CampusPe

Python Basics Assignment 02

Complete Solutions — Modules 1–6

All 20 questions answered with clear comments, meaningful variable names, edge-case handling, and beginner-friendly logic.

Q#	Topic	Difficulty	Marks
1	Personal Bio Card	Easy	2
2	Simple Calculator	Easy	2
3	String Manipulator	Medium	4
4	Age Calculator	Medium	4
5	Bill Splitter	Medium	4
6	Grade Calculator	Easy-Medium	4
7	Temperature Converter	Medium	4
8	Leap Year Checker	Medium	4
9	Ticket Pricing System	Medium	4
10	Simple ATM Simulator	Medium-Hard	7
11	Number Pattern Printer	Medium	4
12	Multiplication Table Generator	Easy-Medium	4
13	Sum and Average Calculator	Easy	4
14	Factorial Calculator	Medium	4
15	Prime Number Checker	Medium-Hard	7
16	Number Guessing Game	Hard	7
17	Palindrome Checker	Medium	4
18	Calculator Functions	Medium	7
19	Text Analysis Functions	Hard	9
20	Number System Functions	Hard	9
	TOTAL		**100**

Q1: Personal Bio Card

Difficulty: Easy | Marks: 2

```
Q1: Personal Bio Card
# We store each piece of information in a clearly named variable.
# Then we print everything inside a formatted box using string formatting.

# Step 1: Store student information in variables
student_name= "John Doe"
student_age= 20
student_course = "Python Programming"
student_college= "ABC University"
student_email  = "john@example.com"

# Step 2: Define the box width so all rows look the same
BOX_WIDTH = 36   # total inner width (characters)

# Step 3: Helper function to build one row of the card
def make_row(label, value):
    # Format: "  Label : Value" padded to BOX_WIDTH, then surrounded by ■
    content = f"  {label:<8}: {value}"
    # Pad the content so all rows are the same length
    content = content.ljust(BOX_WIDTH)
    return f"■{content}■"

# Step 4: Print the bio card
print("■" + "■" * BOX_WIDTH + "■")
print("■" + " STUDENT BIO CARD".center(BOX_WIDTH) + "■")
print("■" + "■" * BOX_WIDTH + "■")
print(make_row("Name",     student_name))
print(make_row("Age",      f"{student_age} years"))
print(make_row("Course",  student_course))
print(make_row("College", student_college))
print(make_row("Email",    student_email))
print("■" + "■" * BOX_WIDTH + "■")
```

Q2: Simple Calculator

Difficulty: Easy | Marks: 2

```
Q2: Simple Calculator
# Ask the user for two numbers, then perform and display all 6 operations.
# We use try-except to handle cases where the user types non-numeric input.

# Step 1: Get input from the user (inside try-except for safety)
try:
    first_number  = float(input("Enter first number: "))
    second_number = float(input("Enter second number: "))
except ValueError:
    print("Invalid input! Please enter numbers only.")
    exit()

# Step 2: Perform all the required arithmetic operations
addition       = first_number + second_number
subtraction    = first_number - second_number
multiplication = first_number * second_number
modulus        = first_number % second_number

# Step 3: Handle division carefully (cannot divide by zero)
if second_number == 0:
    division_result = "Error (division by zero)"
else:
    division_result = round(first_number / second_number, 2)

exponentiation = first_number ** second_number

# Step 4: Display all results in a clean format
print("\nResults:")
print(f"{int(first_number)} + {int(second_number)} = {addition}")
print(f"{int(first_number)} - {int(second_number)} = {subtraction}")
print(f"{int(first_number)} * {int(second_number)} = {multiplication}")
print(f"{int(first_number)} / {int(second_number)} = {division_result}")
print(f"{int(first_number)} % {int(second_number)} = {modulus}")
print(f"{int(first_number)} ^ {int(second_number)} = {exponentiation}")
```

Q3: String Manipulator

Difficulty: Medium | Marks: 4

```
Q3: String Manipulator
# We take a sentence from the user and show various string properties.

# Step 1: Get a sentence from the user
sentence = input("Enter a sentence: ")

# Step 2: Compute each required property using built-in string methods

# Total characters including spaces
total_chars_with_spaces     = len(sentence)

# Total characters excluding spaces
total_chars_without_spaces = len(sentence.replace(" ", ""))

# Count words by splitting on whitespace
word_list  = sentence.split()
total_words = len(word_list)

# Various case transformations
uppercase_text = sentence.upper()
lowercase_text = sentence.lower()
titlecase_text = sentence.title()

# First and last word from the word list
first_word = word_list[0]  if word_list else ""
last_word  = word_list[-1] if word_list else ""

# Reversed sentence (reverse each character)
reversed_sentence = sentence[::-1]

# Step 3: Display all results
print(f"\nOriginal: {sentence}")
print(f"Characters (with spaces): {total_chars_with_spaces}")
print(f"Characters (without spaces): {total_chars_without_spaces}")
print(f"Words: {total_words}")
print(f"UPPERCASE: {uppercase_text}")
print(f"lowercase: {lowercase_text}")
print(f"Title Case: {titlecase_text}")
print(f"First word: {first_word}")
print(f"Last word: {last_word}")
print(f"Reversed: {reversed_sentence}")
```

Q4: Age Calculator

Difficulty: Medium | Marks: 4

```
Q4:  Age  Calculator
# We ask the user for their birth year, then calculate age-related values.
# Bonus: We also ask for exact birth date for a precise calculation.

from datetime import date   # built-in module for working with dates

# Step 1: Get the current year automatically
current_year = date.today().year
current_date = date.today()

# Step 2: Ask the user for their birth year (with error handling)
try:
    birth_year = int(input("Enter your birth year: "))

    if birth_year > current_year or birth_year < 1900:
        print("Please enter a valid birth year between 1900 and today.")
        exit()
except ValueError:
    print("Invalid input! Please enter a number.")
    exit()

# Step 3: Calculate basic age (using just the year)
current_age = current_year - birth_year

# Step 4: Calculate age in different units
age_in_months  = current_age * 12
age_in_days    = current_age * 365          # approximate (ignores leap years)
age_in_hours   = age_in_days * 24
age_in_minutes = age_in_hours * 60
years_to_100   = 100 - current_age

# Step 5: Display results
print(f"\n=== AGE CALCULATOR ===")
print(f"Current Age    : {current_age} years")
print(f"Age in Months  : {age_in_months} months")
print(f"Age in Days    : {age_in_days} days (approx.)")
print(f"Age in Hours   : {age_in_hours} hours (approx.)")
print(f"Age in Minutes : {age_in_minutes} minutes (approx.)")
print(f"Years to 100   : {years_to_100} years")

# Step 6: BONUS - Ask for exact birth date
print("\n--- BONUS: Exact Calculation ---")
try:
    birth_day   = int(input("Enter birth day (1-31): "))
    birth_month = int(input("Enter birth month (1-12): "))
    birth_year2 = int(input("Enter birth year again: "))
    exact_birth = date(birth_year2, birth_month, birth_day)
    exact_days  = (current_date - exact_birth).days
    exact_age   = current_date.year - exact_birth.year - (
        (current_date.month, current_date.day) <
        (exact_birth.month, exact_birth.day)
    )
    print(f"Exact Age      : {exact_age} years")
    print(f"Exact Days     : {exact_days} days")
except Exception as e:
    print(f"Could not calculate exact age: {e}")
```

Q5: Bill Splitter

Difficulty: Medium | Marks: 4

```
Q5:  Bill  Splitter
# We collect bill details and show a full breakdown with tax and tip.

# Step 1: Collect inputs from the user (with error handling)
try:
    total_bill      = float(input("Enter total bill: "))
    number_of_people= int(input("Number of people: "))
    tax_percentage  = float(input("Tax percentage: "))
    tip_percentage  = float(input("Tip percentage: "))

    # Basic validation
    if number_of_people <= 0:
        print("Number of people must be at least 1.")
        exit()
except ValueError:
    print("Invalid input! Please enter valid numbers.")
    exit()

# Step 2: Calculate each part of the bill
subtotal       = total_bill
tax_amount     = subtotal * (tax_percentage / 100)
bill_after_tax = subtotal + tax_amount
tip_amount     = bill_after_tax * (tip_percentage / 100)
grand_total    = bill_after_tax + tip_amount
amount_per_person = grand_total / number_of_people

# Step 3: Display the full breakdown
print("\n=== BILL BREAKDOWN ===")
print(f"Subtotal: Rs.{subtotal:.2f}")
print(f"Tax ({tax_percentage:.0f}%)         : Rs.{tax_amount:.2f}")
print(f"After tax: Rs.{bill_after_tax:.2f}")
print(f"Tip ({tip_percentage:.0f}%)         : Rs.{tip_amount:.2f}")
print(f"Total: Rs.{grand_total:.2f}")
print(f"Per person: Rs.{amount_per_person:.2f}")
```

Q6: Grade Calculator

Difficulty: Easy-Medium | Marks: 4

```
Q6: Grade Calculator
# We take marks for 5 subjects, calculate total, percentage, grade & result.

subject_names = ["Maths", "Science", "English", "History", "Computer"]
subject_marks = []   # we will fill this list with user-entered marks

# Step 1: Take marks for each subject one by one
print("Enter marks for each subject (out of 100):")
for subject in subject_names:
    while True:   # keep asking until valid input
        try:
            mark = float(input(f"  {subject}: "))
            if 0 <= mark <= 100:
                subject_marks.append(mark)
                break   # valid mark, move to next subject
            else:
                print("  Marks must be between 0 and 100. Try again.")
        except ValueError:
            print("  Invalid input! Please enter a number.")
```

```
# Step 2: Calculate total and percentage
total_marks  = sum(subject_marks)      # sum of all 5 subjects
max_marks    = 500                     # 5 subjects × 100
percentage   = (total_marks / max_marks) * 100

# Step 3: Determine grade based on percentage
if   percentage >= 90: grade = "A+ (Outstanding)"
elif percentage >= 80: grade = "A  (Excellent)"
elif percentage >= 70: grade = "B  (Good)"
elif percentage >= 60: grade = "C  (Average)"
elif percentage >= 50: grade = "D  (Pass)"
else:                  grade = "F  (Fail)"

# Step 4: Check if the student passed (all subjects must be >= 40)
passed_all = all(mark >= 40 for mark in subject_marks)
result_text = "PASS" if passed_all else "FAIL"

# Step 5: Display the complete report
print("\n=== GRADE REPORT ===")
for name, mark in zip(subject_names, subject_marks):
    print(f"  {name:<10}: {mark:.1f}")
print(f"Total: {total_marks:.1f} / {max_marks}")
print(f"Percentage: {percentage:.2f}%")
print(f"Grade: {grade}")
print(f"Result: {result_text}")
```

Q7: Temperature Converter

Difficulty: Medium | Marks: 4

```
Q7: Temperature Converter
# A menu-driven program with conversion functions for C, F, and K.

# --- Conversion functions ---
def celsius_to_fahrenheit(c):
    return (c * 9/5) + 32

def fahrenheit_to_celsius(f):
    return (f - 32) * 5/9

def celsius_to_kelvin(c):
    return c + 273.15

def kelvin_to_celsius(k):
    return k - 273.15

def fahrenheit_to_kelvin(f):
    return (f - 32) * 5/9 + 273.15

def kelvin_to_fahrenheit(k):
    return (k - 273.15) * 9/5 + 32
```

```
# --- Main menu loop ---
def temperature_converter():
    while True:
        # Show the menu each loop iteration
        print("\n=== TEMPERATURE CONVERTER ===")
        print("1. Celsius to Fahrenheit")
        print("2. Fahrenheit to Celsius")
        print("3. Celsius to Kelvin")
        print("4. Kelvin to Celsius")
        print("5. Fahrenheit to Kelvin")
        print("6. Kelvin to Fahrenheit")
        print("7. Exit")

        choice = input("\nEnter your choice (1-7): ").strip()

        if choice == "7":
            print("Goodbye!")
            break   # exit the loop

        # Make sure the choice is valid before asking for temperature
        if choice not in ["1","2","3","4","5","6"]:
            print("Invalid choice! Please enter a number from 1 to 7.")
            continue

        # Get the temperature value (with error handling)
        try:
            temp_value = float(input("Enter temperature value: "))
        except ValueError:
            print("Invalid number! Please try again.")
            continue

        # Perform the selected conversion
        if   choice == "1":
            print(f"Result: {celsius_to_fahrenheit(temp_value):.2f} F")
        elif choice == "2":
            print(f"Result: {fahrenheit_to_celsius(temp_value):.2f} C")
        elif choice == "3":
            print(f"Result: {celsius_to_kelvin(temp_value):.2f} K")
        elif choice == "4":
```

```
            print(f"Result: {kelvin_to_celsius(temp_value):.2f} C")
        elif choice == "5":
            print(f"Result: {fahrenheit_to_kelvin(temp_value):.2f} K")
        elif choice == "6":
            print(f"Result:  {kelvin_to_fahrenheit(temp_value):.2f}  F")

# Call the main function to start the program
temperature_converter()
```

Q8: Leap Year Checker

Difficulty: Medium | Marks: 4

```
Q8: Leap Year Checker
# We check leap year rules and also explain WHY the year is or isn't a leap year.

# Step 1: Get the year from the user
try:
    year = int(input("Enter a year: "))
    if year <= 0:
        print("Please enter a positive year.")
        exit()
except ValueError:
    print("Invalid input! Please enter a whole number.")
    exit()

# Step 2: Apply the leap year rules
# Rule: divisible by 4 AND (NOT divisible by 100 OR divisible by 400)
if year % 4 == 0:
    if year % 100 == 0:
        if year % 400 == 0:
            is_leap = True
            reason  = f"{year} is divisible by 400, so it IS a leap year."
        else:
            is_leap = False
            reason  = f"{year} is divisible by 100 but NOT 400, so it is NOT a leap year."
    else:
        is_leap = True
        reason  = f"{year} is divisible by 4 but not 100, so it IS a leap year."
else:
    is_leap = False
    reason  = f"{year} is not divisible by 4, so it is NOT a leap year."

# Step 3: Display the result
print(f"\nYear   : {year}")
print(f"Result : {'Leap Year' if is_leap else 'NOT a Leap Year'}")
print(f"Reason : {reason}")
```

Q9: Ticket Pricing System

Difficulty: Medium | Marks: 4

```
Q9: Ticket Pricing System
# We determine the base ticket price by age, apply a weekend discount, then
# multiply by the number of tickets to get the final total.

# Step 1: Get all the required inputs from the user
try:
    customer_age    = int(input("Enter age: "))
    day_of_week     = input("Enter day (e.g. Monday): ").strip().capitalize()
    number_of_tickets = int(input("Number of tickets: "))

    if number_of_tickets <= 0:
        print("Number of tickets must be at least 1.")
        exit()
except ValueError:
    print("Invalid input! Please enter valid values.")
    exit()

# Step 2: Determine base price based on age category
if customer_age < 3:
    base_price    = 0
    category_name = "Child (Below 3 - Free)"
elif customer_age <= 12:
    base_price    = 150
    category_name = "Child (3-12)"
elif customer_age <= 59:
    base_price    = 300
    category_name = "Adult (13-59)"
else:
    base_price    = 200
    category_name = "Senior (60+)"

# Step 3: Check if a weekend discount applies (Friday, Saturday, Sunday)
weekend_days    = ["Friday", "Saturday", "Sunday"]
discount_rate   = 0.20 if day_of_week in weekend_days else 0.0
discount_amount = base_price * discount_rate
price_after_discount = base_price - discount_amount

# Step 4: Calculate total amount for all tickets
total_amount = price_after_discount * number_of_tickets

# Step 5: Display the breakdown
print("\n=== TICKET PRICING ===")
print(f"Category     :     {category_name}")
print(f"Base Price       : Rs.{base_price}")
if discount_rate > 0:
    print(f"Discount  (20%)  : -Rs.{discount_amount:.2f}")
else:
    print(f"Discount         : None")
print(f"Price per Ticket : Rs.{price_after_discount:.2f}")
print(f"Tickets          : {number_of_tickets}")
print(f"Total Amount     : Rs.{total_amount:.2f}")
```

Q10: Simple ATM Simulator

Difficulty: Medium-Hard | Marks: 7

```
Q10: Simple ATM Simulator
# We simulate an ATM with a balance, allowing deposit and withdrawal,
# while enforcing a minimum balance rule of Rs.500.
```

```
# Step 1: Set the initial account balance
account_balance  = 10000.0    # starting balance in Rupees
MINIMUM_BALANCE  = 500.0      # balance must never go below this

def show_balance():
    """Display the current account balance."""
    print(f"  Current Balance: Rs.{account_balance:.2f}")

def deposit_money():
    """Add money to the account after validating the amount."""
    global account_balance
    try:
        deposit_amount = float(input("  Enter amount to deposit: Rs."))
        if deposit_amount <= 0:
            print("  Deposit amount must be greater than zero.")
            return
        account_balance += deposit_amount
        print(f"  Rs.{deposit_amount:.2f} deposited successfully!")
        print(f"  New Balance: Rs.{account_balance:.2f}")
    except ValueError:
        print("  Invalid amount! Please enter a number.")

def withdraw_money():
    """Withdraw money if there is enough balance (keeping minimum Rs.500)."""
    global account_balance
    try:
        withdraw_amount = float(input("  Enter amount to withdraw: Rs."))
        if withdraw_amount <= 0:
            print("  Withdrawal amount must be greater than zero.")
            return
        # Check if withdrawal would bring balance below minimum
        if account_balance - withdraw_amount < MINIMUM_BALANCE:
            print(f"  Cannot withdraw! Minimum balance of Rs.{MINIMUM_BALANCE} must remain.")
            print(f"  Maximum you can withdraw: Rs.{account_balance - MINIMUM_BALANCE:.2f}")
            return
        account_balance -= withdraw_amount
        print(f"  Withdrawal successful!")
        print(f"  New Balance: Rs.{account_balance:.2f}")
    except ValueError:
        print("  Invalid amount! Please enter a number.")

# Step 2: Main ATM loop – keep showing menu until user chooses Exit
def atm_simulator():
    print("\n====== ATM SIMULATOR ======")
    while True:
        print("\n  1. Check Balance")
        print("  2. Deposit Money")
        print("  3. Withdraw Money")
        print("  4. Exit")
        menu_choice = input("\n  Enter choice: ").strip()

        if   menu_choice == "1": show_balance()
        elif menu_choice == "2": deposit_money()
        elif menu_choice == "3": withdraw_money()
        elif menu_choice == "4":
            print("  Thank you for using our ATM. Goodbye!")
            break
        else:
            print("  Invalid choice! Please enter 1, 2, 3, or 4.")

# Start the ATM
atm_simulator()
```

Q11: Number Pattern Printer

Difficulty: Medium | Marks: 4

```
Q11: Number Pattern Printer
# We offer 4 pattern choices and let the user set the height (rows).

def pattern_1(height):
    """Print increasing numbers row by row: 1 / 1 2 / 1 2 3 ..."""
    for row in range(1, height + 1):
        # On each row, print numbers 1 up to the current row number
        numbers = " ".join(str(num) for num in range(1, row + 1))
        print(numbers)

def pattern_2(height):
    """Print repeated row number: 1 / 2 2 / 3 3 3 ..."""
    for row in range(1, height + 1):
        # Repeat the row number 'row' times
        numbers = " ".join(str(row) for _ in range(row))
        print(numbers)

def pattern_3(height):
    """Countdown rows: 5 4 3 2 1 / 4 3 2 1 / ..."""
    for row in range(height, 0, -1):
        # Count down from current row number to 1
        numbers = " ".join(str(num) for num in range(row, 0, -1))
        print(numbers)

def pattern_4(height):
    """Pyramid pattern: 1 / 121 / 12321 ..."""
    for row in range(1, height + 1):
        # Build ascending part then descending (without repeating middle)
        ascending  = "".join(str(n) for n in range(1, row + 1))
        descending = "".join(str(n) for n in range(row - 1, 0, -1))
        print(ascending + descending)

# --- Main program ---
print("=== NUMBER PATTERN PRINTER ===")
print("1. Increasing Numbers")
print("2. Repeated Row Number")
print("3. Countdown Rows")
print("4. Pyramid Pattern")

try:
    pattern_choice = int(input("\nChoose a pattern (1-4): "))
    row_height     = int(input("Enter height (number of rows): "))

    if pattern_choice not in [1, 2, 3, 4] or row_height <= 0:
        print("Invalid choice or height!")
        exit()
except ValueError:
    print("Please enter valid numbers!")
    exit()

print()  # blank line before the pattern
if  pattern_choice == 1: pattern_1(row_height)
elif pattern_choice == 2: pattern_2(row_height)
elif pattern_choice == 3: pattern_3(row_height)
elif pattern_choice == 4: pattern_4(row_height)
```

Q12: Multiplication Table Generator

Difficulty: Easy-Medium | Marks: 4

```
Q12: Multiplication Table Generator
# We print a multiplication table for a user-chosen number up to a given range.
# Bonus: print a full 10x10 grid.

# Step 1: Get the number and the end of the range from the user
try:
    base_number = int(input("Enter number: "))
    range_end   = int(input("Enter range (end): "))
    if range_end <= 0:
        print("Range must be a positive number.")
        exit()
except ValueError:
    print("Invalid input! Please enter whole numbers.")
    exit()

# Step 2: Print the multiplication table for the given number
print(f"\nMultiplication Table of {base_number}")
print("-" * 20)
for multiplier in range(1, range_end + 1):
    result = base_number * multiplier
    print(f"  {base_number} x {multiplier:2d} = {result}")

# Step 3: BONUS - Full 10x10 multiplication table grid
print("\n--- BONUS: Full 10x10 Multiplication Table ---")
# Print the header row (1 through 10)
header = "    " + "  ".join(f"{col:3d}" for col in range(1, 11))
print(header)
print("    " + "-" * 40)

for row_num in range(1, 11):
    # Build each row: row_num × 1, row_num × 2, ... row_num × 10
    row_values = "  ".join(f"{row_num * col:3d}" for col in range(1, 11))
    print(f"{row_num:2d} | {row_values}")
```

Q13: Sum and Average Calculator

Difficulty: Easy | Marks: 4

```
Q13: Sum and Average Calculator
# We collect a list of numbers from the user and calculate key statistics.

# Step 1: Ask how many numbers the user wants to enter
try:
    count = int(input("How many numbers? "))
    if count <= 0:
        print("Please enter a positive count.")
        exit()
except ValueError:
    print("Invalid input! Enter a whole number.")
    exit()

# Step 2: Collect each number using a loop
numbers_list = []
for i in range(1, count + 1):
    while True:   # keep asking until a valid number is given
        try:
            num = float(input(f"Enter number {i}: "))
            numbers_list.append(num)
            break
        except ValueError:
            print("  Invalid! Please enter a number.")

# Step 3: Calculate the required statistics
total_sum   = sum(numbers_list)
average    =  total_sum  /  count
maximum_num  = max(numbers_list)
minimum_num  = min(numbers_list)

# Step 4: Display the results
print("\n=== RESULTS ===")
print(f"Sum     : {total_sum}")
print(f"Average : {average:.2f}")
print(f"Maximum : {maximum_num}")
print(f"Minimum : {minimum_num}")
```

Q14: Factorial Calculator

Difficulty: Medium | Marks: 4

```
Q14: Factorial Calculator
# We calculate n! using a loop and show each step of the multiplication.

# Step 1: Get the number from the user
try:
    number = int(input("Enter a number: "))
except ValueError:
    print("Invalid input! Please enter a whole number.")
    exit()

# Step 2: Handle special cases before the loop
if number < 0:
    print("Factorial is not defined for negative numbers.")
    exit()
elif number == 0:
    print("0! = 1  (by definition, the factorial of 0 is 1)")
    exit()
```

```
# Step 3: Calculate factorial step by step using a loop
factorial_result = 1
steps = []   # we collect each factor to display the calculation

for factor in range(number, 0, -1):
    factorial_result *= factor
    steps.append(str(factor))

# Step 4: Build and display the step-by-step output
steps_string = " x ".join(steps)   # e.g. "5 x 4 x 3 x 2 x 1"
print(f"\n{number}! = {steps_string} = {factorial_result}")
```

Q15: Prime Number Checker

Difficulty: Medium-Hard | Marks: 7

```
Q15: Prime Number Checker
# Part 1: Check if a single number is prime.
# Part 2: Find all primes in a given range.

import math   # we use math.sqrt for an efficient primality check

def is_prime(n):
    """Return True if n is a prime number, False otherwise."""
    # Handle special cases first
    if n < 2:
        return False   # 0, 1, and negatives are NOT prime
    if n == 2:
        return True    # 2 is the only even prime
    if n % 2 == 0:
        return False   # all other even numbers are NOT prime

    # Check divisibility from 3 up to the square root of n
    # We only need to check up to sqrt(n) because if n has a factor
    # larger than sqrt(n), it must also have one smaller than sqrt(n)
    for divisor in range(3, int(math.sqrt(n)) + 1, 2):
        if n % divisor == 0:
            return False   # found a factor, so n is NOT prime

    return True   # no factors found - n IS prime

# --- Part 1: Check a single number ---
try:
    single_number = int(input("Enter a number to check: "))
except ValueError:
    print("Invalid input!")
    exit()

if is_prime(single_number):
    print(f"{single_number} is a PRIME number")
else:
    print(f"{single_number} is NOT a prime number")

# --- Part 2: Find all primes in a range ---
try:
    start_range = int(input("\nEnter start of range: "))
    end_range   = int(input("Enter end of range: "))
    if start_range > end_range:
        print("Start must be less than or equal to end.")
        exit()
except ValueError:
    print("Invalid range input!")
    exit()

# Collect all prime numbers in the given range
primes_in_range = [n for n in range(start_range, end_range + 1) if is_prime(n)]

if primes_in_range:
    print(f"\nPrime numbers between {start_range} and {end_range}:")
    print(", ".join(str(p) for p in primes_in_range))
else:
    print(f"No prime numbers found between {start_range} and {end_range}.")
```

Q16: Number Guessing Game

Difficulty: Hard | Marks: 7

```
Q16: Number Guessing Game
# The computer picks a random number 1-100; user has 7 attempts to guess it.
# Bonus: tracks the best score and gives hints when the guess is close.

import random   # built-in module to generate random numbers

MAX_ATTEMPTS = 7       # maximum guesses allowed per game
best_score   = None    # tracks the fewest guesses used to win (None = no win yet)

def play_game():
    """Run one round of the guessing game. Returns number of attempts if won."""
    global best_score

    # Step 1: Computer secretly picks a random number between 1 and 100
    secret_number = random.randint(1, 100)
    attempts_used = 0

    print("\n=== NUMBER GUESSING GAME ===")
    print(f"I've picked a number between 1 and 100. You have {MAX_ATTEMPTS} attempts.")

    # Step 2: Let the user guess up to MAX_ATTEMPTS times
    while attempts_used < MAX_ATTEMPTS:
        attempts_left = MAX_ATTEMPTS - attempts_used
        print(f"\nAttempts remaining: {attempts_left}")

        try:
            user_guess = int(input("Your guess: "))
        except ValueError:
            print("Please enter a whole number!")
            continue   # don't count invalid input as an attempt

        attempts_used += 1   # count this as one attempt

        # Step 3: Check the guess and give feedback
        if user_guess == secret_number:
            print(f"Correct! You guessed it in {attempts_used} attempt(s). Well done!")
            # Update best score if this is better (fewer attempts)
            if best_score is None or attempts_used < best_score:
                best_score = attempts_used
                print(f"New best score: {best_score} attempt(s)!")
            else:
                print(f"Best score: {best_score} attempt(s).")
            return   # game over - user won

        elif user_guess < secret_number:
            print("Too LOW!")
        else:
            print("Too HIGH!")

        # BONUS: Give a hint when the guess is within 5 of the secret number
        if abs(user_guess - secret_number) <= 5:
            print("  Hint: You are VERY CLOSE!")

    # Step 4: All attempts used - reveal the answer
    print(f"\nGame over! You used all {MAX_ATTEMPTS} attempts.")
    print(f"The secret number was: {secret_number}")

# --- Main loop: allow the user to play multiple times ---
while True:
    play_game()
    play_again = input("\nPlay again? (yes/no): ").strip().lower()
    if play_again not in ["yes", "y"]:
        print("Thanks for playing! Goodbye!")
```

```
        break
```

Q17: Palindrome Checker

Difficulty: Medium | Marks: 4

```
Q17: Palindrome Checker
# A palindrome reads the same forwards and backwards (e.g. "racecar", 121).
# We ignore case for words.

# Step 1: Get input from the user
user_input = input("Enter a word or number: ").strip()

# Step 2: Clean the input for comparison
# Convert to lowercase so "Racecar" and "racecar" are treated the same
cleaned_input    = user_input.lower().replace(" ", "")

# Step 3: Reverse the cleaned input
reversed_input  = cleaned_input[::-1]

# Step 4: Show the step-by-step verification
print(f"\nOriginal  : {user_input}")
print(f"Cleaned    : {cleaned_input}")
print(f"Reversed  : {reversed_input}")

# Step 5: Compare and print the result
if cleaned_input == reversed_input:
    print("Result     : PALINDROME ✓")
else:
    print("Result     : NOT a palindrome ✗")
```

Q18: Calculator Functions

Difficulty: Medium | Marks: 7

```
Q18: Calculator Functions
# We define individual functions for each operation and a menu function
# that ties them all together.

# --- Individual operation functions ---

def add(a, b):
    """Return the sum of a and b."""
    return a + b

def subtract(a, b):
    """Return a minus b."""
    return a - b

def multiply(a, b):
    """Return the product of a and b."""
    return a * b

def divide(a, b):
    """Return a divided by b. Returns an error message if b is zero."""
    if b == 0:
        return "Error: Cannot divide by zero!"
    return a / b

def modulus(a, b):
    """Return the remainder of a divided by b."""
    if b == 0:
        return "Error: Cannot take modulus with zero!"
    return a % b
```

```
def power(a, b):
    """Return a raised to the power of b."""
    return a ** b

# --- Main calculator function with a menu ---

def calculator():
    """Display a menu, take input, call the right function, show the result."""
    while True:
        print("\n=== CALCULATOR ===")
        print("1.Addition(a+b)")
        print("2.Subtraction(a-b)")
        print("3. Multiplication (a * b)")
        print("4. Division(a / b)")
        print("5. Modulus(a % b)")
        print("6. Power(a ^ b)")
        print("7. Exit")

        choice = input("\nEnter choice (1-7): ").strip()

        if choice == "7":
            print("Goodbye!")
            break   # exit the loop

        if choice not in ["1","2","3","4","5","6"]:
            print("Invalid choice! Enter a number from 1 to 7.")
            continue

        # Get the two numbers for the operation
        try:
            num_a = float(input("Enter first number (a): "))
            num_b = float(input("Enter second number (b): "))
        except ValueError:
            print("Invalid number! Please enter numeric values.")
            continue

        # Call the appropriate function based on the user's choice
        if   choice == "1": result = add(num_a, num_b)
        elif choice == "2": result = subtract(num_a, num_b)
        elif choice == "3": result = multiply(num_a, num_b)
        elif choice == "4": result = divide(num_a, num_b)
        elif choice == "5": result = modulus(num_a, num_b)
        elif choice == "6": result = power(num_a, num_b)

        print(f"Result: {result}")

# Start the calculator
calculator()
```

Q19: Text Analysis Functions

Difficulty: Hard | Marks: 9

```
Q19: Text Analysis Functions
# A collection of text-analysis functions that can each be called individually.
# The analyze_text() master function calls all of them and shows a report.

def count_words(text):
    """Return the number of words in the text."""
    return len(text.split())

def count_vowels(text):
    """Return the count of vowel letters (a e i o u) in the text."""
    vowels = "aeiouAEIOU"
    return sum(1 for char in text if char in vowels)

def count_consonants(text):
    """Return the count of consonant letters in the text."""
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"
    return sum(1 for char in text if char in consonants)

def reverse_text(text):
    """Return the text with all characters reversed."""
    return text[::-1]

def is_palindrome(text):
    """Return True if the text (cleaned) reads same forwards and backwards."""
    cleaned = text.lower().replace(" ", "")
    return cleaned == cleaned[::-1]

def remove_vowels(text):
    """Return the text with all vowels removed."""
    vowels = "aeiouAEIOU"
    return "".join(char for char in text if char not in vowels)

def word_frequency(text):
    """Return a dictionary mapping each word to how many times it appears."""
    frequency = {}
    for word in text.lower().split():
        # If the word is already in the dict, increment its count, else set to 1
        frequency[word] = frequency.get(word, 0) + 1
    return frequency

def longest_word(text):
    """Return the longest word in the text."""
    words = text.split()
    if not words:
        return ""
    # Find the word with the maximum length
    return max(words, key=len)

def analyze_text(text):
    """Call all analysis functions and display a full report."""
    freq_dict    = word_frequency(text)
    freq_display = ", ".join(f"{w}: {c}" for w, c in freq_dict.items())
    top_word     = longest_word(text)

    print("\n=== TEXT ANALYSIS ===")
    print(f"Words           : {count_words(text)}")
    print(f"Vowels          : {count_vowels(text)}")
    print(f"Consonants      : {count_consonants(text)}")
    print(f"Reversed        : {reverse_text(text)}")
    print(f"Palindrome      : {'Yes' if is_palindrome(text) else 'No'}")
    print(f"Without vowels : {remove_vowels(text)}")
```

```
    print(f"Longest word   : {top_word} ({len(top_word)} letters)")
    print(f"Word Frequency : {freq_display}")

# --- Run the program ---
user_text = input("Enter text: ")
if user_text.strip():
    analyze_text(user_text)
else:
    print("No text entered!")
```

Q20: Number System Functions

Difficulty: Hard | Marks: 9

```
Q20: Number System Functions
# A collection of mathematical functions with a menu to test each one.

def factorial(n):
    """Return n! (factorial). Returns 1 for n=0, and None for negatives."""
    if n < 0:
        return None   # factorial not defined for negative numbers
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

def is_prime(n):
    """Return True if n is a prime number."""
    if n < 2: return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def fibonacci(n):
    """Return the nth Fibonacci number (0-indexed: fib(0)=0, fib(1)=1)."""
    if n < 0: return None
    if n == 0: return 0
    if n == 1: return 1
    a, b = 0, 1
    for _ in range(2, n + 1):
        a, b = b, a + b   # shift: a becomes old b, b becomes a+b
    return b

def sum_of_digits(n):
    """Return the sum of all digits in n."""
    return sum(int(d) for d in str(abs(n)))

def reverse_number(n):
    """Return n with its digits reversed."""
    reversed_str = str(abs(n))[::-1]
    return int(reversed_str) * (-1 if n < 0 else 1)

def is_armstrong(n):
    """Return True if n is an Armstrong number (sum of digits^len == n)."""
    digits = str(abs(n))
    power  = len(digits)
    total  = sum(int(d) ** power for d in digits)
    return total == abs(n)

def gcd(a, b):
    """Return the Greatest Common Divisor of a and b using Euclid's algorithm."""
```

```
    a, b = abs(a), abs(b)
    while b:
        a, b = b, a % b   # remainder replaces b; old b becomes new a
    return a

def lcm(a, b):
    """Return the Least Common Multiple of a and b."""
    if a == 0 or b == 0: return 0
    return abs(a * b) // gcd(a, b)

def is_perfect_number(n):
    """Return True if the sum of proper divisors of n equals n (e.g., 6)."""
    if n <= 1: return False
    divisors_sum = sum(i for i in range(1, n) if n % i == 0)
    return divisors_sum == n

# --- Menu to test all functions ---
def math_menu():
    menu_options = {
        "1": "Factorial",
        "2": "Is Prime?",
        "3": "Fibonacci(n)",
        "4": "Sum of Digits",
        "5": "Reverse Number",
        "6": "Is Armstrong?",
        "7": "GCD of two numbers",
        "8": "LCM of two numbers",
        "9": "Is Perfect Number?",
        "10": "Exit"
    }

    while True:
        print("\n=== MATH FUNCTIONS MENU ===")
        for key, val in menu_options.items():
            print(f" {key}. {val}")

        choice = input("\nEnter choice: ").strip()

        if choice == "10":
            print("Goodbye!")
            break

        try:
            if choice == "1":
                n = int(input("Enter n for factorial: "))
                print(f" {n}! = {factorial(n)}")
            elif choice == "2":
                n = int(input("Enter a number: "))
                print(f" {n} is {'PRIME' if is_prime(n) else 'NOT prime'}")
            elif choice == "3":
                n = int(input("Enter position n (0-based): "))
                print(f" Fibonacci({n}) = {fibonacci(n)}")
            elif choice == "4":
                n = int(input("Enter a number: "))
                print(f"  Sum of digits of {n} = {sum_of_digits(n)}")
            elif choice == "5":
                n = int(input("Enter a number: "))
                print(f" Reversed: {reverse_number(n)}")
            elif choice == "6":
                n = int(input("Enter a number: "))
                print(f"  {n} is {'an Armstrong' if is_armstrong(n) else 'NOT an Armstrong'} num
ber")
            elif choice == "7":
                a = int(input("Enter first number: "))
```

```
                b = int(input("Enter second number: "))
                print(f"  GCD({a}, {b}) = {gcd(a, b)}")
            elif choice == "8":
                a = int(input("Enter first number: "))
                b = int(input("Enter second number: "))
                print(f"  LCM({a}, {b}) = {lcm(a, b)}")
            elif choice == "9":
                n = int(input("Enter a number: "))
                print(f"  {n} is {'a PERFECT' if is_perfect_number(n) else 'NOT a perfect'} numb
er")
            else:
                print("  Invalid choice! Enter a number from 1 to 10.")
        except ValueError:
            print("  Invalid input! Please enter a whole number.")

# Start the menu
math_menu()
```