Digits of a Number (Medium)**

Use while loop to count digits in an integer.

**Solution**

```
n = int(input("Enter number: "))
count = 0
temp = abs(n)
```

```
while temp > 0:
    count += 1
    temp //= 10
print("Number of digits:", count)
```

---

**Q7. Menu-Driven Calculator (Medium–High)**

Create menu: 1.Add 2.Subtract 3.Multiply 4.Divide. Use while loop until user chooses exit.

**Solution**

```
while True:
    print("1.Add  2.Subtract  3.Multiply  4.Divide  5.Exit")
    ch = int(input("Choice: "))
    if ch == 5:
        break
    a = float(input("a: "))
    b = float(input("b: "))

    if ch == 1:
        print("Result:", a + b)
    elif ch == 2:
        print("Result:", a - b)
    elif ch == 3:
        print("Result:", a * b)
    elif ch == 4:
        if b != 0:
            print("Result:", a / b)
        else:
```

```
        print("Division by zero not allowed")
    else:
        print("Invalid choice")
```

---

**Q8. Number Guessing Game using random (High)**

Generate random number 1–10. User has 3 chances to guess.

**Solution**

```
import random


secret = random.randint(1, 10)
attempts = 3


while attempts > 0:
    guess = int(input("Guess (1-10): "))
    if guess == secret:
        print("Correct!")
        break
    elif guess < secret:
        print("Too low")
    else:
        print("Too high")
    attempts -= 1


if attempts == 0 and guess != secret:
    print("Out of attempts. Number was", secret)
```

---

**Q9. Print Prime Numbers in Range (High)**

Print all prime numbers between 2 and n using nested loops.

**Solution**

```
n = int(input("Upper limit: "))
for num in range(2, n+1):
    is_prime = True
    for i in range(2, int(num**0.5)+1):
        if num % i == 0:
            is_prime = False
            break
    if is_prime:
        print(num, end=" ")
```

---

**Q10. Pattern Printing (Medium–High)**

For n=4, print:

```
*
* *
* * *
* * * *
```

**Solution**

```
n = int(input("Rows: "))
for i in range(1, n+1):
    print("* " * i)
```

---

## UNIT III – Functions & Recursion, Math, Type Conversion

**Q1. Simple Function (Easy)**

Define function greet(name) that prints Hello, <name>.

**Solution**

```
def greet(name):
    print("Hello,", name)


greet("Riya")
```

---

**Q2. Function with Default Argument (Easy)**

def power(a, b=2) returns a raised to power b.

**Solution**

```
def power(a, b=2):
    return a ** b


print(power(5))     # 25
print(power(2, 3))  # 8
```

---

**Q3. Sum of List using Function (Easy–Medium)**

Write function list_sum(lst) that returns sum of elements.

**Solution**

```
def list_sum(lst):
    s = 0
    for x in lst:
        s += x
    return s
```

```
print(list_sum([1, 2, 3, 4]))
```

---

**Q4. Use of math Module (Medium)**

Write function that takes radius and returns area and circumference.

**Solution**

```
import math


def circle_stats(r):
    area = math.pi * r**2
    circ = 2 * math.pi * r
    return area, circ


a, c = circle_stats(3)
print("Area:", a, "Circumference:", c)
```

---

**Q5. Type Conversion Function (Easy–Medium)**

Write function to_int_list(str_list) that receives list of strings and returns list of integers.

**Solution**

```
def to_int_list(str_list):
    result = []
    for s in str_list:
        result.append(int(s))
    return result


print(to_int_list(["10", "20", "30"]))
```

---

**Q6. Recursive Factorial (Medium)**

Write recursive function fact(n).

**Solution**

```
def fact(n):
    if n <= 1:
        return 1
    return n * fact(n-1)

print(fact(5))
```

---

**Q7. Recursive Fibonacci (Medium–High)**

Return nth Fibonacci number using recursion.

**Solution**

```
def fib(n):
    if n <= 1:
        return n
    return fib(n-1) + fib(n-2)

print(fib(6))   # 8
```

---

**Q8. Lambda with map (Medium)**

Given list [1,2,3,4], create new list of squares using lambda.

**Solution**

```
nums = [1, 2, 3, 4]
squares = list(map(lambda x: x**2, nums))
```

```
print(squares)
```

---

**Q9. Higher-Order Function (High)**

Write function apply_twice(f, x) that applies function f two times.

**Solution**

```
def apply_twice(f, x):
    return f(f(x))


def inc(n):
    return n + 1


print(apply_twice(inc, 5))   # 7
```

---

**Q10. Recursive Sum of Digits (High)**

Write recursive function that returns sum of digits of an integer.

**Solution**

```
def sum_digits(n):
    n = abs(n)
    if n == 0:
        return 0
    return n % 10 + sum_digits(n // 10)


print(sum_digits(1234))   # 10
```

---

**UNIT IV – Strings, Lists, Tuples, Dictionaries, Matrices**

**Q1. Count Vowels in String (Easy)**

Count number of vowels in a string.

**Solution**

```
s = input("Enter string: ").lower()

count = 0

for ch in s:

    if ch in "aeiou":

        count += 1

print("Vowels:", count)
```

---

**Q2. Check Palindrome String (Easy–Medium)**

Check if given string is palindrome (ignore case).

**Solution**

```
s = input("Enter string: ").lower()

if s == s[::-1]:

    print("Palindrome")

else:

    print("Not palindrome")
```

---

**Q3. Find Substring (Medium)**

Input string and substring; print first index if found, else -1 (without using find).

**Solution**

```
s = input("Main string: ")

sub = input("Substring: ")
```

```
pos = -1
for i in range(len(s) - len(sub) + 1):
    if s[i:i+len(sub)] == sub:
        pos = i
        break

print("Position:", pos)
```

---

**Q4. List Operations (Easy–Medium)**

Given list of integers, remove all even numbers.

**Solution**

```
nums = [1, 2, 3, 4, 5, 6]
result = []
for x in nums:
    if x % 2 != 0:
        result.append(x)
print(result)
```

---

**Q5. List of Squares (Easy)**

Create list containing squares of numbers 1–10 using list comprehension.

**Solution**

```
squares = [i*i for i in range(1, 11)]
print(squares)
```

---

**Q6. Nested List: Matrix Addition (Medium–High)**

Add two 2×2 matrices.

**Solution**

```
A = [[1, 2],
     [3, 4]]
B = [[5, 6],
     [7, 8]]

C = [[0, 0], [0, 0]]

for i in range(2):
    for j in range(2):
        C[i][j] = A[i][j] + B[i][j]

print(C)
```

---

**Q7. Tuple as Return Value (Medium)**

Write function that takes two numbers and returns (sum, difference, product) as tuple.

**Solution**

```
def operations(a, b):
    return a+b, a-b, a*b

res = operations(10, 4)
print(res)
```

---

**Q8. Dictionary Word Count (Medium–High)**

Count occurrences of each word in a sentence and store in dictionary.

**Solution**

```
text = input("Enter sentence: ").lower()
words = text.split()
freq = {}

for w in words:
    if w in freq:
        freq[w] += 1
    else:
        freq[w] = 1

print(freq)
```

---

**Q9. Dictionary Operations (Medium)**

Store names and marks of 3 students in a dictionary and print name of student with highest marks.

**Solution**

```
marks = {}
for i in range(3):
    name = input("Name: ")
    m = int(input("Marks: "))
    marks[name] = m

topper = max(marks, key=marks.get)
print("Topper:", topper, "Marks:", marks[topper])
```

---

**Q10. Copy vs Alias in Lists (High – concept)**

Demonstrate aliasing and copying of a list.

**Solution**

```
lst1 = [1, 2, 3]
alias = lst1          # aliasing
copy_lst = lst1[:]    # shallow copy

alias[0] = 100
print("lst1:", lst1)         # changed
print("alias:", alias)       # same as lst1
print("copy_lst:", copy_lst) # unchanged
```

---

**UNIT V – Classes, Objects, Inheritance, OOP Concepts**

**Q1. Simple Class (Easy)**

Create class Student with attributes name and age and method display().

**Solution**

```
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display(self):
        print("Name:", self.name, "Age:", self.age)

s1 = Student("Riya", 20)
s1.display()
```

---

**Q2. Class with Method for Area of Rectangle (Easy–Medium)**

Create class Rectangle with length, breadth and method area().

**Solution**

```
class Rectangle:
    def __init__(self, l, b):
        self.l = l
        self.b = b

    def area(self):
        return self.l * self.b

r = Rectangle(5, 4)
print("Area:", r.area())
```

---

**Q3. Encapsulation using Private Attribute (Medium)**

Create class BankAccount with private attribute __balance and methods deposit, withdraw, get_balance.

**Solution**

```
class BankAccount:
    def __init__(self, balance=0):
        self.__balance = balance

    def deposit(self, amt):
        self.__balance += amt

    def withdraw(self, amt):
        if amt <= self.__balance:
            self.__balance -= amt
        else:
            print("Insufficient balance")

    def get_balance(self):
        return self.__balance

acc = BankAccount(1000)
acc.deposit(500)
acc.withdraw(200)
print("Balance:", acc.get_balance())
```

---

**Q4. Single Inheritance (Medium)**

Create base class Person and derived class Employee adding salary.

**Solution**

```
class Person:
    def __init__(self, name):
        self.name = name

    def show(self):
        print("Name:", self.name)

class Employee(Person):
    def __init__(self, name, salary):
        super().__init__(name)
        self.salary = salary

    def show(self):
        super().show()
        print("Salary:", self.salary)

e = Employee("Amit", 50000)
e.show()
```

---

**Q5. Method Overriding (Medium–High)**

Using previous example, show that show() is overridden in Employee.

*(Already shown in Q4: Employee.show overrides Person.show.)*

---

**Q6. Multiple Constructors using Default Arguments (Function Overloading Style) (Medium)**

Create class Point that can be created as Point() or Point(x, y).

**Solution**

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

p1 = Point()
p2 = Point(3, 4)
print(p1.x, p1.y)
print(p2.x, p2.y)
```

---

**Q7. Operator Overloading (__add__) (High)**

Overload + to add two Point objects.

**Solution**

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)


p1 = Point(1, 2)
p2 = Point(3, 4)
p3 = p1 + p2
print(p3.x, p3.y)
```

---

**Q8. Class Variable vs Instance Variable (Medium–High)**

Create class Counter with class variable count that tracks how many objects are created.

**Solution**

```
class Counter:
    count = 0  # class variable

    def __init__(self):
        Counter.count += 1


c1 = Counter()
```

```
c2 = Counter()
c3 = Counter()
print("Objects created:", Counter.count)
```

---

**Q9. Data Hiding Example (Medium)**

Show that private variable cannot be accessed directly from object.

**Solution**

```
class Sample:
    def __init__(self):
        self.__secret = 10


s = Sample()
# print(s.__secret)   # AttributeError
print(s._Sample__secret)  # name-mangled access (not recommended)
```

---

**Q10. Polymorphism using Common Interface (High)**

Classes Cat and Dog each having speak() method. Write function that calls speak() for any animal object.

**Solution**

```
class Dog:
    def speak(self):
        print("Woof")


class Cat:
    def speak(self):
        print("Meow")
```

```
def animal_sound(animal):
    animal.speak()


d = Dog()
c = Cat()
animal_sound(d)
animal_sound(c)
```

---

**UNIT VI – Files, Exceptions, Regular Expressions, Web Scraping Basics**

**Q1. Write to Text File (Easy)**

Write program to create file sample.txt and write a line into it.

**Solution**

```
f = open("sample.txt", "w")
f.write("This is a sample file.\n")
f.close()
```

---

**Q2. Read File Line by Line (Easy–Medium)**

Read and display contents of sample.txt.

**Solution**

```
f = open("sample.txt", "r")
for line in f:
    print(line, end="")
f.close()
```

---

**Q3. Count Lines in File (Medium)**

Count number of lines in a text file.

**Solution**

```
filename = "sample.txt"

count = 0

with open(filename, "r") as f:

    for _ in f:

        count += 1

print("Lines:", count)
```

---

**Q4. Handle File Not Found Exception (Medium)**

Try to open a file given by user; if it doesn't exist, print proper message.

**Solution**

```
fname = input("Filename: ")

try:

    with open(fname, "r") as f:

        print(f.read())

except FileNotFoundError:

    print("File does not exist")
```

---

**Q5. Division with Exception Handling (Easy–Medium)**

Take two numbers and divide, handle ZeroDivisionError.

**Solution**

```
try:

    a = int(input("a: "))

    b = int(input("b: "))

    print("Result:", a / b)
```

```
except ZeroDivisionError:
    print("Cannot divide by zero")
```

---

**Q6. try–except–else–finally (Medium–High)**

Demonstrate all four parts.

**Solution**

```
try:
    n = int(input("Enter integer: "))
except ValueError:
    print("Not an integer")
else:
    print("You entered:", n)
finally:
    print("Program ended")
```

---

**Q7. Simple Regex – Validate Mobile Number (Medium)**

Check if given string is a valid 10-digit number using regex.

**Solution**

```
import re

s = input("Enter mobile number: ")
pattern = r"^[0-9]{10}$"

if re.match(pattern, s):
    print("Valid")
else:
    print("Invalid")
```

---

**Q8. Find All Email IDs in Text (High)**

Use regex to extract all email IDs from a given text string.

**Solution**

```
import re


text = "Contact: abc@example.com and info@test.org"

pattern = r"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"


emails = re.findall(pattern, text)

print(emails)
```

---

**Q9. Simple Web Page Text Extraction (High – conceptual)**

Assume HTML stored in string html = "<h1>Title</h1><p>Hello World</p>". Extract content between <p> and </p> using regex.

**Solution**

```
import re


html = "<h1>Title</h1><p>Hello World</p>"

match = re.search(r"<p>(.*?)</p>", html)

if match:

    print("Paragraph:", match.group(1))
```

---

**Q10. Copy Content from One File to Another (Medium)**

Read from source.txt and write to target.txt.

**Solution**

```
with open("source.txt", "r") as src, open("target.txt", "w") as tgt:
    for line in src:
        tgt.write(line)
print("Copied successfully")
```

---