[¶ 4.1]

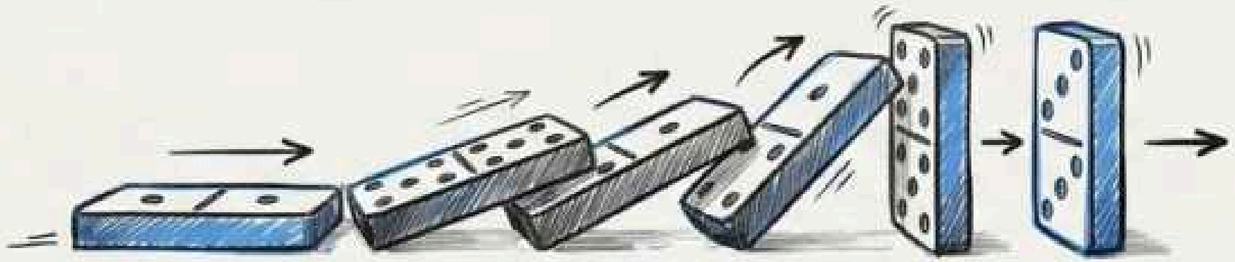
Lecture 4: Making Decisions and Repeating

- [ ] Conditionals
- [ ] Loops
- [ ] Ranges
- [ ] Accumulators
- [ ] Fizz Buzz
- [ ] Debugging

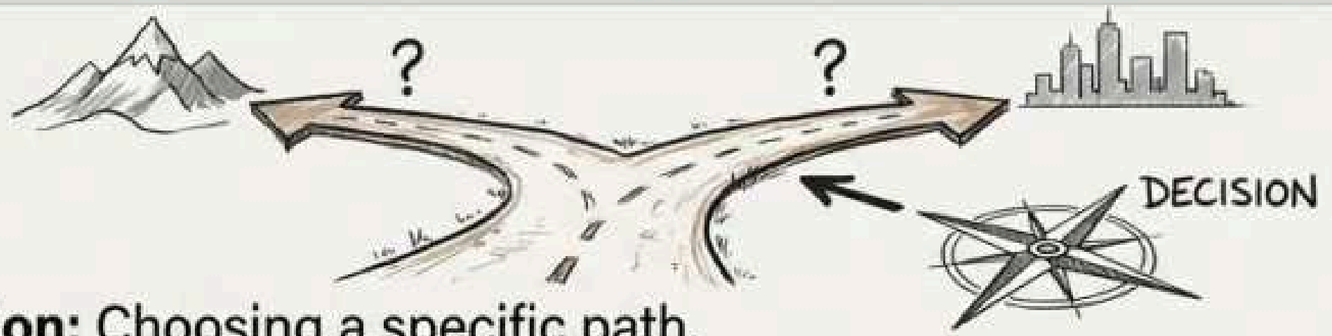


Real programs choose between actions and repeat them

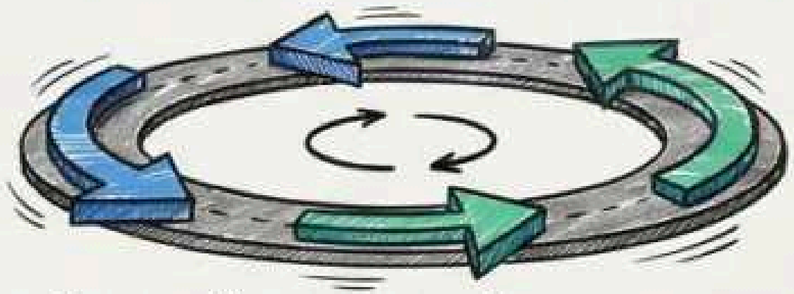
[¶ 4.2]



Sequence: Straight from first line to last.



Selection: Choosing a specific path.



Repetition: Repeating actions many times.

The **true** power of programming.

[¶ 4.3]

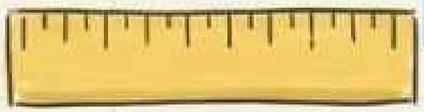
Anatomy of a Python conditional

The keyword

The boolean expression
(must be True or False)

The mandatory colon

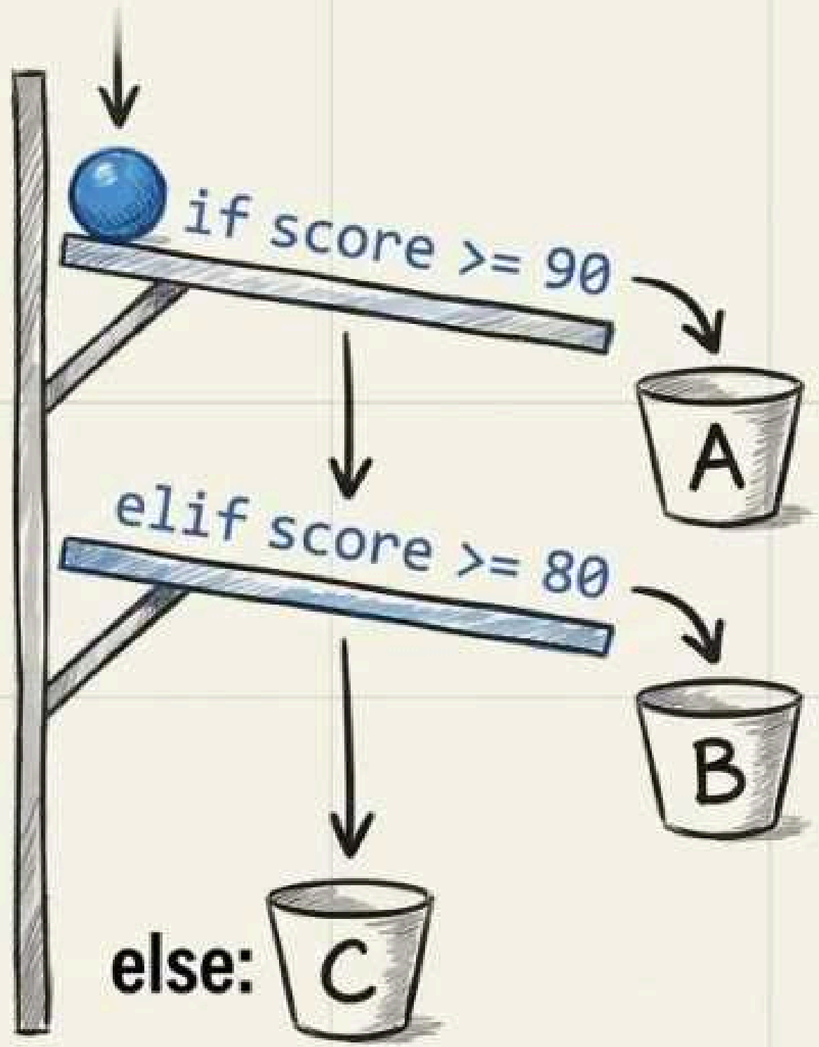
```
if age >= 18:
    print("adult")
```



The 4-space block indent
(not just decoration!)

[¶ 4.4]

Cascading alternatives catch the first true condition



Python checks top-to-bottom.

It executes the FIRST true condition.

It SKIPS the rest.

Order matters!

[¶ 4.5]

Combining conditions to map complex rules

(Divisible by 4) AND (NOT Divisible by 100)

OR (Divisible by 400))

1900 ⟶ False

2000 ⟶ True

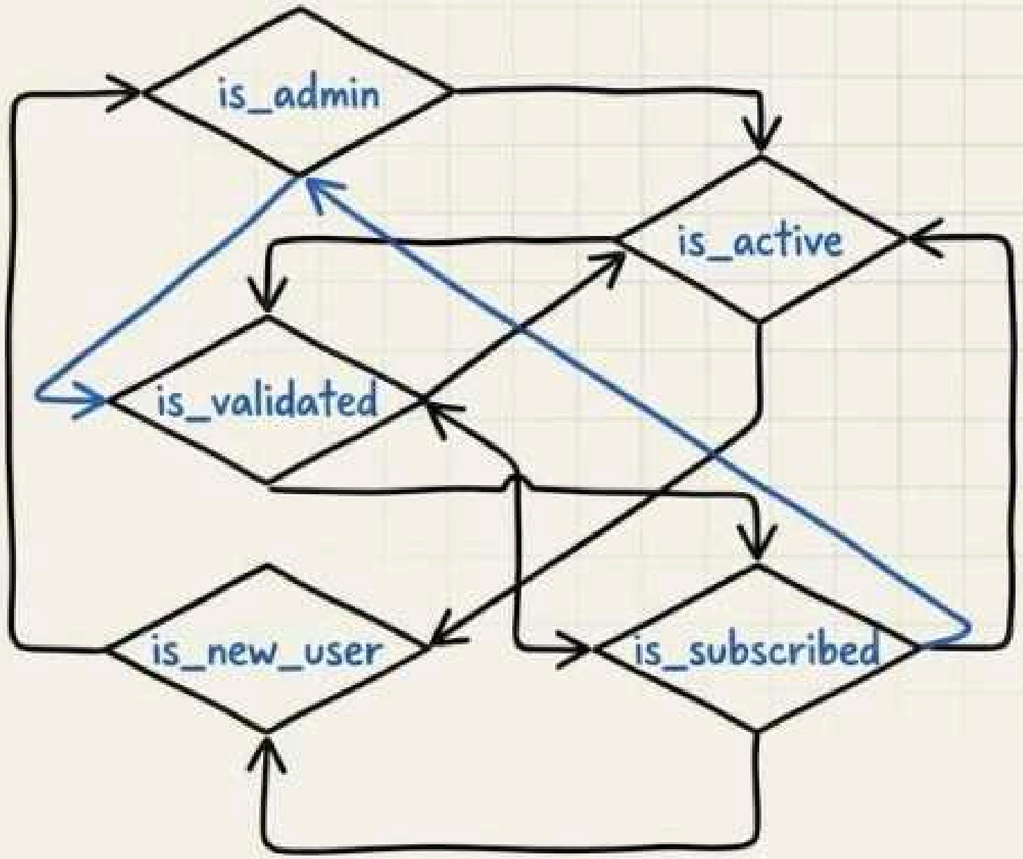
2024 ⟶ True

A leap year is divisible by 4, except for centuries, which must be divisible by 400.

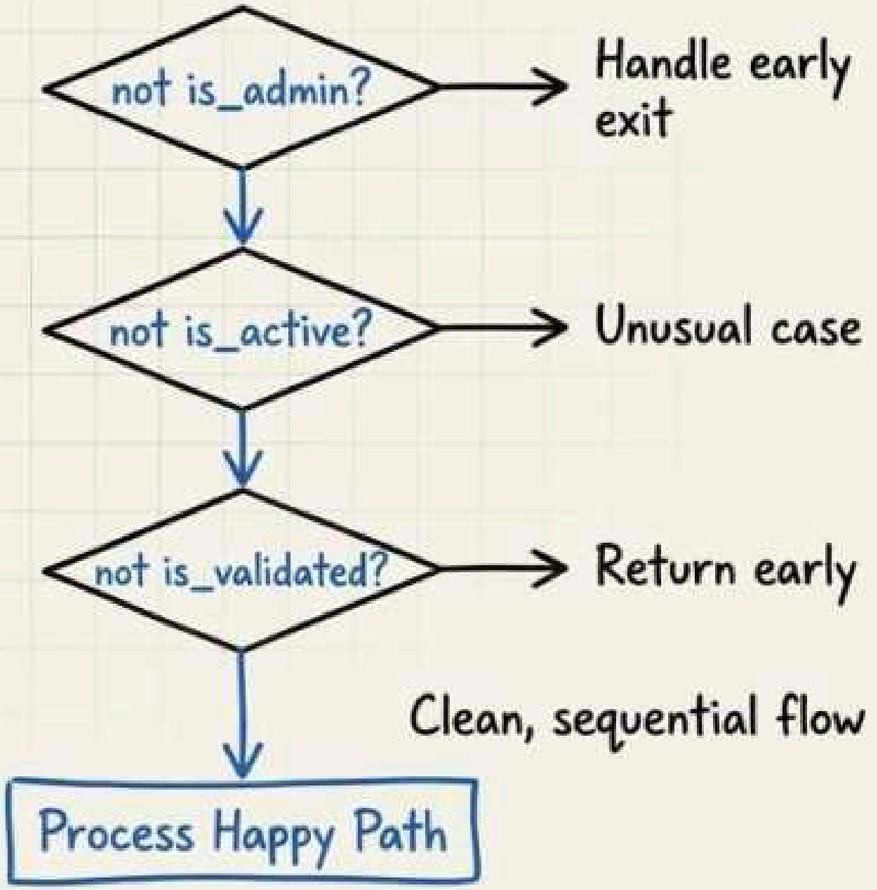
[¶ 4.6]

Keep conditions flat and handle the unusual cases first

Deep Nesting ✗

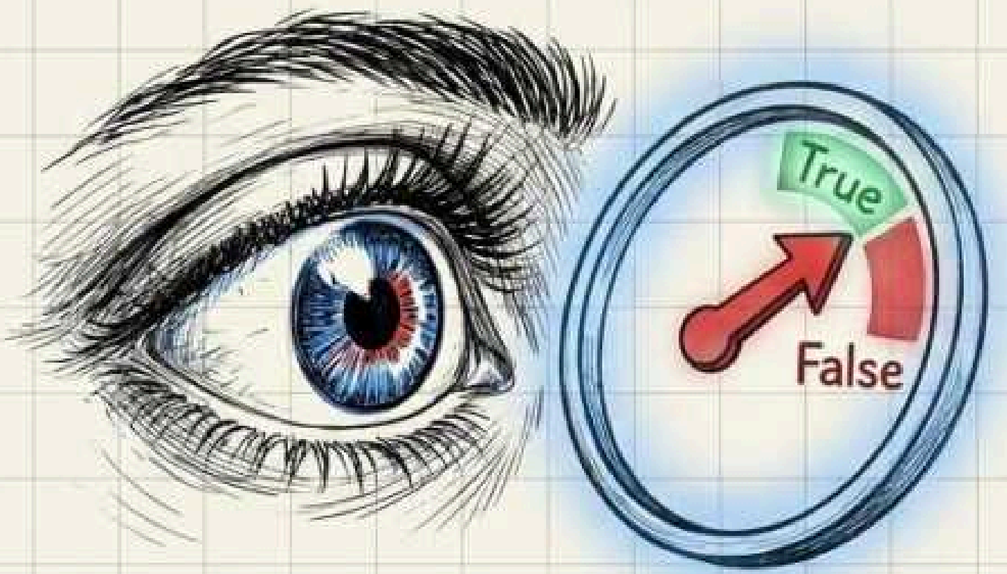


Best Practice ✓



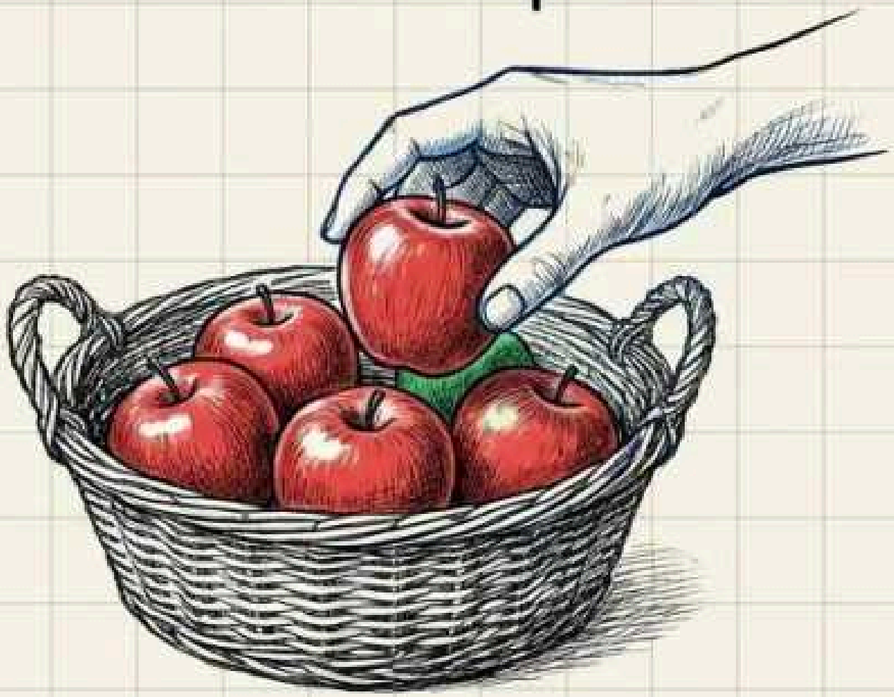
Two engines of repetition: `while` vs. `for`

`while` loop



Repeats as long as a condition stays true.

`for` loop



Repeats once for each item in a collection.

Choosing between them depends entirely on what you know in advance.

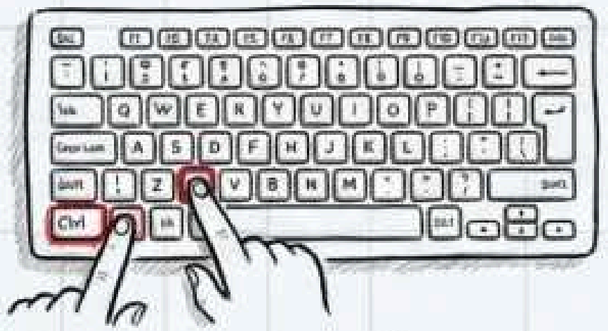
The while loop and the infinite trap

```
count = 3
while count > 0:
    print(count)
    count = count - 1
```

```
3
2
1
```



If the loop changes nothing, the condition never becomes false.



Emergency Escape

Generating sequences with range

[¶ 4.9]



Start is INCLUSIVE.
End is EXCLUSIVE.

```
total = 0
for number in range(1, 101):
    total = total + number
```

5050

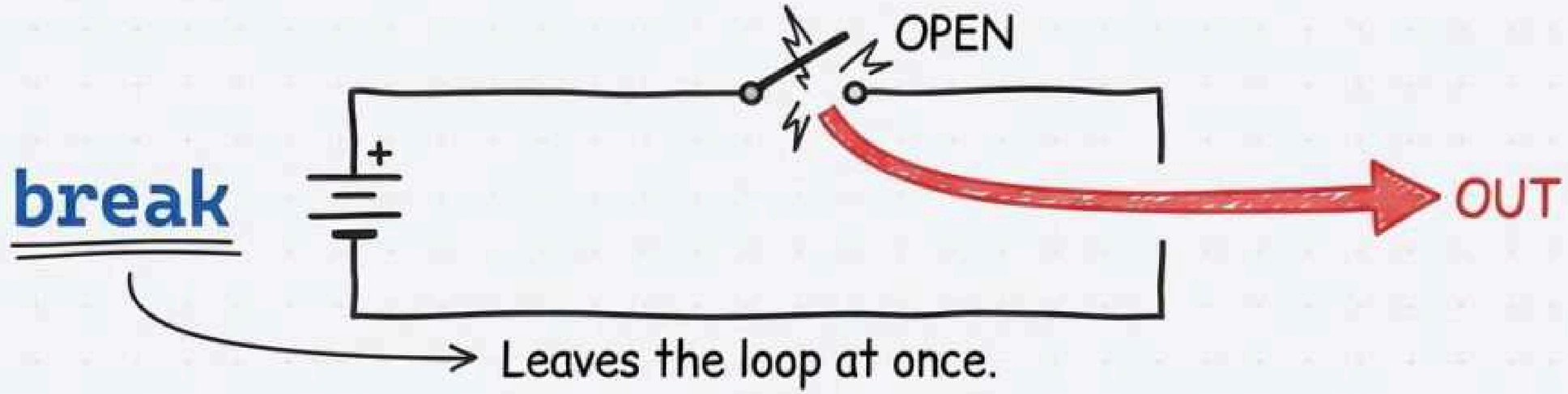
[¶ 4.10]

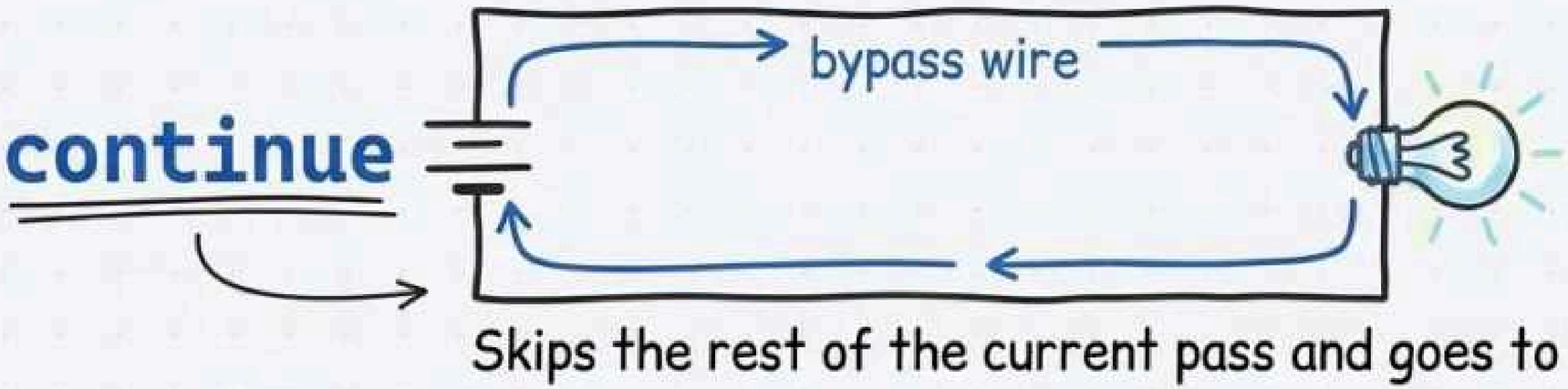
The accumulator pattern gathers value over time



- This identical pattern is used to:
  - Count items
  - Build lists
  - Find the largest value

Modifying the flow with **break** and **continue**

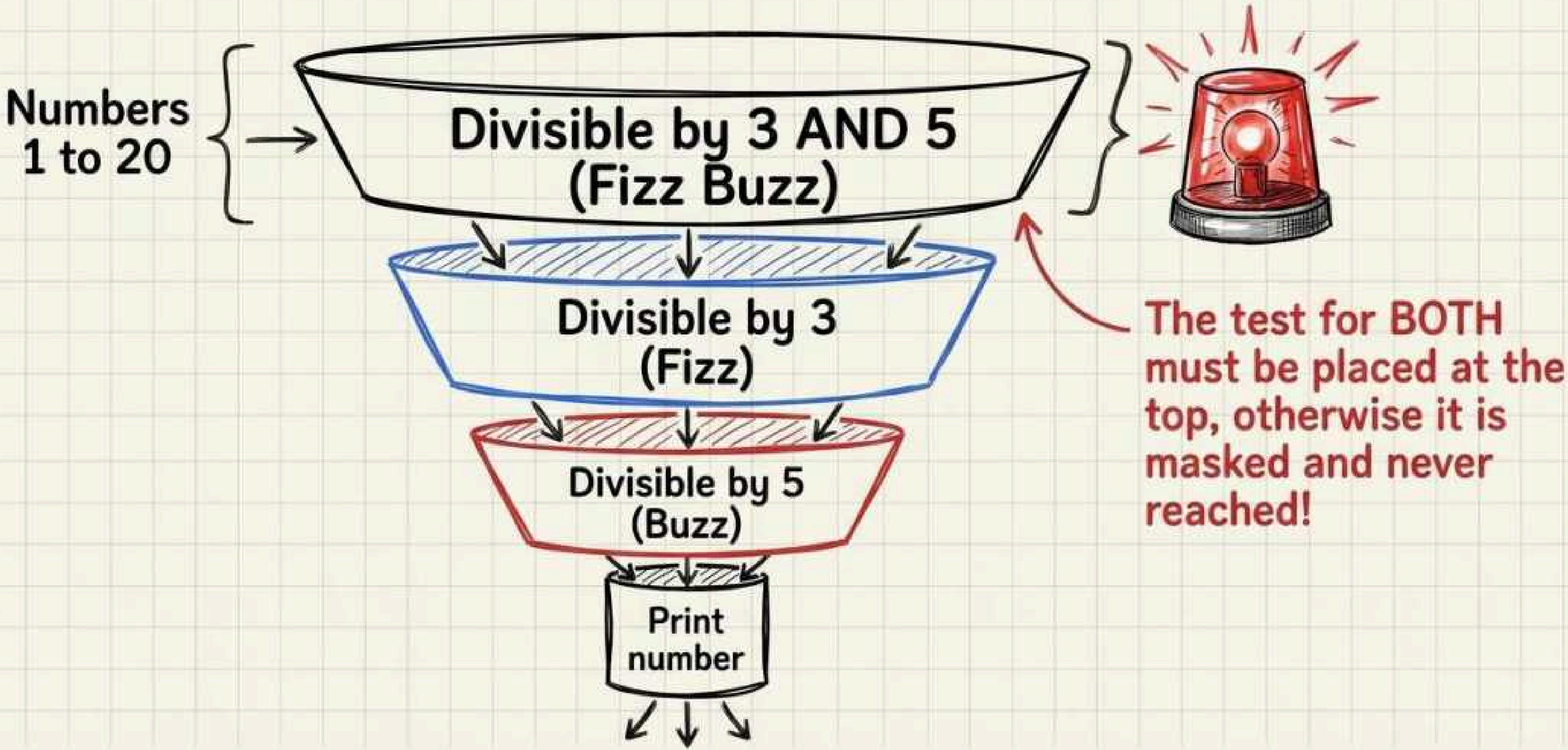




Use sparingly. Careless use makes program flow impossible to follow.

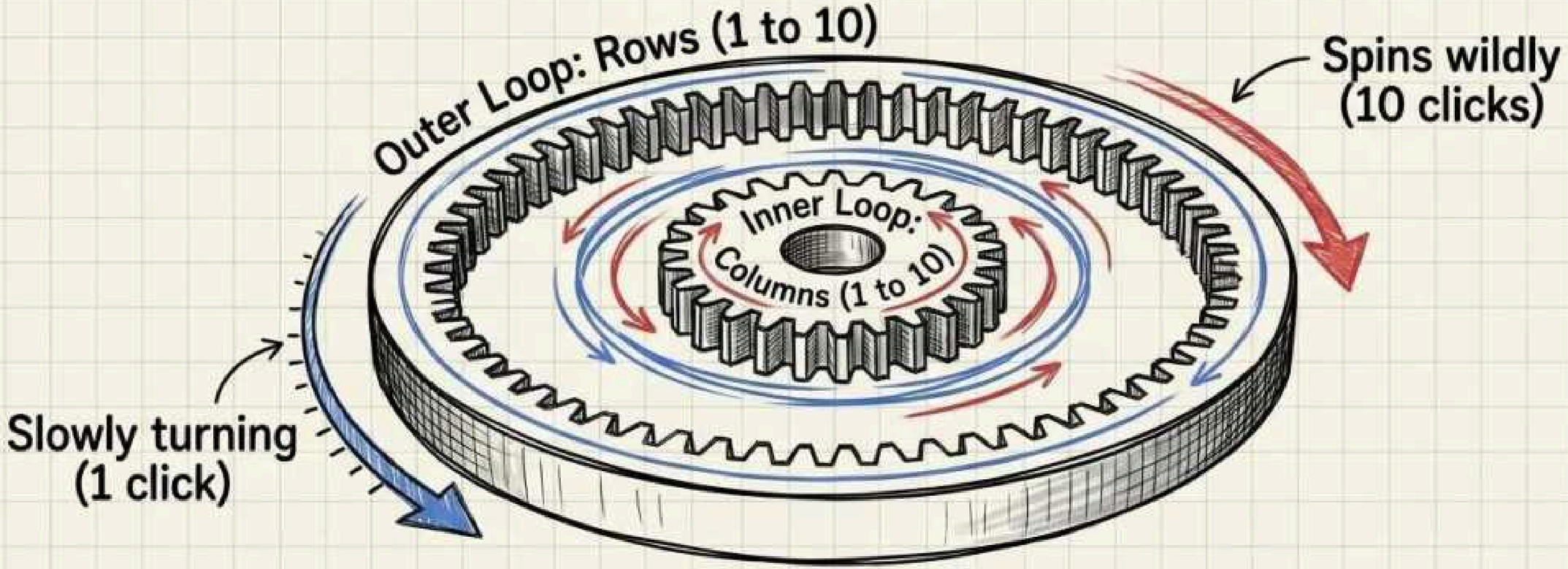
[¶ 4.12]

The Fizz Buzz trap requires testing for both first



[¶ 4.13]

Nested loops multiply the amount of work



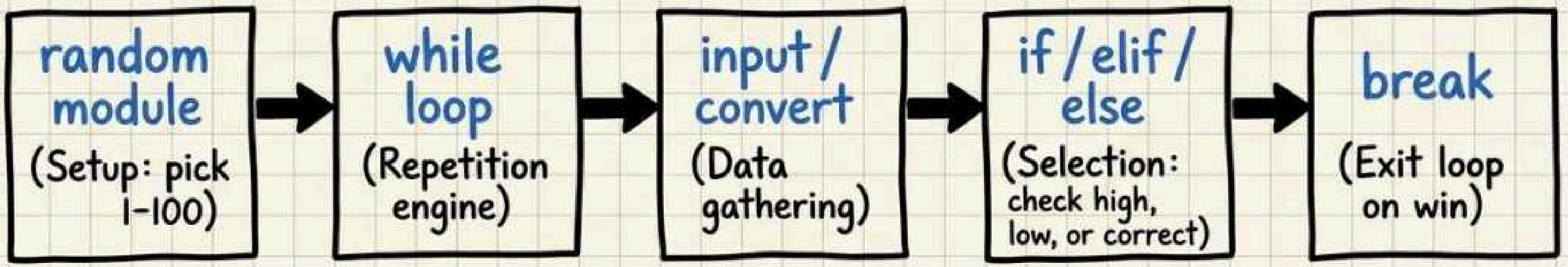
10 outer × 10 inner = 100 total executions

A loop inside another loop runs completely for every single pass of the outer loop.

[¶ 4.14]

Anatomy of a real program: The Guessing Game

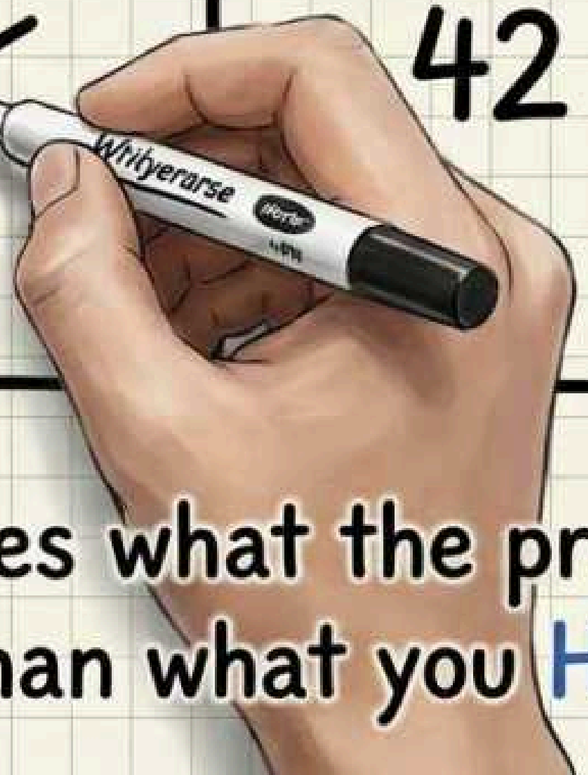
Synthesis of inputs, conditionals, and loops working together.



[¶ 4.15]

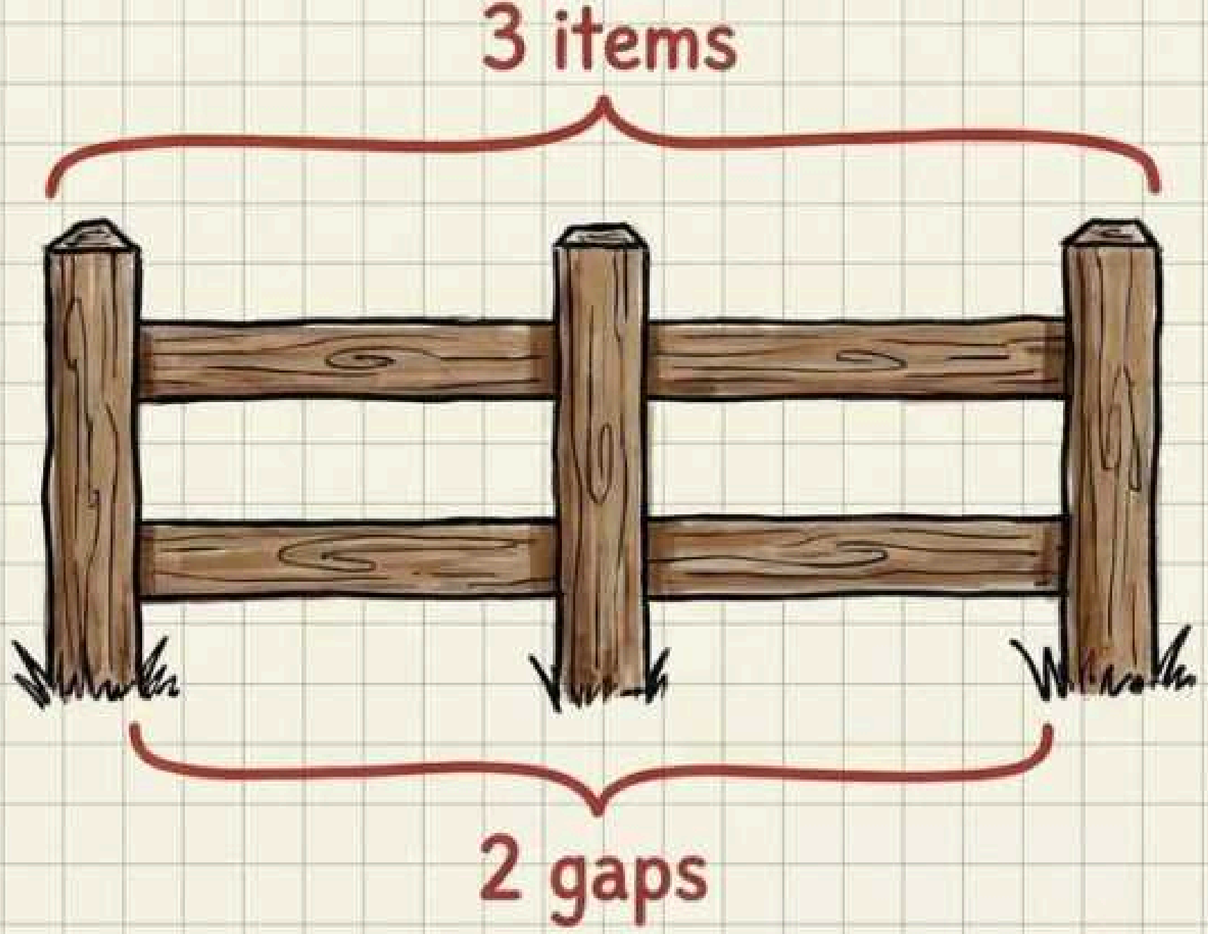
The ultimate debugger is tracing by hand

count	guess	secret
~~3~~ 2	42	77



Tracing exposes what the program ACTUALLY does,
rather than what you HOPED it would do.

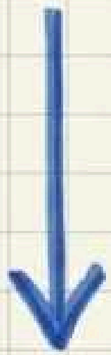
Beware the off-by-one error at the boundaries



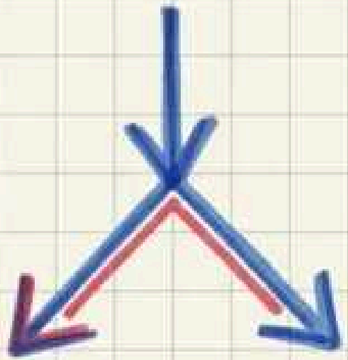
- range(1, 10) stops at 9.
- Confusion arises between counting items vs. gaps.
- Always protect yourself by testing the edges: 0, 1, and many.

[¶ 4.17]

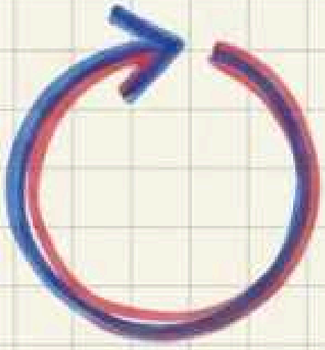
The fundamental theorem of computing (1966)



Sequence



Selection



Repetition

Corrado Böhm and Giuseppe Jacopini proved:
Every algorithm ever written—from simple scripts to modern AI—is built entirely from just these three structures.

The final challenge

Design a program that adds up only the odd numbers from 1 to 99.

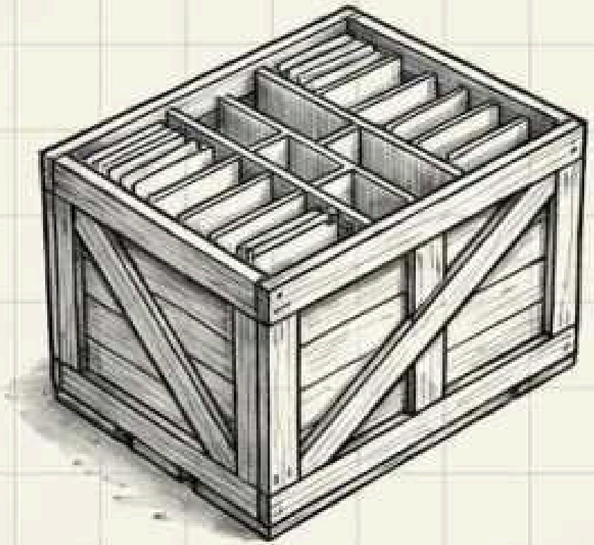
Sum of odds (1 to 99) = 

- What is the start total?
- What is the range?
- What is the if test?

(Hint: The answer is a perfect square). ☺

[¶ 4.19]

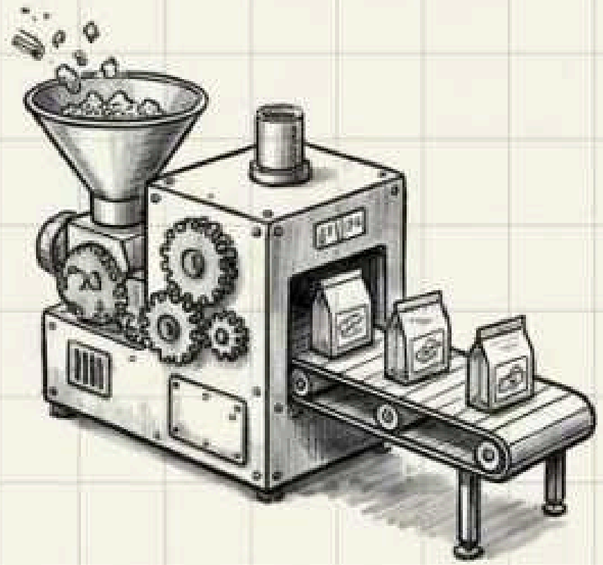
Coming up in Lecture 5



Lists



Dictionaries



Functions

Transitioning from controlling the flow of a program to organizing data and packaging code.