# COMP 141 — Midterm 1 Practice Problems

Fall 2026

## Topics to Review

The practice problems cover the main topics that may appear on Midterm 1. You should be comfortable both **reading Python code** and **writing short programs yourself**.

### Variables, Data Types, Input, and Output

Review:

- variables and assignment;
- integers, floats, strings, and Boolean values;
- converting values with `int()`, `float()`, and `str()`;
- using `input()`;
- using `print()`;
- the `end` argument to `print()`;
- formatted output using `.format()`;
- comments.

Be able to predict the type and value of variables after statements execute.

### Arithmetic

Review:

- `+`, `-`, `*`, and `/`;
- the remainder operator `%`;
- the exponent operator `**`;
- parentheses and order of operations;
- augmented assignment such as `+=`, `-=`, and `*=`.

### Boolean Expressions and Conditionals

Review:

- comparison operators: `<`, `>`, `<=`, `>=`, `==`, and `!=`;
- `and`, `or`, and `not`;
- evaluating Boolean expressions;
- `if`, `elif`, and `else`;
- tracing which branches of a conditional execute.

### Functions

Review:

- defining and calling functions;
- function headers;
- parameters and arguments;
- local and global variables;
- `return` versus `print`;
- tracing function calls and return values.

### Loops

Review:

- `while` loops;
- changing loop variables;
- accumulators such as `total`;
- using conditionals inside loops;
- tracing loops together with functions.

### Writing Programs

Be prepared to write short pieces of Python that:

- use conditionals;
- use loops;
- define and call functions;

- take user input;
- return values from functions;
- accumulate totals;
- determine whether numbers are even or odd;
- generate random numbers.

**Advice:** Do not only reread notes. Practice tracing code by hand and writing small programs without using a computer.

## Practice Problems

**1.** The _____ function reads data entered at the keyboard and returns that data as a string.

a. `input`

b. `output`

c. `eval_input`

d. `string_input`

**2.** In a print statement, you can set the _____ argument to a space or empty string to prevent the output from advancing to a new line.

a. `stop`

b. `end`

c. `separator`

d. `newline`

**3.** What type of data is stored in `sold`?

```
sold = 256.752
```

a. int

b. float

c. str

d. bool

**4.** What type of data is stored in `a`?

```
a = input("Enter a number: ")
```

a. int

b. float

c. str

d. bool

**5.** What value does `price` reference?

```
price = int(68.549)
```

a. 68

b. 69

c. 68.55

d. 68.54

**6.** Programmers should generally avoid using _____ variables when possible.

a. local

b. global

c. string global

d. keyword

Questions 8–10 refer to the following code. Line numbers are included only for reference.

```
1  def average(first, second, third):
2      avg = (first + second + third) / 3
3      print("average is", avg)
4
5  def main():
6      x = float(input("First number? "))
7      y = float(input("Second number? "))
8      z = float(input("Third number? "))
9      average(x, y, z)
10
11 main()
```

**8.** In Lines 2 and 3, `avg` is a _____ variable to the function `average`.

a. global

b. constant

c. defined

d. local

**9.** In Line 1, `first`, `second`, and `third` are _____.

a. headers

b. returns

c. parameters

d. arguments

**10.** In Line 9, `x`, `y`, and `z` are _____ used when calling `average`.

a. headers

b. returns

c. parameters

d. arguments

**11.** What is the result if

```
x = 5
y = 3
z = 8
```

and the expression is:

```
x < y or z > x
```

a. True

b. False

c. 8

d. 5

**12.** Using the same values, what is the result?

```
not (x < y or z > x) and y < z
```

a. True

b. False

c. 8

d. 5

**13.** What does this expression produce?

```
print(str(8) + str(9))
```

Answer: _____

**14.** What is the result?

```
11.3 + 6.6
```

Answer: _____

**15.** What is the output?

```
number = 76.15854
print("The number I'm printing is {:.2f}".format(number))
```

Answer: _____

**16.** What is the result?

```
29 % 6
```

Answer: _____

**17.** What is the result?

```
3 ** 4
```

Answer: _____

**18.** A(n) _____ refers to a sequence of well-defined steps to solve a problem.

**19.** A(n) _____ statement will execute one block of statements if its condition is true, or another block if its condition is false.

**20.** A _____ loop causes a statement or set of statements to repeat as long as a condition is true.

**21.** _____ are notes of explanation that document lines or sections of a program.

**22.** What is the value of `x` after these statements?

```
x = 2
x *= x + 3
```

Answer: _____

**23.** What is printed?

```
y = 0

for i in range(2, 9):
    y += i

print(y)
```

Answer: _____

**24.** What is displayed after this loop terminates?

```
number = 25
isPrime = True
i = 2

while i < number and isPrime:
    if number % i == 0:
        isPrime = False
    i += 1

print("i is", i, "isPrime is", isPrime)
```

Answer:

**25.** What does this code display?

```
age = 19

if age < 18:
    print("Minor")
elif age >= 18 and age < 65:
    print("Adult")
else:
    print("Senior Citizen")
```

Answer: _____

### 26. Function Tracing

What is the complete output?

```
def adjust(x):
    if x < 4:
        return x + 3
    else:
        return x - 2

def combine(a, b):
    first = adjust(a)
    second = adjust(b)
    return first + second

def main():
    x = 2
    y = 5

    print(x, y)
    print(adjust(x))

    result = combine(x, y)
    print(result)

    x = x + 3
    y = y - 2

    print(x, y)
    print(combine(x, y))

main()
```

**Output:**

### 27. Function and While Loop Tracing

What is the complete output?

```
def change(number):
    if number % 2 == 0:
        return number * 2
    else:
        return number + 1
```

```
x = 2

while x < 7:
    print(x, change(x))
    x += 2
```

**Output:**

### 28. Random Number and Conditional

Python can generate a random integer using the `random` module.

First import the module:

```
import random
```

Then:

```
random.randint(a, b)
```

returns a random integer from `a` through `b`, including both endpoints.

For example:

```
number = random.randint(1, 6)
```

stores a random integer from 1 through 6 in `number`.

Write code that:

- generates a random integer from 0 through 100;
- stores it in a variable;
- prints `"Too high"` if the number is greater than 50;
- otherwise prints `"Too low"`.

**29.** Write a function named `calculate_total_bill` that takes two parameters:

```
bill_amt
perc_tip
```

For example, a 15% tip is passed as `0.15`.

The function should **return**, not print, the bill with the tip added.

```
def calculate_total_bill(bill_amt, perc_tip):
```

**30.** Write a function named `compareNumbers` that takes two parameters, `num1` and `num2`, and prints the two numbers in ascending order.

```
def compareNumbers(num1, num2):
```

**31.** Assume `n` is a positive integer.

Write a loop that calculates

$$1^2 + 2^2 + 3^2 + \cdots + n^2$$

and stores the result in a variable named `total`.

For example, if:

```
n = 4
```

then `total` should become:

```
1*1 + 2*2 + 3*3 + 4*4
```

**32.** Write a loop that repeatedly asks the user to enter positive integers.

The user enters a negative number to stop.

For each positive number entered, print whether it is even or odd.

Example:

```
Enter a number: 6
Even
Enter a number: 9
Odd
Enter a number: 3
Odd
Enter a number: -1
```