

Logic and decisions

Big picture

- **Data (Boolean)**

- instead of storing a number or a name, it stores an answer to a question
- is this true, or false?
- Comparisons (`<`, `>`, `==`) are how you ask those questions.

- **Decisions**

- let your program 'choose' what to do next, instead of always doing the same thing in the same order.
- `if`, `elif`, and `else` use Boolean answers to pick a *path*.

- **Boolean data + decisions**

- without these things, your program can only ever do one fixed sequence of steps — it can't respond to anything.

Overview

- **Boolean logic**

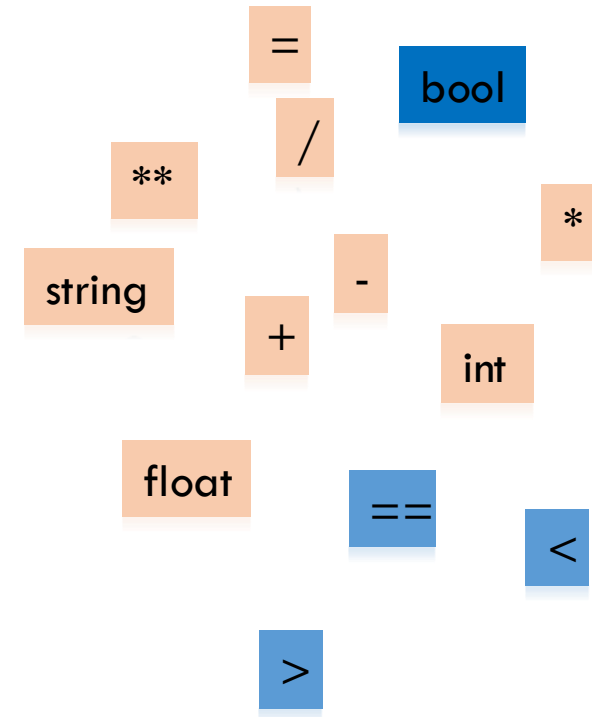
- boolean functions, 'gates', circuits
- high-level behavior from simple units

- **More nuts bolt, glue**

`True, False, <, >, ==, !=, and, not,
or`

- **Paths through a program**

- Booleans as conditions
- Program branches: `if, else, elif`



Boolean variables

- Boolean key words
- Boolean comparisons
- Simple 'decisions' / 'branching'

Binary state

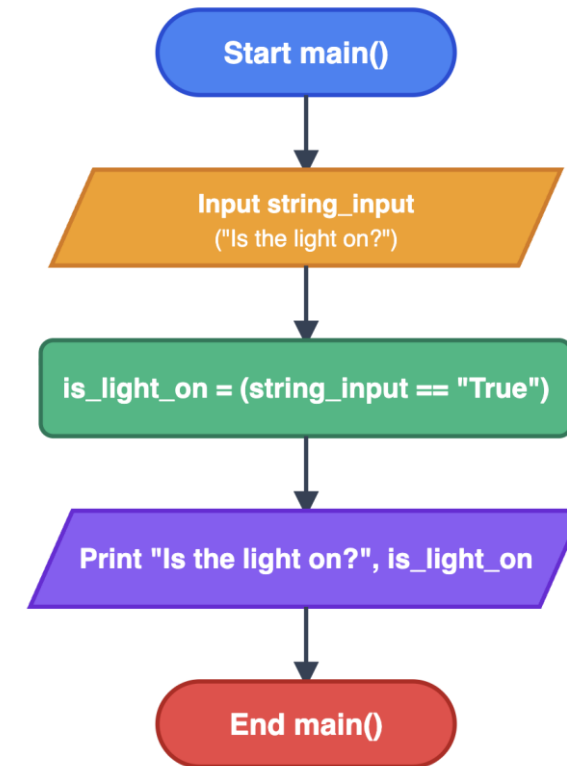


Binary state



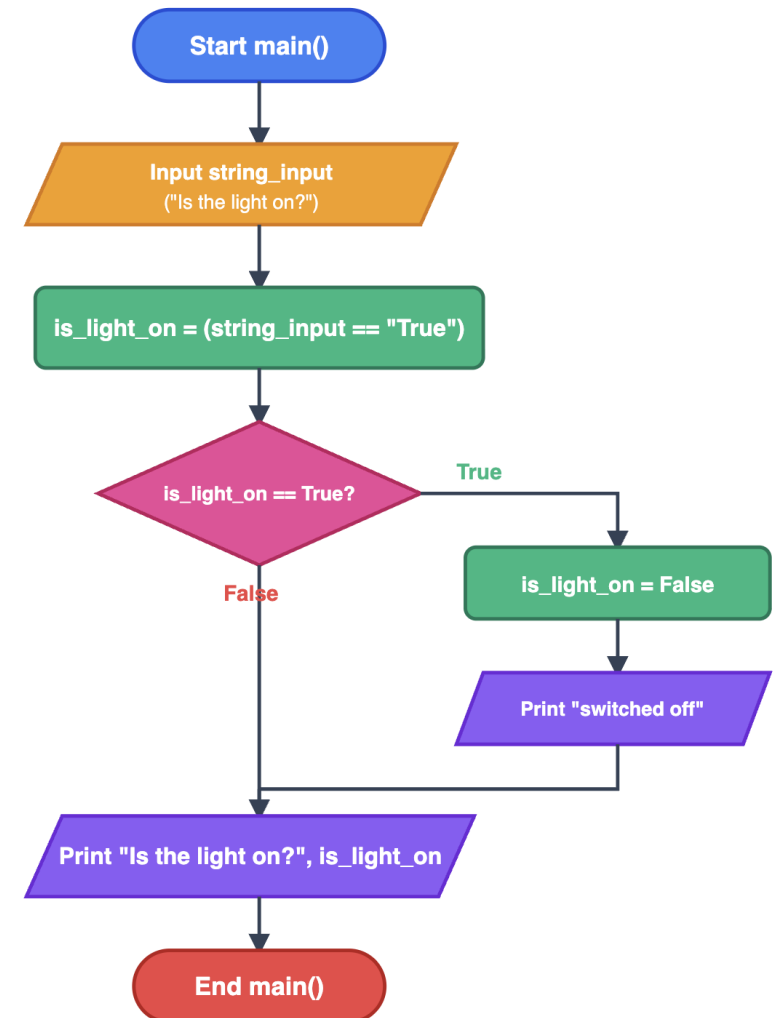
Binary state

```
1 def main():
2     ... string_input = input("Is the light on (Enter True or False)")
3     ... is_light_on = string_input == "True"
4
5     ... print("Is the light on?", is_light_on)
6
7 main()
```



Decision

```
1  def main():
2      ... string_input = input("Is the light on (Enter True or False)")
3      ... is_light_on = string_input == "True"
4
5      ... if is_light_on == True:
6          ...     is_light_on = False
7          ...     print("The light has been switched off")
8
9      ... print("Is the light on?", is_light_on)
10
11  main()
```



Decisions

```
1 def main():
2     ... string_input = input("Is the light on (Enter True or False)")
3     ... is_light_on = string_input == "True"
4
5     ... if is_light_on == True:
6     ...     is_light_on = False
7     ...     print("The light has been switched to off")
8     ... else:
9     ...     is_light_on = True
10    ...     print("The light has been switched to on")
11
12    ... print("Is the light on?", is_light_on)
13
14    main()
```



Boolean conditions

- Comparison logic
- More Boolean states in a condition, not just one

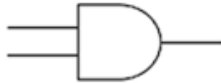
Comparison logic

Comparison Operators

Operator	Meaning	Example	Result
<code>==</code>	equal to	<code>5 == 5</code>	True
<code>!=</code>	not equal to	<code>5 != 3</code>	True
<code><</code>	less than	<code>3 < 5</code>	True
<code>></code>	greater than	<code>7 > 10</code>	False
<code><=</code>	less than or equal to	<code>5 <= 5</code>	True
<code>>=</code>	greater than or equal to	<code>10 >= 7</code>	True

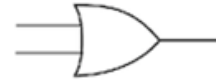
Keywords based on gates (subset)

AND gate



INPUT		OUTPUT
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

OR gate



INPUT		OUTPUT
A	B	
0	0	0
0	1	1
1	0	1
1	1	1

NOT gate



INPUT	OUTPUT
0	1
1	0

Boolean Operators

Operator	Meaning	Example	Result
and	True if <i>both</i> are True	(5 > 3) and (2 < 4)	True
or	True if <i>at least one</i> is True	(5 > 3) or (2 > 4)	True
not	Flips True ↔ False	not (5 > 3)	False

Blocks: 'if' / 'elif'

```
1 def determine_grade(percentage):
2     ... result = "E"
3     ... if percentage >= 90:
4         ... result = "A"
5     ... if 80 <= percentage < 90:
6         ... result = "B"
7     ... if 70 <= percentage < 80:
8         ... result = "C"
9     ... if 60 <= percentage < 70:
10        ... result = "D"
11    ... if percentage < 60:
12        ... result = "E"
13    ... return result
14
15 def main():
16     ... percent = input("What % did you get? Please enter a number\n> ")
17     ... grade = determine_grade(float(percent))
18     ... print(f"Your grade is a {grade}")
19
20 main()
21
```

'if' block

```
1 def determine_grade(percentage):
2     ... if percentage >= 90:
3         ... return "A"
4     ... elif percentage >= 80:
5         ... return "B"
6     ... elif percentage >= 70:
7         ... return "C"
8     ... elif percentage >= 60:
9         ... return "D"
10    ... elif percentage >= 50:
11        ... return "D"
12    ... else:
13        ... return "Fail"
14
15 def main():
16     ... percent = input("What % did you get? Please enter a number\n> ")
17     ... percent = float(percent)
18     ... grade = determine_grade(percent)
19     ... print(f"Your grade is a {grade}")
20
21 main()
```

'elif' block

Conclusions: Python & decisions

- **Booleans hold an answer, not a value**
 - `True` / `False` is a data type in its own right, alongside `int`, `float`, and `string`
 - comparisons (`<`, `>`, `==`, `!=`) are what turn a question into that Boolean answer
- **Decisions give a program branches**
 - `if`, `elif`, and `else` read a Boolean and pick which block of code runs
 - ordering matters: `elif` only gets checked once the branches above it fail, which is what makes it more efficient than a stack of separate ifs
- **Together, they let code respond**
 - without Boolean data and decisions, a program can only run one fixed sequence of steps
 - with them, it can choose its path based on the data it's given

Thanks!