COMP 141: Midterm 2 Practice Problems

Fall 2026

Solutions are at the end of this document. Try each problem before looking.

Topics to Review

The practice problems cover the main topics that may appear on Midterm 2. You should be comfortable both **reading Python code** and **writing short programs yourself**.

Loops

Review:

- `while` loops and `for` loops, including `range`;
- nested loops;
- running totals;
- input validation;
- a sliding window: compare each item with the one just before it;
- changing the loop variable inside the loop, and tracing the result by hand.

Files

Review:

- `open(filename, "r")` and `open(filename, "w")`;
- what happens when the file does or does not already exist;
- reading a file one line at a time;
- `split` on a line, then `int` or `float` on the pieces;
- closing the file when you are done.

Strings

Review:

- indexes, starting at 0, and `len`;
- slices, including omitted endpoints and negative indexes;
- strings are immutable: concatenation builds a new string;
- `find`, `startswith`, `endswith`, `replace`, and `lower`;
- `isdigit`, and walking a string one character at a time;
- counting and filtering characters.

Functions

Review:

- parameters and arguments;
- local variables;
- `return` versus `print`;
- tracing a call does not change the caller's variables unless the caller assigns the return value.

Practice Problems

1. Do this on paper, not on the computer. What does this code print?

```
x = 0
while x < 10:
    x += 1
    if x % 2 == 0:
        x += 2
    print(x)
```

2. Do this on paper, not on the computer. What does this code print?

```
for i in range(0, 3):
    for j in range(i + 1, 6):
        print(i, j)
```

3. Write a loop that prints this sequence:

```
1, 2, 4, 7, 11, 16, 22, 29, 37, 46, ...
```

Stop when the number would be 100 or more. Do not print any number greater than or equal to 100. One number per line is fine. Figure out the pattern between the numbers. A `while` loop is a natural fit. A `for` loop can also work.

4. Write a function called `count_evens` that counts the even digits in a string.

Example: `count_evens("abc425a")` returns 2.

5. Write a function called `common_factors` that takes two arguments. It counts how many integers greater than or equal to 1 divide both arguments evenly.

Example: `common_factors(12, 18)` returns 4, because both numbers have factors 1, 2, 3, and 6.

6. Write a function called `test_alphabetical` that takes a string and returns whether the characters are in alphabetical order. Each character must come later in the alphabet than the one immediately before it. The function will only be tested with lowercase letters.

- `test_alphabetical("abcde")` returns `True`
- `test_alphabetical("cba")` returns `False`
- `test_alphabetical("ace")` returns `True`
- `test_alphabetical("abcdd")` returns `False`
- `test_alphabetical("ghost")` returns `True`
- `test_alphabetical("bcda")` returns `False`
- `test_alphabetical("almost")` returns `True`

7. Alice and Bob are running in an election. A file called `results.txt` stores the results. Each line has the hour in 24-hour form, the minute, and the name `Alice` or `Bob`. The file is sorted by time. Write a program that answers each part below. Solve them in order.

a. Print who won the election, Alice or Bob.

b. Print every time when two people voted at the exact same time: two consecutive lines with the same hour and minute.

c. Print the largest gap, in minutes, between two consecutive votes.

d. Print how many times Alice received two or more votes in a row, and how many times Bob did. One run of several identical votes counts once.

8. Write a function called `find_biggest_letter` that takes one string and returns the alphabetically latest letter. The string contains only lowercase letters.

- `find_biggest_letter("apple")` returns `"p"`
- `find_biggest_letter("correct")` returns `"t"`

9. Everyone using a program must be at least 18. Write a piece of code that asks for an age and keeps asking until the user types an age of at least 18.

10. Write a function called `roman` that takes one integer and returns a simplified Roman numeral. Ignore the usual subtraction rule. Normally 9 is `IX`, which is 10 minus 1. Your function may return `VIIII` instead.

The symbols are `M` = 1000, `D` = 500, `C` = 100, `L` = 50, `X` = 10, `V` = 5, and `I` = 1.

Start with an empty string. Loop until the number is 0. Inside the loop, if the number is at least one of those values, concatenate that symbol and subtract the value.

- `roman(8)` returns `"VIII"`
- `roman(49)` returns `"XXXXVIIII"`
- `roman(1999)` returns `"MDCCCCLXXXXVIIII"`

Challenge: make a version that uses the real subtraction rules, so 9 is `IX`, not `VIIII`.

11. Write a guess-the-number program. Use `random.randint` to pick an integer from 1 through 100. Loop until the user guesses it. After each wrong guess, print whether it was too low or too high.

12. Which function converts a numeric value to a string?

a. `str`

b. `value`

c. `num`

d. `chr`

13. Which of the following are true? More than one may be true.

a. Opening a file for reading raises an error if the file does not exist.

b. Opening a file for writing raises an error if the file does not exist.

c. Opening a file for reading creates an empty file if the file does not exist.

d. Opening a file for writing creates a new file if the file does not exist.

e. Opening a file for writing overwrites the file if it already exists.

14. Which method can tell you whether a substring occurs anywhere in a string?

a. `endswith`

b. `find`

c. `replace`

d. `startswith`

15. What is `string1` after this code?

```
string1 = "Hello"
string1 += " world"
```

16. What is printed?

```
y = 0
for i in range(1, 10):
    y += i
print(y)
```

17. What is printed?

```
y = 0
for i in range(2, 10, 2):
    y += i
print(y)
```

18. Each character in a string has a value that specifies its position. What is that value called?

19. Once a string is created, its characters cannot be changed. What is the word for that property?

20. What is the name for a span of characters taken from inside a string, such as `s[2:5]`?

21. Write the call that opens `scores.txt` for writing.

22. Write the call that opens `scores.txt` for reading.

23. A file object is named `infile`. Write the call that reads the next line.

24. Let `s = "Programming is fun"`. Give each value.

a. `s[:2]`

b. `s[4:6]`

c. `len(s)`

d. `s.find("ram")`

e. `s.startswith("m")`

f. `s.replace("fun", "awesome")`

g. `s.lower()`

25. What does `nPrint("a", 4)` display?

```
def nPrint(message, n):
    while n > 0:
        print(message)
        n -= 1
```

26. What value of `k` is printed in `main`?

```
def nPrint(message, n):
    while n > 0:
        print(message)
        n -= 1

def main():
    k = 2
```

```
    nPrint("A message", k)
    print(k)

main()
```

27. What does this program display?

```
def f1(x):
    y = x + 2
    print(y)

def main():
    x = 1
    f1(x)
    print(x)

main()
```

28. Write a function called `productDigits` that takes a string of letters and digits and returns the product of the digits.

Example: `productDigits("a2514b")` returns 40, because $2 \times 5 \times 1 \times 4 = 40$.

29. Write a function called `total_time` that takes a string `"Hours:Minutes:Seconds"`. Hours, minutes, and seconds may each have any number of digits. Return the total number of seconds.

30. Write a function called `isValid` that takes three integer side lengths. Return `True` when the sum of any two sides is greater than the third side, and `False` otherwise.

31. Write a function called `sumFile` that takes a file name and returns the sum of the numbers in that file. Each line contains exactly one number.

32. Write a function called `interleave` that takes two strings, `s1` and `s2`, of equal length. Return a new string that takes a character from `s1`, then a character from `s2`, then the next from `s1`, and so on.

Example: `interleave("abc", "xyz")` returns `"axbycz"`.

Solutions

1. Each pass adds 1 first. When the result is even, it adds 2 more before printing. The printed values are 1, 4, 5, 8, 9, 12.

2.

```
0 1
0 2
0 3
0 4
0 5
1 2
1 3
1 4
1 5
2 3
2 4
2 5
```

3. The gaps are 1, then 2, then 3, and so on. The last value printed is 92.

```
n = 1
step = 1
while n < 100:
    print(n)
    n += step
    step += 1
```

4.

```
def count_evens(s):
    count = 0
    for ch in s:
        if ch.isdigit() and int(ch) % 2 == 0:
            count += 1
    return count
```

5.

```
def common_factors(a, b):
    count = 0
    limit = a
    if b < a:
        limit = b
    for i in range(1, limit + 1):
        if a % i == 0 and b % i == 0:
            count += 1
    return count
```

6. Equal letters are not in order, because the next letter must come strictly later.

```
def test_alphabetical(s):
    for i in range(1, len(s)):
        if s[i] <= s[i - 1]:
            return False
    return True
```

7. A tie can be printed as `Tie`. The same-time lines are printed as they are found. The gap is the difference of the two times, measured in minutes from midnight. For part d, a run is counted once, when the second

vote of that run appears. Later votes in the same run do not add another count.

```
def main():
    infile = open("results.txt", "r")
    alice = 0
    bob = 0
    prev_minutes = None
    prev_name = None
    max_gap = 0
    alice_runs = 0
    bob_runs = 0
    in_run = False

    for line in infile:
        hour, minute, name = line.split()
        hour = int(hour)
        minute = int(minute)
        minutes = hour * 60 + minute
        if name == "Alice":
            alice += 1
        else:
            bob += 1

        if prev_name is not None:
            if minutes == prev_minutes:
                print(hour, minute)
            gap = minutes - prev_minutes
            if gap > max_gap:
                max_gap = gap
            if name == prev_name and not in_run:
                if name == "Alice":
                    alice_runs += 1
                else:
                    bob_runs += 1
                in_run = True
            if name != prev_name:
                in_run = False

        prev_minutes = minutes
        prev_name = name

    infile.close()
    if alice > bob:
        print("Alice")
    elif bob > alice:
        print("Bob")
    else:
        print("Tie")
    print(max_gap)
    print(alice_runs, bob_runs)

main()
```

8.

```
def find_biggest_letter(s):
    biggest = s[0]
    for ch in s:
```

```
        if ch > biggest:
            biggest = ch
    return biggest
```

9.

```
age = int(input("Age? "))
while age < 18:
    print("You must be at least 18.")
    age = int(input("Age? "))
```

10.

```
def roman(n):
    result = ""
    while n > 0:
        if n >= 1000:
            result += "M"
            n -= 1000
        elif n >= 500:
            result += "D"
            n -= 500
        elif n >= 100:
            result += "C"
            n -= 100
        elif n >= 50:
            result += "L"
            n -= 50
        elif n >= 10:
            result += "X"
            n -= 10
        elif n >= 5:
            result += "V"
            n -= 5
        else:
            result += "I"
            n -= 1
    return result
```

Challenge, with the subtraction pairs checked before the plain symbols:

```
def roman_real(n):
    result = ""
    while n > 0:
        if n >= 1000:
            result += "M"
            n -= 1000
        elif n >= 900:
            result += "CM"
            n -= 900
        elif n >= 500:
            result += "D"
            n -= 500
        elif n >= 400:
            result += "CD"
            n -= 400
        elif n >= 100:
            result += "C"
            n -= 100
```

```
        elif n >= 90:
            result += "XC"
            n -= 90
        elif n >= 50:
            result += "L"
            n -= 50
        elif n >= 40:
            result += "XL"
            n -= 40
        elif n >= 10:
            result += "X"
            n -= 10
        elif n >= 9:
            result += "IX"
            n -= 9
        elif n >= 5:
            result += "V"
            n -= 5
        elif n >= 4:
            result += "IV"
            n -= 4
        else:
            result += "I"
            n -= 1
    return result
```

11.

```
import random

secret = random.randint(1, 100)
guess = int(input("Guess? "))
while guess != secret:
    if guess < secret:
        print("too low")
    else:
        print("too high")
    guess = int(input("Guess? "))
print("You got it.")
```

12. a. `str`

13. a, d, and e.

14. b. `find`. It returns `-1` when the substring is absent. `startswith` and `endswith` only check one end, and `replace` always returns a string.

15. `"Hello world"`

16. 45, the sum of the integers from 1 through 9.

17. 20, the sum of 2, 4, 6, and 8.

18. An index.

19. Immutable.

20. A slice. It is also called a substring.

21. `open("scores.txt", "w")`

22. `open("scores.txt", "r")`

23. `infile.readline()`

24.

a. `"Pr"`

b. `"ra"`

c. 18

d. 4

e. `False`

f. `"Programming is awesome"`

g. `"programming is fun"`

25. The letter `a`, printed on four lines.

26. 2. The parameter `n` is local to `nPrint`, so changing it does not change `k`.

27.

```
3
1
```

`f1` prints `x + 2`. The `x` in `main` is a different variable and stays 1.

28. If the string has no digits, this version returns 1.

```
def productDigits(s):
    product = 1
    for ch in s:
        if ch.isdigit():
            product *= int(ch)
    return product
```

29.

```
def total_time(text):
    parts = text.split(":")
    hours = int(parts[0])
    minutes = int(parts[1])
    seconds = int(parts[2])
    return hours * 3600 + minutes * 60 + seconds
```

30.

```
def isValid(a, b, c):
    return a + b > c and a + c > b and b + c > a
```

31.

```
def sumFile(filename):
    infile = open(filename, "r")
    total = 0
    for line in infile:
        total += float(line)
    infile.close()
    return total
```

32.

```
def interleave(s1, s2):
    result = ""
    for i in range(len(s1)):
        result += s1[i] + s2[i]
    return result
```