Review

- String processing
  - Basics of string
  - Index of string
  - Length of string, len()
  - Loop and counting
  - Slicing strings
  - Conditional statement
  - String library and functions
- File processing
  - Opening files
  - File processing
  - Searching through a file
  - Writing to a file

Index of String

- Indexes start at 0 (first character).
- Negative indexes count backward from the end (-1 is last character).

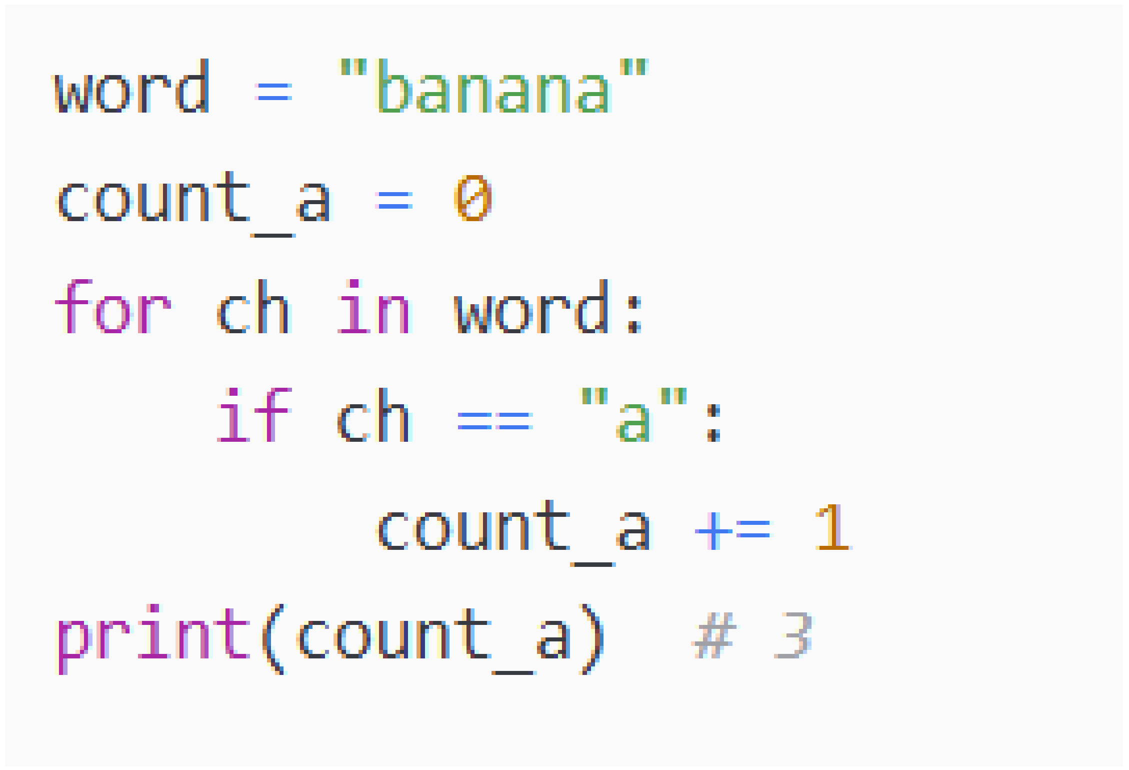
```python
text = "Python"
print(text[0])   # P
print(text[-1])  # n
```

- Counts the number of characters in a string (including spaces and punctuation).

```python
text = "Python"
print(len(text))    # 6
```

Loop and Counting

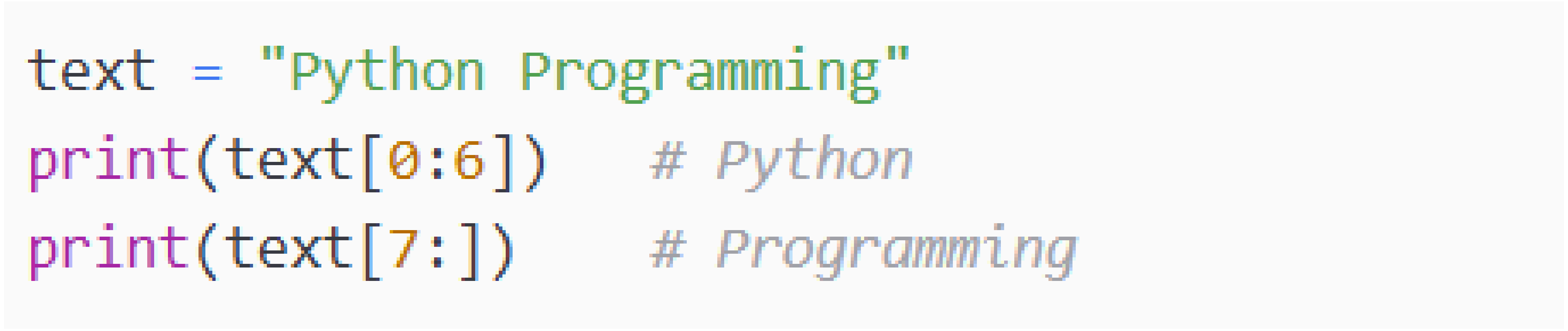
- Loop through each character in a string using for ... in.
- Count characters with conditions.



```
word = "banana"
count_a = 0
for ch in word:
    if ch == "a":
        count_a += 1
print(count_a)  # 3
```

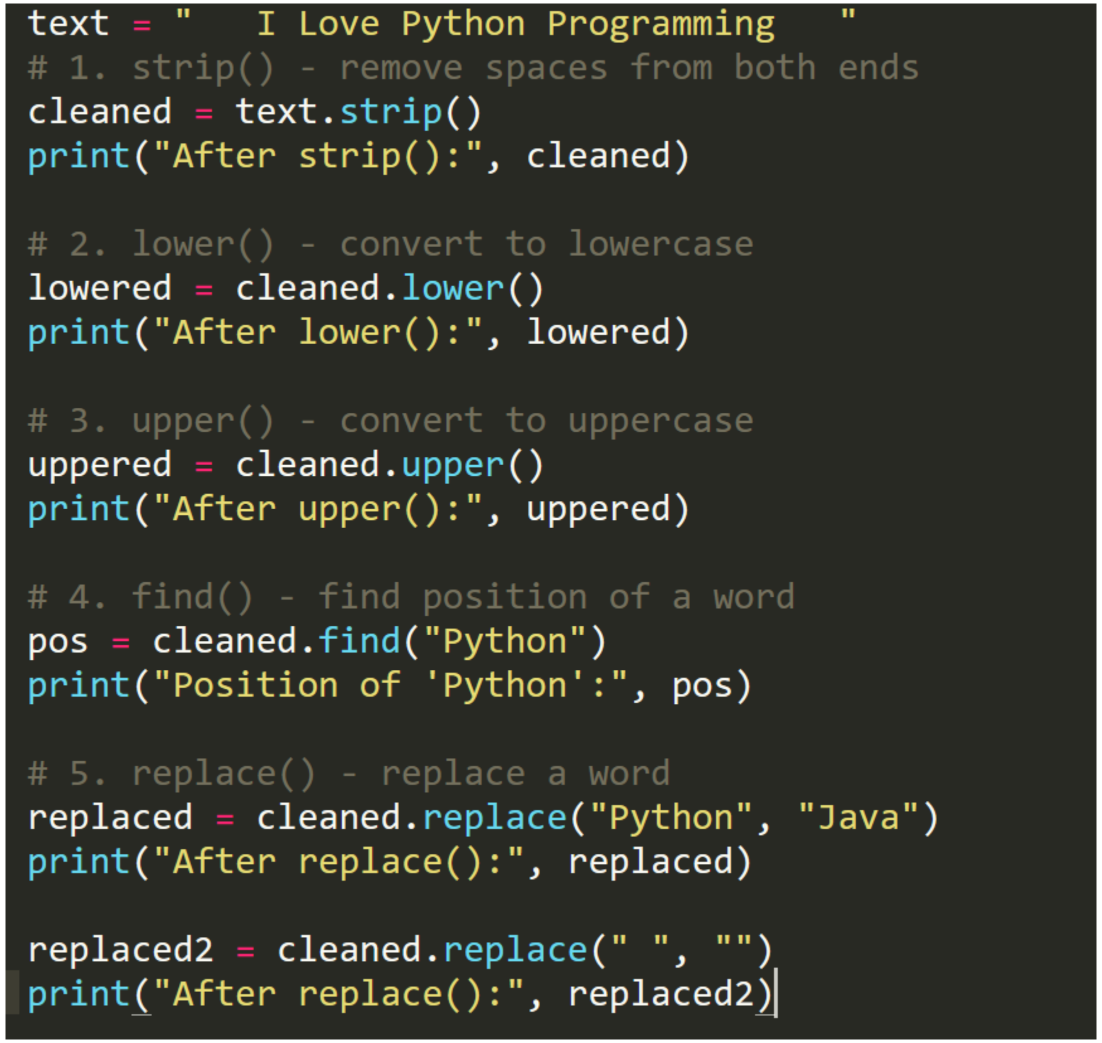
Slicing Strings

- Use string[start:end] to get part of a string.
- End index is not included.

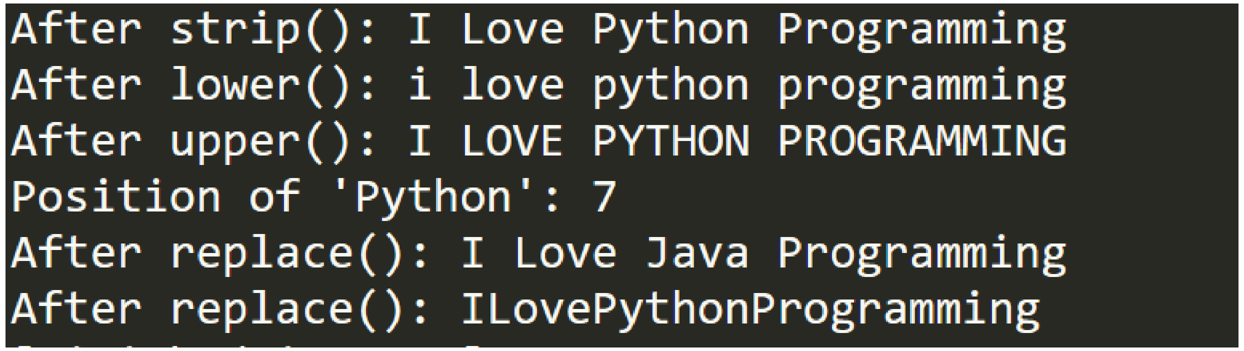


```python
text = "Python Programming"
print(text[0:6])   # Python
print(text[7:])    # Programming
```

Built-in functions for string



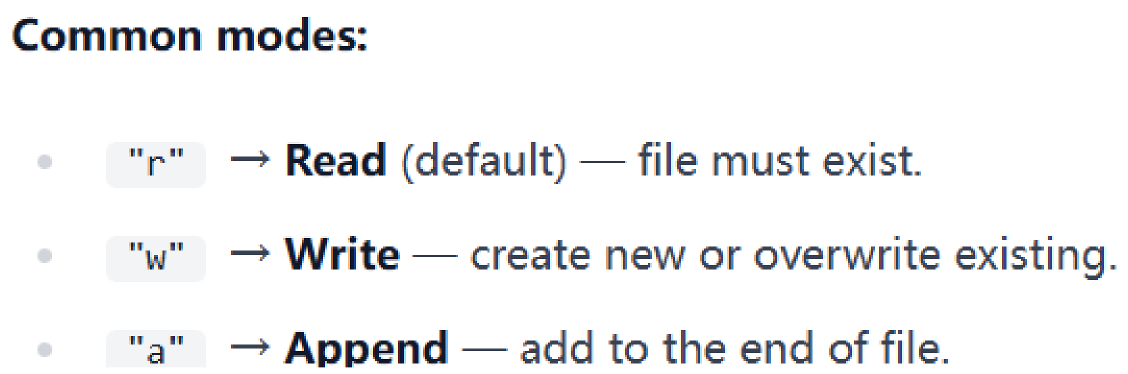
```python
text = "   I Love Python Programming   "
# 1. strip() - remove spaces from both ends
cleaned = text.strip()
print("After strip():", cleaned)

# 2. lower() - convert to lowercase
lowered = cleaned.lower()
print("After lower():", lowered)

# 3. upper() - convert to uppercase
uppered = cleaned.upper()
print("After upper():", uppered)

# 4. find() - find position of a word
pos = cleaned.find("Python")
print("Position of 'Python':", pos)

# 5. replace() - replace a word
replaced = cleaned.replace("Python", "Java")
print("After replace():", replaced)

replaced2 = cleaned.replace(" ", "")
print("After replace():", replaced2)
```



```
After strip(): I Love Python Programming
After lower(): i love python programming
After upper(): I LOVE PYTHON PROGRAMMING
Position of 'Python': 7
After replace(): I Love Java Programming
After replace(): ILovePythonProgramming
```

File processing

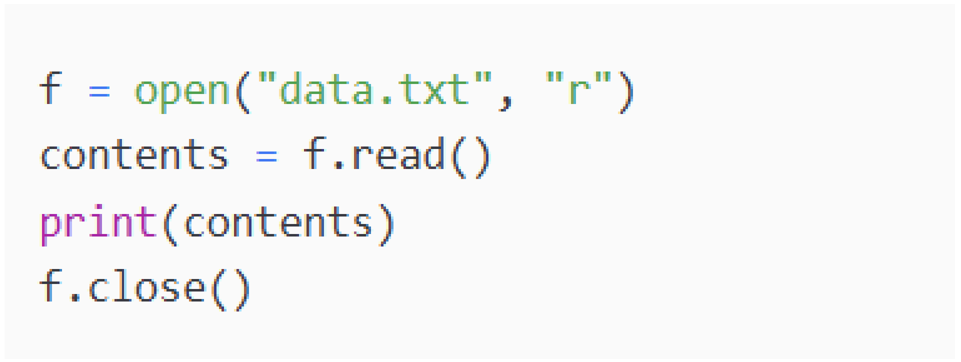
- We use open(filename, mode) to get a file handle.



Common modes:

- `"r"` → **Read** (default) — file must exist.
- `"w"` → **Write** — create new or overwrite existing.
- `"a"` → **Append** — add to the end of file.

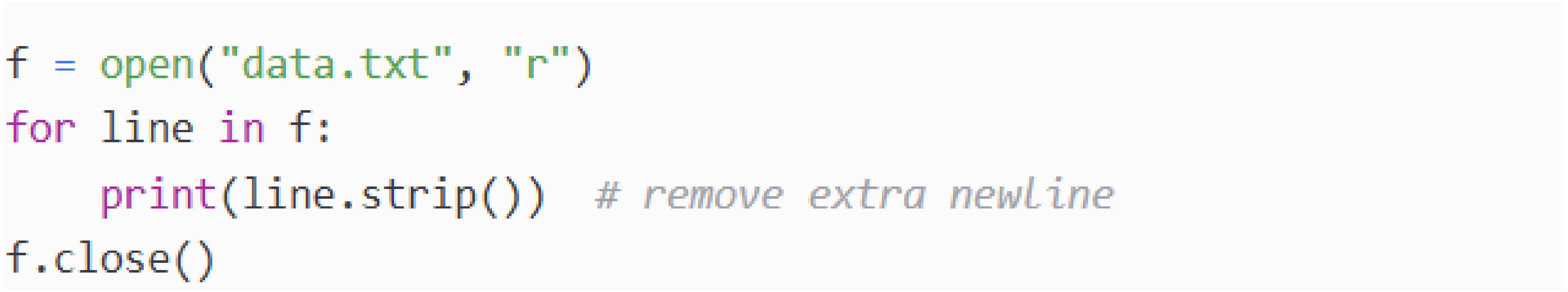
- Read entire file at once:



```
f = open("data.txt", "r")
contents = f.read()
print(contents)
f.close()
```

File processing

- Read line-by-line with a loop:



```
f = open("data.txt", "r")
for line in f:
    print(line.strip())  # remove extra newline
f.close()
```

File processing

- Write text (overwrite mode "w"):

```python
f = open("notes.txt", "w")
f.write("Hello, world!\n")
f.write("Python file writing.\n")
f.close()
```

Append text (mode "a"):

```python
f = open("notes.txt", "a")
f.write("This line is added at the end.\n")
f.close()
```

`"w"` → **overwrite file completely** (removes old content).

`"a"` → **append new content** after what's already in the file.

- Using with closes the file automatically

```python
with open("data.txt", "r") as f:
    for line in f:
        print(line.strip())
```