Programming Fundamentals
Lab 04

Iteration & Problem Solving

Looping Patterns

```
count = 0
for i in range(n):
    if s[i] == "C":
        count += 1
```

Comp 102 — Forman Christian University

Spring 2026

Estimated Time: ∼**3 hours** — Based on Lectures 7 & 8

How to Use This Lab

This lab is a **walkthrough tutorial**—it is designed to guide you through learning, not to test you. Work through each section at your own pace:

- **Predict** — Write down what you think will happen *before* touching the computer.
- **Type** — Enter the code in Thonny's **Code Editor** (top pane) and press **Run** (or **F5**).
- **Verify** — Compare your prediction with the actual result. If they differ, figure out *why*.
- Exercises are graded by difficulty:
  - ★ Easy — Immediate practice. Follow the pattern you just saw.
  - ★★ Medium — Requires some thinking. Combines concepts.
  - ★★★ Hard — Stretch goal. It is OK to need help!
- **Ask for help** whenever you are stuck for more than 5 minutes.

Prerequisites

This lab assumes you completed **Lab 03**. You should be comfortable with variables, `input()`, type conversion with `int()`, basic `if` statements, and string indexing (`s[0]`) and length (`len(s)`).

Predict – Type – Verify

Before you run any loop, write a short prediction for what will print. Use Thonny's **Code Editor** for multi-line loops; use the Shell only for quick one-line experiments. If Lecture 7 used `repeat` and `while` puzzles, treat them as Python `for` and `while` loops here.

1 While Loops: Repeat Until a Condition (~35 min)

Ref: Lecture 7 — Loops

A `while` loop repeats as long as its condition is `True`. Make sure something inside the loop changes so it can eventually stop.

Common Loop Pitfalls

If the loop variable never changes, the loop never ends (infinite loop). Also remember: `range(n)` produces values from `0` to `n-1`.

1. ★ Easy Predict the output, then verify:

```
i = 1
while i <= 3:
    print(i)
    i = i + 1
print("done")
```

Your prediction:

2. ★★ Medium Trace this loop. What prints?

```
n = 7
while n > 0:
    print(n)
    n = n - 2
print("blastoff")
```

Your prediction:

3. ★★ Medium Fill in the trace table. Start with `i = 0` and `total = 0`.

```
i = 0
total = 0
while i < 4:
    i = i + 1
    total = total + i
```

After step	(i, total)
1	(_____, _____)
2	(_____, _____)
3	(_____, _____)
4	(_____, _____)

4. ★★★ Hard **Write a program** that keeps asking for a secret word until the user types `"loop"`. When the word is correct, print `"Access granted"`.

Write your code:

5. ★★★ Hard **Smallest power of 2.** Read an integer `n`. Start with `p = 1` and keep doubling until `p >= n`. Print `p`. Example: if `n = 20`, the program prints `32`.

Write your code:

2 Counters & Accumulators (~30 min)

Ref: Lecture 7 — Loop Patterns

Many loops use a **counter** (how many times) or an **accumulator** (a running total).

6. ★ Easy Predict the output:

```
total = 0
i = 1
while i <= 4:
    total = total + i
    i = i + 1
print(total)
```

Your prediction: _____

7. ★★ Medium Trace this loop and predict the final values of `count` and `total`:

```
count = 0
total = 0
i = 2
while i <= 8:
    total = total + i
    count = count + 1
    i = i + 2
print(count, total)
```

count: _____ **total:** _____

8. ★★★ Hard **Write a program** that reads an integer `n` and prints `n!` using a `while` loop. Test with `n = 4` (should print 24).

Write your code:

3 `for` Loops and `range` (~25 min)

Ref: Lecture 7 — `for` Loops and `range`

A `for` loop runs a fixed number of times. The `range` function generates the sequence of values for the loop variable.

9. ★ Easy Predict the output:

```
for i in range(5):
    print(i)
```

Your prediction:

10. ★★ Medium What prints?

```
for i in range(3, 8):
    print(i)
```

Your prediction:

Then predict:

```
for j in range(10, 4, -2):
    print(j)
```

Your prediction:

11. ★★★ Hard **Write a program** that asks for `n` and prints the squares from 1 to `n` (one per line) using a `for` loop. **Sample run:**

```
Input:
5
Output:
```

```
1
4
9
16
25
```

Write your code:

4 Looping Over Strings (~30 min)

Ref: Lecture 7 — Looping over Strings

Strings are sequences. Loop by character with `for ch in s`, or by index with `for i in range(len(s))`.

12. ★ Easy Predict the output:

```
word = "loop"
for ch in word:
    print(ch)
```

Your prediction:

13. ★★ Medium **Write a program** that asks for a lowercase word and counts how many vowels it has. Print the count. *Use vowels: `a e i o u`.* **Sample run:**

```
Input:
casablanca
Output:
4
```

Write your code:

14. ★★★ Hard **Even positions.** Given a string `s` that contains only the letters `a`, `b`, `c` in consecutive sorted blocks (e.g. `"aaabbc"`), count how many times the letter `"a"` appears

at even indices (0,2,4,...) and print the count. **Sample run:**

```
Input:
aaabbc
Output:
2
```

Write your code:

5 CCC 2018 J2: Occupy Parking (Stage 1 — Understand & Trace) (~30 min)

Ref: Lecture 8 — CCC Problem Solving

You are given the number of parking spaces `n`, then two strings of length `n`: one for **yesterday** and one for **today**. Each character is either `C` (car) or `.` (empty). Your task is to count how many positions have a car on **both** days.

15. ★ Easy How many spaces have cars on both days?

```
n = 6
yesterday = "CC..C."
today = "C.CCC."
```

> **Your answer:** _____

16. ★★ Medium Trace the loop by filling the count after each step. Start with `count = 0`.

`n = 4  yesterday = "C.CC"  today = "CC.C"`

i	yesterday[i]	today[i]	count after step
0	C	C	_____
1	.	C	_____
2	C	.	_____
3	C	C	_____

17. ★★★ Hard Fill in the missing condition:

```
n = int(input())
yesterday = input()
today = input()
count = 0
for i in range(n):
    if ________________________________:
        count = count + 1
print(count)
```

6 CCC 2018 J2: Occupy Parking (Stage 2 — Implement) (~35 min)

Ref: Lecture 8 — CCC Problem Solving

Now implement the full program in Thonny.

18. ★ Easy Fill in the missing parts of this `while` loop version:

```
n = int(input())
yesterday = input()
today = input()
count = 0
```

```
i = 0
while __________________:
    if _____________________________:
        count = count + 1
    i = __________________
print(count)
```

19. ★★ Medium **Write the full program** (no blanks) and test it with this sample input:

```
5
CC..C
.CC.C
```

The program should print:

```
2
```

20. ★★★ Hard **Extension.** Modify your program to also count spaces that are **newly occupied** today (today is `C` but yesterday is `.`). Print two lines:

`Both days: <count>`

`New today: <count>`

7 Capstone Challenges (~20 min)

These problems combine everything from this lab. Write your solutions in Thonny's code editor.

21. ★ Easy **Digit counter.** Read a string that may contain letters and digits. Count how many digits it has. If the count is 0, print `"no digits"`. Otherwise, print the count. *Use digits: 0 1 2 3 4 5 6 7 8 9.* **Input:** one line string. **Output:** the digit count or `"no digits"`. **Sample run:**

```
Input:
comp102
Output:
3
```

Write your code:

22. ★★ Medium **First `a`.** Read a word and print the index of the first `"a"`. If there is no `"a"`, print `-1`. (Do not use `find`.) **Input:** one line word. **Output:** the first index of `"a"` or `-1`. **Sample run:**

```
Input:
```

```
banana
Output:
1
```

Write your code:

23. ★★★ Hard **Longest run.** Read a string and find the length of the longest run of the same character. Example: `"aaabbc"` → `3`. **Input:** one line string. **Output:** length of the longest run. **Sample run:**

```
Input:
aaabbc
Output:
3
```

Write your code:

Self-Assessment Checklist

Before you leave, check off what you can do:

- ☐ I can write a `while` loop and update a variable so it stops
- ☐ I can trace a loop step by step and predict its output
- ☐ I can use counters and accumulators inside loops
- ☐ I can use `for` with `range(start, stop, step)`
- ☐ I can loop over a string by character or by index
- ☐ I can use `in` to test membership in a string

- ☐ I can compare two same-length strings position by position
- ☐ I can translate a CCC J2 problem into inputs, a loop, and a condition
- ☐ I can test my program with sample inputs and explain the result