Python Strings and Files — Practice

CSCI 40: Computing for the Web

Fall 2026

How current LLMs scored on this practice bank *2026-08-26*

Model	Reasoning	Score
claude-haiku-4.5	low effort	39/39 (100%)
claude-sonnet-5	low effort	38/39 (97%)
claude-fable-5	low effort	39/39 (100%)
claude-opus-4.8	low effort	37/39 (95%)
gpt-5.6-luna	low reasoning	37/39 (95%)
gpt-5.4	low reasoning	38/39 (97%)
gpt-5.6-terra	low reasoning	37/39 (95%)
gpt-5.5	low reasoning	39/39 (100%)
gpt-5.6-sol	low reasoning	37/39 (95%)
gemini-3.1-flash-lite	minimal thinking	32/39 (82%)
gemini-3.5-flash-lite	minimal thinking	31/39 (79%)
gemini-3.6-flash	minimal thinking	37/39 (95%)
gemini-3.7-flash	low thinking	37/39 (95%)

One independent, tool-free prompt per question; deterministic executable ground truth.

Python Strings

1 Basic String Operations

Note 1. Indexing and slicing of strings works similarly to indexing and slicing of lists. The main difference is that the *type* of the result when indexing/slicing a string is always another string.

Problem 2. What is the output of the following python code?

```
s = 'hello world'
print(s[:-6])
```

Problem 3. What is the output of the following python code?

```
s = '12345'
xs = [1, 2, 3, 4, 5]
b = s[-1] == xs[-1]
print('b=', b)
```

Problem 4. What is the output of the following python code?

```
s = 'hello'
xs = ['h', 'e', 'l', 'l', 'o']
b = s[-2:] == xs[-2:]
print('b=', b)
```

Problem 5. What is the output of the following python code?

```
s = 'python␣is␣awesome'
if 'python' in s:
    print('True')
else:
    print('False')
```

Problem 6. What is the output of the following python code?

```
s = 'python␣is␣awesome'
if 'Python' in s:
    print('True')
else:
    print('False')
```

Note 7. Python strings support a number of methods that lists do not support. It is important to remember that strings are *immutable*, and so methods will never change the content of a string.

Problem 8. What is the output of the following python code?

```
s = 'python␣is␣awesome'
t = s.replace('python', 'everything')
print('t=', t)
```

Problem 9. What is the output of the following python code?

```
s = 'Python␣is␣awesome'
s.replace('python', 'everything')
print('s=', s)
```

Problem 10. What is the output of the following python code?

```
s = 'python␣is␣awesome'
t = s.split()
u = t[-1]
print('u=', u)
```

Problem 11. What is the output of the following python code?

```
s = 'python\nis\tawesome'
t = s.split()
u = '␣'.join(t)
print('u=', u)
```

Problem 12. What is the output of the following python code?

```
s = 'python␣␣is\tawesome'
t = s.split('␣')
u = t[1]
print('u=', u)
```

Problem 13. What is the output of the following python code?

```
s = 'python␣is␣awesome'
t = s.split('o')
u = 'XXX'.join(t)
print('u=', u)
```

Problem 14. What is the output of the following python code?

```
s = 'guido␣van␣rossum'
i = s.find('␣')
t = s[:i]
print('t=',t)
```

Problem 15. What is the output of the following python code?

```
s = 'guido␣van␣rossum'
i = s.find('G')
t = s[:i]
print('t=',t)
```

Note 16. Strings are represented internally as binary numbers (i.e. 1s and 0s). Special conversion tables are used to map these binary numbers into semantically meaningful strings. The process of going from strings to numbers is called *encoding* and the process of going from numbers to strings is called *decoding*. You are responsible for understanding:

1. ASCII encoding/decoding,
2. the difference between *alphabetical* ordering and *ASCIIbetical* ordering, and
3. basic binary/octal/hexadecimal math.

Problem 17. What is the output of the following python code?

```
b = 'G' > 'a'
print('b=', b)
```

Problem 18. What is the output of the following python code?

```
s = 'Guido␣van␣Rossum'
xs = s.split()
xs.sort()
print('xs=', xs)
```

Problem 19. What is the output of the following python code?

```
s = 'Guido␣van␣rossum'
xs = s.split()
xs.sort()
print('xs=', xs)
```

Problem 20. What is the output of the following python code?

```
print(0b1010)
```

Problem 21. What is the output of the following python code?

```
print(0o110)
```

Problem 22. What is the output of the following python code?

```
print(0x0010)
```

Problem 23. What is the output of the following python code?

```
print(chr(ord('a') - 32))
```

Problem 24. What is the output of the following python code?

```
print(chr(ord('f') - 3))
```

Note 25. Escape codes let us enter hard-to-type characters into strings. The special character \ (backslash) is used to begin an escape code. In python, there are four methods of entering "strings" that handle escape codes differently:

1. normal strings `'...'`
2. r-strings `r'...'`
3. f-strings `f'...'`
4. bytes `b'...'`

In the problems below, I will only write normal strings for brevity. But you should be able to solve the problems for any type of string.

Problem 26. What is the output of the following python code?

```
s = '\x48\x45\x4C\x4C\x4F'
print('s=', s)
```

Problem 27. What is the output of the following python code?

```
name = 'world'
s = 'hello␣{name}!'
print('s=', s)
```

Problem 28. What is the output of the following python code?

```
b = "\""␣>␣'\n'
␣print('b=',␣b)
```

Problem 29. What is the output of the following python code?

```
s = '\x57o\x72L\x44'
print('s=', s)
```

Problem 30. What is the output of the following python code?

```
x = '"'>"'"
print('x=', x)
```

Problem 31. What is the output of the following python code?

```
x = '">"'
print('x=', x)
```

Problem 32. What is the output of the following python code?

```
xs = ['there,',"isn't","[a','syntax]",']error']
print(xs[2])
```

Problem 33. What is the output of the following python code?

```
xs = ['there,',"isn't","[a','syntax]",']error']
print(xs[3])
```

2 Manipulating Markdown

Note 34. The following problems are designed to get you practice for you markdown compiler assignment. They contain a combination of correct and incorrect implementations for functions that are needed for that assignment.
Recall that *reading* is generally easier than *writing*, and so I strongly recommend that you work through these functions before beginning the coding of your markdown compiler assignment.

Problem 35. What is the output of the following python code?

```
def compile_italic_star(line):
    result = ''
    i = 0
    while i < len(line):
        if line[i] == '*':
            if i + 1 < len(line) and '*' in line[i+1:]:
                end = line.find('*', i+1)
                result += '<i>' + line[i+1:end] + '</i>'
                i = end + 1
            else:
                result += '*'
                i += 1
        else:
            result += line[i]
            i += 1
    return result
result = compile_italic_star('alpha␣*beta*␣gamma␣*delta')
print(result)
```

Problem 36. What is the output of the following python code?

```
def compile_italic_star(line):
    result = ''
    i = 0
    while i < len(line):
        if line[i] == '*':
            end = line.find('*', i+1)
            if end != -1:
                result += '<i>' + line[i+1:end] + '</i>'
                i = end + 1
            else:
                i = len(line)
        else:
            result += line[i]
            i += 1
    return result
result = compile_italic_star('alpha␣*beta*␣gamma␣*delta')
print(result)
```

Problem 37. What is the output of the following python code?

```
def compile_italic_star(line):
    result = ''
    i = 0
    while i < len(line):
        if line[i] == '*':
            end = line.find('*', i+1)
            if end != -1:
                result += '<i>' + line[i+1:end] + '</i>'
                i = end + 1
            else:
                i = len(line)
        else:
            result += line[i]
            i += 1
    return result
result = compile_italic_star('alpha␣*beta*␣gamma␣*delta*')
print(result)
```

Problem 38. What is the output of the following python code?

```
def compile_italic_star(line):
    result = ''
    i = 0
    while i < len(line):
        if line[i] == '*':
            end = line.find('*', i+1)
            if end != -1:
                result += '<i>' + line[i+1:end] + '</i>'
                i = end + 1
            else:
                i = len(line)
        else:
            result += line[i]
            i += 1
    return result
result = compile_italic_star('alpha␣*beta␣gamma␣delta')
print(result)
```

Problem 39. What is the output of the following python code?

```
def compile_italic_star(line):
    result = ''
    i = 0
    while i < len(line):
        if line[i] == '*':
            end = line.find('*', i+1)
            if end != -1:
                result += '<i>' + line[i+1:end] + '</i>'
                i = end + 1
            else:
                i += 1
        else:
            result += line[i]
            i += 1
    return result
result = compile_italic_star('alpha␣*beta*␣gamma␣*delta')
print(result)
```

Problem 40. What is the output of the following python code?

```
def compile_italic_star(line):
    result = ''
    i = 0
    while i < len(line):
        if line[i] == '*':
            end = line.find('*', i+1)
            if end != -1:
                result += '<i>' + line[i+1:end] + '</i>'
                i = end + 1
            else:
                i += 1
        else:
            result += line[i]
            i += 1
    return result
result = compile_italic_star('alpha␣*beta␣gamma␣delta')
print(result)
```

Problem 41. What is the output of the following python code?

```
def compile_italic_star(line):
    result = ''
    i = 0
    while i < len(line):
        if line[i] == '*':
            end = line.find('*', i+1)
            if end != -1:
                result += '<i>' + line[i+1:end] + '</i>'
                i = end + 1
            else:
                i += 1
        else:
            result += line[i]
            i += 1
    return result
result = compile_italic_star('alpha␣*beta␣gamma␣delta*')
print(result)
```

Problem 42. What is the output of the following python code?

```
def compile_bold_stars(line):
    start = line.find('**')
    if start == -1 or len(line) < 4:
        return line

    end = line[start + 2:].find('**')
    if end == -1:
        return line

    end = end + start + 2
    return line[:start] + '<b>' + line[start + 2:end] + '</b>' + line[end + 2:]
result = compile_bold_stars('alpha␣**beta**␣gamma␣**delta**')
print(result)
```

Problem 43. What is the output of the following python code?

```
def compile_bold_stars(line):
    start = line.find('**')
    if start == -1 or len(line) < 4:
        return line

    end = line[start + 2:].find('**')
    if end == -1:
        return line

    end = end + start + 2
    return line[:start] + '<b>' + line[start + 2:end] + '</b>' + line[end + 2:]
result = compile_bold_stars('alpha␣**beta**␣gamma␣**delta')
print(result)
```

Problem 44. What is the output of the following python code?

```
def compile_bold_stars(line):
    start = line.find('**')
    if start == -1 or len(line) < 4:
        return line

    end = line[start + 2:].find('**')
    if end == -1:
        return line

    end = end + start + 2
    return line[:start] + '<b>' + line[start + 2:end] + '</b>' + line[end + 2:]
result = compile_bold_stars('alpha␣beta␣gamma␣**delta')
print(result)
```