

LECTURE 3.1

Making decisions

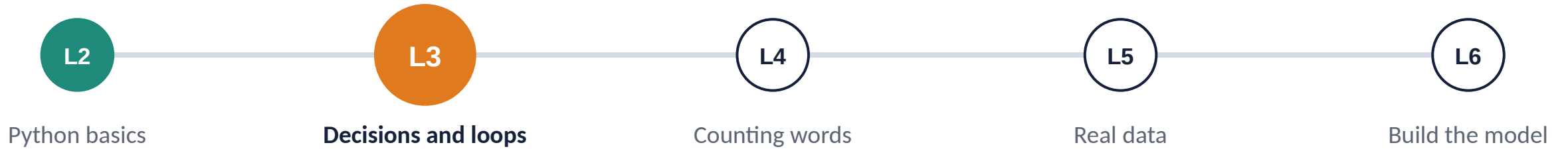
if, elif, else, and match: teaching Python to choose

Today's key line of code

```
if marks >= 50:  
    print("Pass")  
else:  
    print("Try again")
```

Where we are: Lecture 3, part 1 of 3

Part 1 of 3



Today you will

- 1 Ask True/False questions with comparison operators
- 2 Read every part of an if statement
- 3 Choose between paths with else and elif, and see why order matters
- 4 Combine conditions with and, or, not
- 5 Use match-case when one value has many fixed options

Comparison operators give True or False

Every decision starts with a question. Python answers with **True** or **False** (a bool).

You type	Python gives	Meaning
<code>7 == 7</code>	True	equal to (two = signs)
<code>7 != 5</code>	True	not equal to
<code>3 > 8</code>	False	greater than
<code>3 < 8</code>	True	less than
<code>5 >= 5</code>	True	greater than or equal to
<code>4 <= 2</code>	False	less than or equal to
<code>"free" == "Free"</code>	False	text comparison is case-sensitive

= stores, == compares

Predict first

Code

```
x = 5
print(x == 5)
x = 10
print(x == 5)
```



What will this print? Write your prediction as a comment before you run the cell.

Answer: = stores, == compares

Code

```
x = 5
print(x == 5)
x = 10
print(x == 5)
```

Output

```
True
False
```

- `x = 10` stores a new value in x
- `x == 5` only asks a question; it changes nothing
- Mixing up `=` and `==` is the most common beginner bug

Anatomy of an if statement

Code

```
marks = 72
if marks >= 50:
    print("You passed!")
print("Result checked")
```

Output

```
You passed!
Result checked
```

Keyword `if` starts a decision

Condition: a question that is True or False

Colon: the block starts on the next line

4 spaces: this line belongs to the if and runs only when True

No indent: outside the if, runs every time

When the condition is False, the block is skipped

- `35 >= 50` is False, so the indented line is skipped
- The unindented line still runs
- An `if` does not need an `else`
- Try it: change marks to 72 and run again

Code

```
marks = 35
if marks >= 50:
    print("You passed!")
print("Result checked")
```

Output

```
Result checked
```

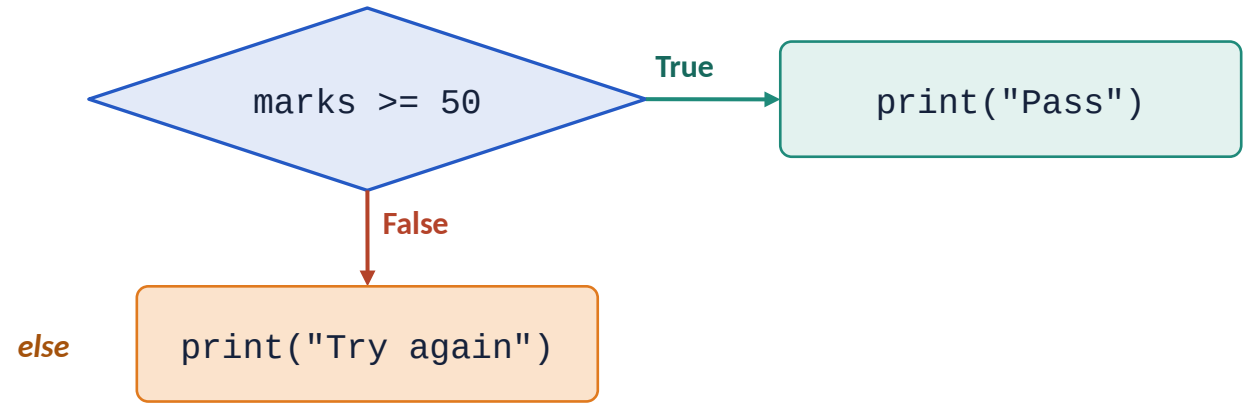
if / else: exactly one of two paths

Code

```
marks = 35
if marks >= 50:
    print("Pass")
else:
    print("Try again")
```

Output

Try again



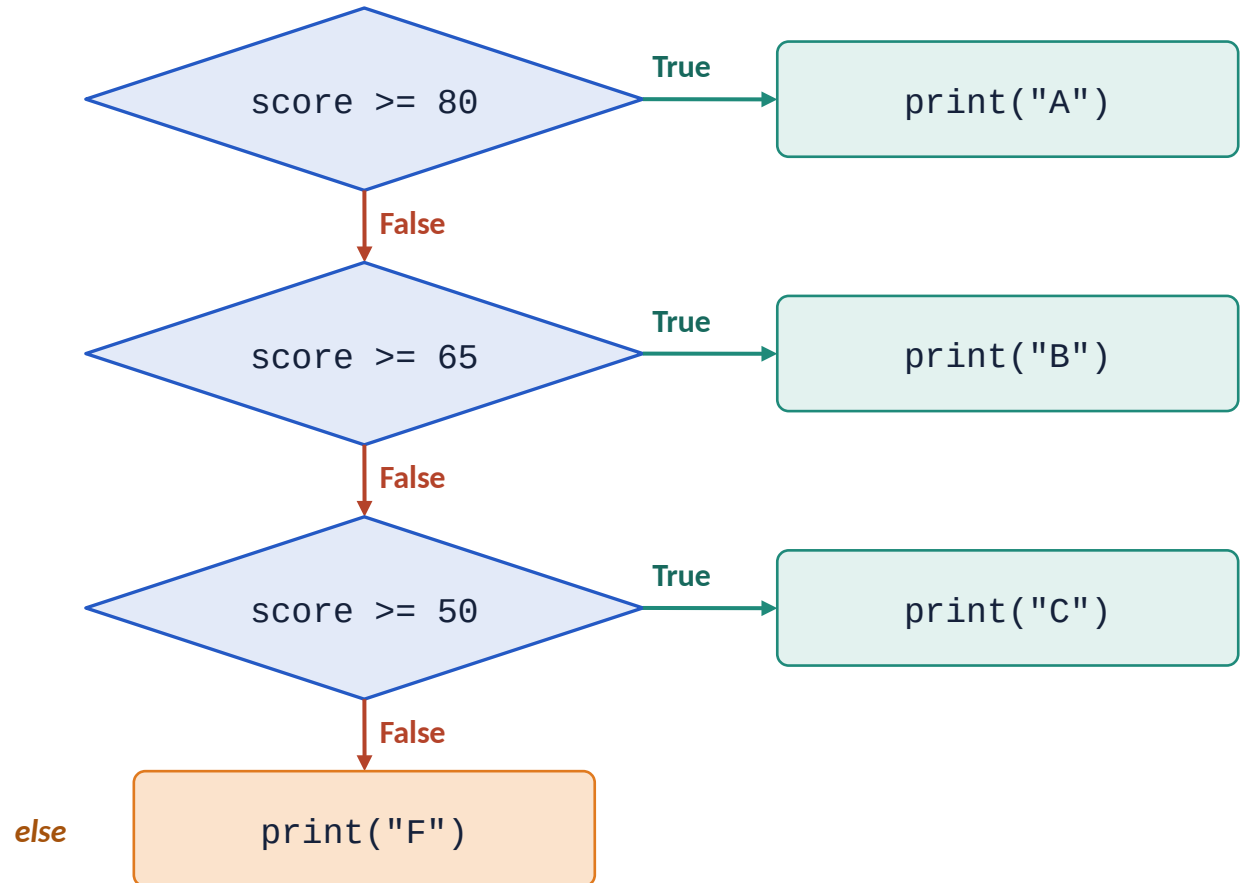
elif: more than two paths

Code

```
score = 72
if score >= 80:
    print("A")
elif score >= 65:
    print("B")
elif score >= 50:
    print("C")
else:
    print("F")
```

Output

B



Tracing the elif chain

Python checks from the top and stops at the first True. Later conditions are not even checked.

Code

```
if score >= 80:
    print("A")
elif score >= 65:
    print("B")
elif score >= 50:
    print("C")
else:
    print("F")
```

score	>= 80	>= 65	>= 50	prints
95	True	not checked	not checked	A
72	False	True	not checked	B
55	False	False	True	C
20	False	False	False	F

Click to reveal the next row. Predict it first!

Watch it run: the elif chain

WATCH FOR

- score = 72 is checked against each condition
- 72 >= 80 is False: skip
- 72 >= 65 is True: print B
- Everything below is skipped

WATCH THE CODE RUN, STEP BY STEP

Which branch of the elif chain runs?

Python checks from the top and stops at the first True condition.
Watch which lines are skipped.

Click the video to play, click again to pause. Silent: talk the class through it.

Does the order of elif matter?

Predict first

Code

```
score = 92
if score >= 50:
    print("C")
elif score >= 65:
    print("B")
elif score >= 80:
    print("A")
```



What will this print? Write your prediction as a comment before you run the cell.

Answer: Does the order of elif matter?

Code

```
score = 92
if score >= 50:
    print("C")
elif score >= 65:
    print("B")
elif score >= 80:
    print("A")
```

Output

C

- `92 >= 50` is True, so Python stops at the first branch
- The A and B branches can never run for any score
- Rule: put the strictest condition first

and, or, not: combining conditions

and needs both. **or** needs at least one. **not** flips True and False.

You type	Python gives	Meaning
True and True	True	both True, so True
True and False	False	one False is enough to make and False
False or True	True	one True is enough to make or True
False or False	False	nothing True, so False
not True	False	not flips the value
5 > 3 and "a" in "cat"	True	real conditions combine the same way

Combining conditions in an if

- Both parts must be True for **and**
- Each side of **and** is a full condition
- Write **age >= 18 and age <= 60**,
not **age >= 18 and <= 60**
- Try it: set has_id to False

Code

```
age = 20
has_id = True
if age >= 18 and has_id:
    print("Can enter")
else:
    print("Cannot enter")
```

Output

```
Can enter
```

An if inside an if (nesting)

- Each level indents 4 more spaces
- The inner question is asked only if the outer one is True
- Use nesting when the second question only makes sense after the first
- Otherwise **and** keeps the code flatter

Code

```
marks = 72
attendance = 60
if marks >= 50:
    if attendance >= 75:
        print("Pass")
    else:
        print("Pass marks, low attendance")
else:
    print("Fail")
```

Output

```
Pass marks, low attendance
```

Decisions about text: the first spam rule

- `in` checks whether text appears inside other text
- `.lower()` first, so FREE and free both match
- `or`: any one suspicious word is enough
- Lecture 3.3 turns this into a filter for a whole inbox

Code

```
message = "Claim your FREE prize"
text = message.lower()
if "free" in text or "prize" in text:
    print("Suspicious")
else:
    print("Looks normal")
```

Output

```
Suspicious
```

match-case: one value, many fixed options

- **match** compares one value with several patterns
- **|** means or inside a case
- **case _:** catches everything else, like else
- **number[:3]** = first 3 characters (Lecture 3.2)
- Needs Python 3.10+; Colab has it

Code

```
number = "017111111111"
match number[:3]:
    case "017" | "013":
        print("Grameenphone")
    case "019" | "014":
        print("Banglalink")
    case "018":
        print("Robi")
    case _:
        print("Unknown operator")
```

Output

```
Grameenphone
```

Common mistakes with decisions

Read the last line of an error first. It names the problem.

What you typed	What Python says	What it means
<pre>if marks >= 50 print("ok")</pre>	<code>SyntaxError: expected ':'</code>	Missing colon at the end of the if line.
<pre>if marks = 50: print("ok")</pre>	<code>SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?</code>	One = stores. To compare, use ==.
<pre>if marks >= 50: print("ok")</pre>	<code>IndentationError: expected an indented block after 'if' statement on line 1</code>	The block must be indented under the if.
<pre>if "5" > 3: print("ok")</pre>	<code>TypeError: '>' not supported between instances of 'str' and 'int'</code>	Text and numbers cannot be compared. Convert with <code>int("5")</code> .

In-class exercises

Work in pairs. 15 minutes. $n \% 2$ is the remainder after dividing n by 2.

1 Even or odd

Given n , print Even or Odd. Test: 7 gives Odd, 10 gives Even.

2 Largest of three

Given a , b , c , print the largest using `if/elif/else`, without `max()`. Test: 4, 9, 2 gives 9.

3 Leap year

Leap if divisible by 4 and not by 100, or divisible by 400.
Test: 2024 Leap, 1900 Not leap, 2000 Leap.

4 Operator check

Extend the match example with Teletalk (015) and Airtel (016). Print INV if the number is not exactly 11 characters.

Solutions

Show these after students have tried the exercises.

Solutions 1 and 2

Test every branch: try values that should give each possible answer.

Code

```
n = 7
if n % 2 == 0:
    print("Even")
else:
    print("Odd")

a = 4
b = 9
c = 2
if a >= b and a >= c:
    print(a)
elif b >= c:
    print(b)
else:
    print(c)
```

Output

```
Odd
9
```

Solution 3: leap year

The brackets group the first two conditions. Test the tricky years: 1900 and 2000.

Code

```
year = 1900
if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
    print("Leap")
else:
    print("Not leap")
```

Output

Not leap

Solution 4: operator check

The **else** matters: without it, a wrong-length number would print INV and then also an operator name.

Code

```
number = "0151234567"
if len(number) != 11:
    print("INV")
else:
    match number[:3]:
        case "017" | "013":
            print("Grameenphone")
        case "019" | "014":
            print("Banglalink")
        case "015":
            print("Teletalk")
        case "016":
            print("Airtel")
        case "018":
            print("Robi")
        case _:
            print("INV")
```

Output

INV

Interactive lab: Decisions Lab

In the lab you can

1 Grade machine

Move the score, follow the path, then break it with the wrong elif order.

2 and · or · not

Flip p and q, build truth tables, and test the entry rule.

3 Spam rule tester

Pick words, test the rule on 12 messages, count misses and false alarms.

4 match-case

Type a phone number and watch which case matches.

LECTURE 3.1 · INTERACTIVE LAB

Decisions Lab

See which branch runs, build truth tables, and write your own spam rules

Introduction to Data Science and Machine Learning
Md Mahbub E Noor · University of Barishal
[All Labs](#)

1 Grade machine 2 and · or · not 3 Spam rule tester 4 match-case

Grade machine: which branch runs, and why order matters

Move the score. Python checks the conditions from the top and stops at the first **True**. Then switch to the wrong order and watch whole grades disappear.

score = 72

Order: Strictest first (correct) ▾

A ≥ 80 B ≥ 65 C ≥ 50

```
1 score = 72
2 if score >= 80:
3     print("A")
4 elif score >= 65:
5     print("B")
6 elif score >= 50:
7     print("C")
8 else:
9     print("F")
```

72 >= 80 is False

72 >= 65 is True

runs

not checked

Python checked 2 conditions and printed B. The remaining conditions were never looked at.

THE PATH PYTHON TAKES

```
graph TD
    D1{score >= 80} -- True --> A[print("A")]
    D1 -- False --> D2{score >= 65}
    D2 -- True --> B[print("B")]
    D2 -- False --> D3{score >= 50}
    D3 -- True --> C[print("C")]
    D3 -- False --> F[print("F")]
```

EVERY SCORE FROM 0 TO 100 WITH THIS CODE

Open the lab ▶

Works offline in any browser
Labs_open_in_browser/
L3.1_Decisions_Lab.html

Today's toolkit

== != > < >= <=	Compare, gives True/False
if condition:	Run the block only if True
else:	Run when nothing above was True
elif condition:	Another check, only if earlier failed
and / or / not	Combine or flip conditions

"x" in text	Does text contain x?
match value:	Compare one value with many options
case "a" "b":	Matches a or b
case _:	Anything else

Homework 3.1

Due before Lecture 3.2 • [date]

1 Predict, then run

Write predictions as comments first:
`print(10 != 10)`, `print(not (3 > 5))`, `print("a"`
in "Data" and `2 < 1)`.

2 Weather advice

Store a temperature. Print Cold (below 15), Pleasant (15 to 27) or Hot (above 27). Test 14, 15, 27, 28.

3 Ticket price

Price is 50 for age under 12 or students, 100 otherwise. Store age and `is_student` and print the price.

4 Spam level

Store a message. Print High if it contains both "win" and "cash", Medium if either, otherwise Low. Ignore capitals.

5 Find the bug

The "Does the order matter" slide prints C for 92. Fix the code and explain the bug in one sentence.

Submit on [LMS / Google Classroom]: your notebook file `HW3_1_<your ID>.ipynb`, or a Colab link set to "Anyone with the link can view".

Recap: Lecture 3.1 in 90 seconds

RECAP

Lecture 3.1 in 90 seconds

A short recap of today's lecture. Pause any time.

Click to play or pause. Watch it again at home to revise.