DS2000 – Programming with Data

Control Flow

John Rachlin, PhD
Northeastern University

Northeastern University

Algorithms

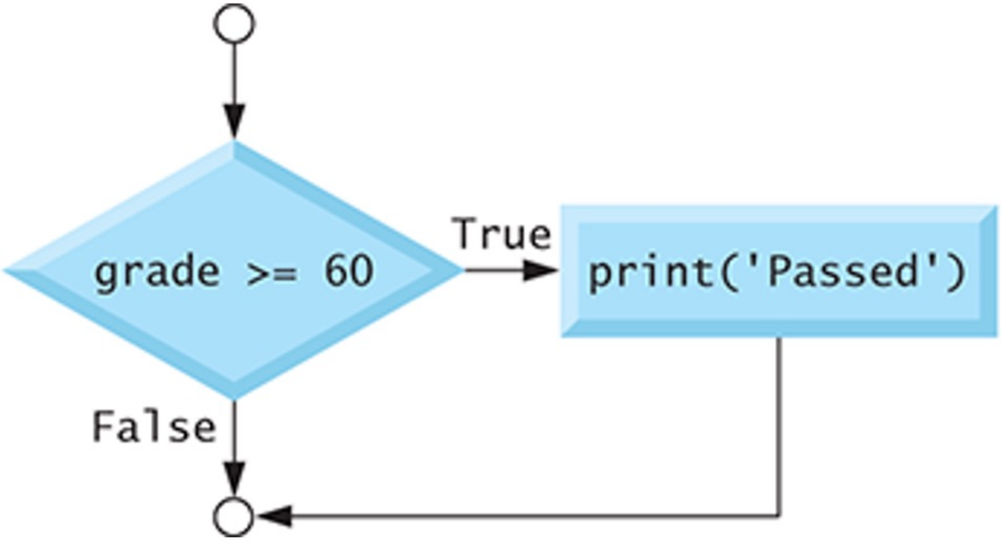
An *algorithm* is a finite sequence of instructions, typically implemented in some programming language such as *Python,* used to carry out some computational task. Computer Scientists try to design algorithms that are both correct and efficient.

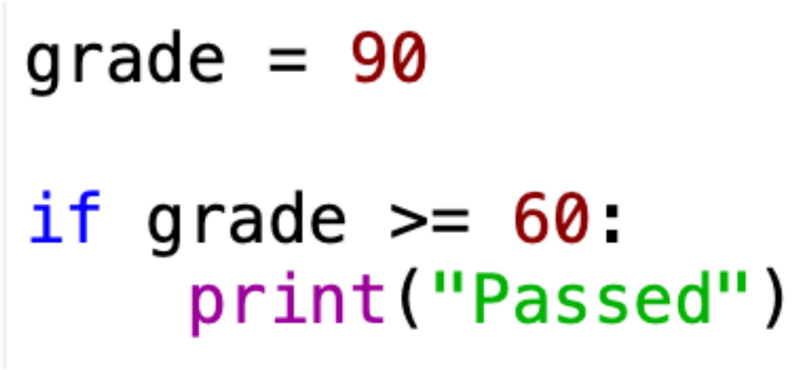
Correct: The algorithm carries out the calculation correctly often according to a well-defined specification.

Efficient: The program runs quickly and without using a lot of processing and memory resources.

New Scientist
WEEKLY 19 June 2021
SECRETS OF ANCIENT GLACIERS
CAN YOU CATCH COVID-19 TWICE?
MYSTERY SPACE SIGNAL EXPLOSION
YOUR DOG MAY NOT LIKE YOU AS MUCH AS YOU THINK IT DOES
MON
18°C
JPG
THE ALGORITHMS THAT RUN YOUR LIFE
What they do, and how they shape your every decision
NOISY THINKING
Why human judgement varies so wildly
PLUS A SELFIE ON MARS / ACCURATELY COUNTING CALORIES / QUANTUM MICROSCOPE / HONEYBEES PRODUCE PERFECT CLONES

if: A simple branch



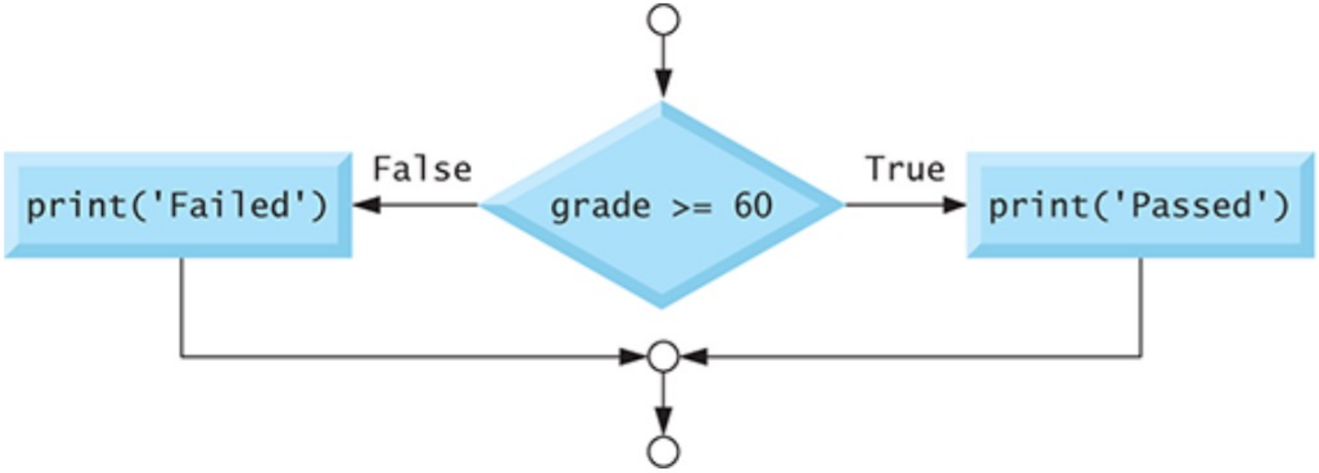


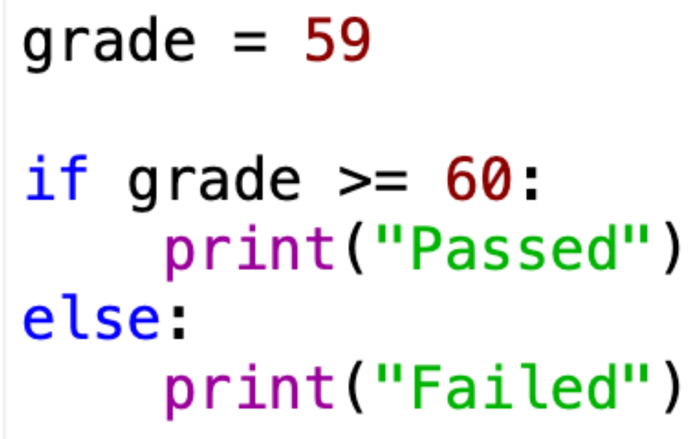
```
grade = 90

if grade >= 60:
    print("Passed")
```

If the test is **True**, branch.
Otherwise continue on with the rest of the program.

if…else





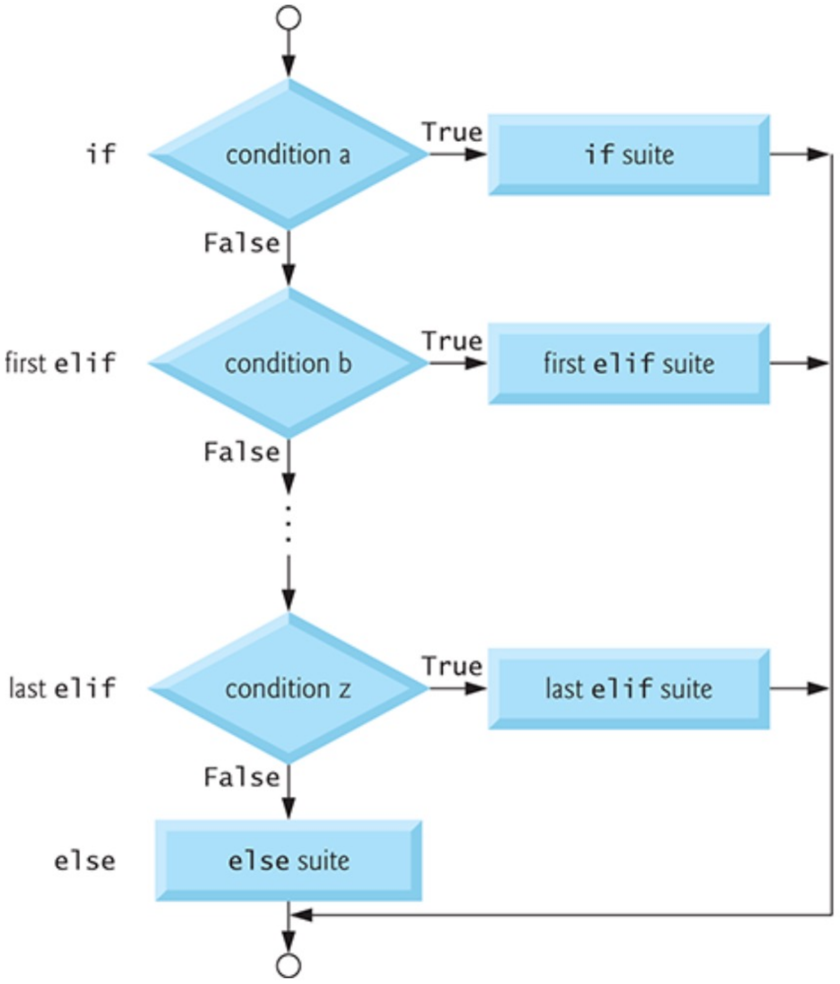
```
grade = 59

if grade >= 60:
    print("Passed")
else:
    print("Failed")
```

If the test succeeds, branch one way.
Otherwise (else) branch a different way.

Exactly ONE of the branches will execute because the test is either **True** or **False**.

if..elif…else: Multiple possible branches



Run a series of tests.

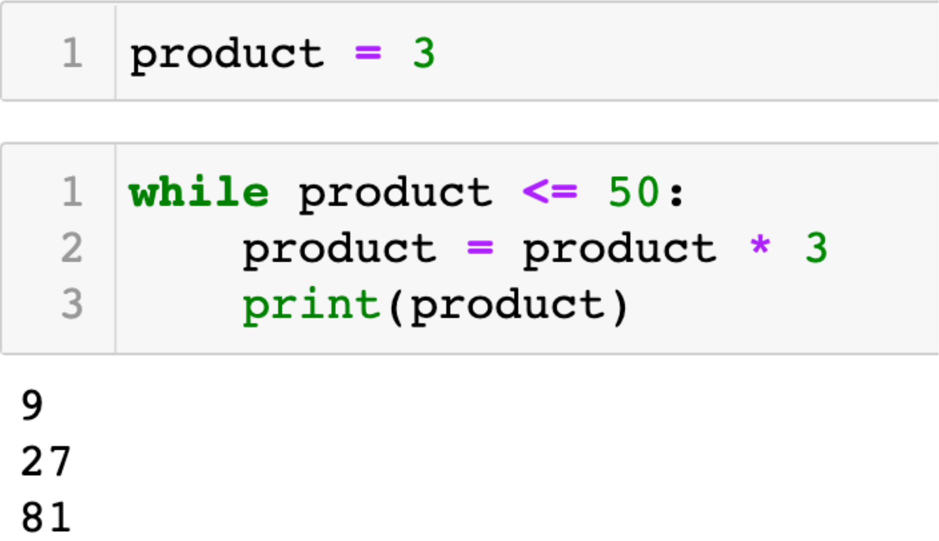
As soon as one test succeeds,
branch.

If no test succeeds run the
else branch (if there is one).

```
grade = 86

if grade >= 90:
    print("A")
elif grade >= 80:
    print("B")
elif grade >=70:
    print("C")
else:
    print("F")
```

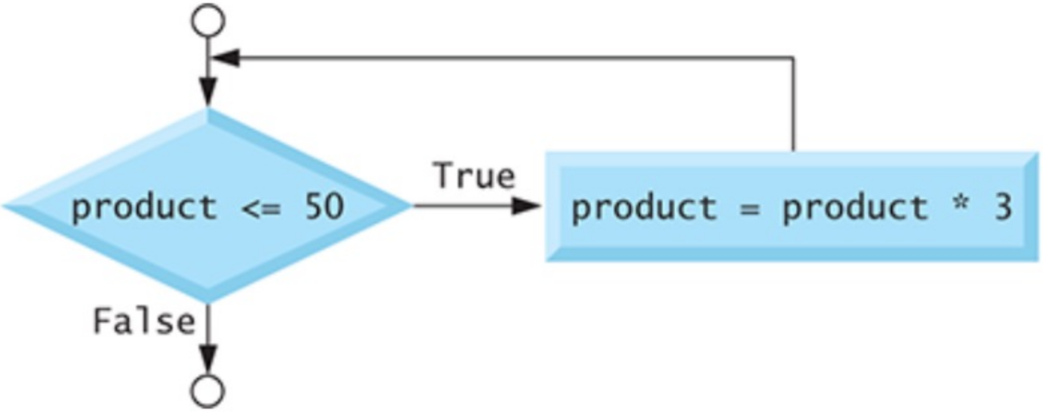
while



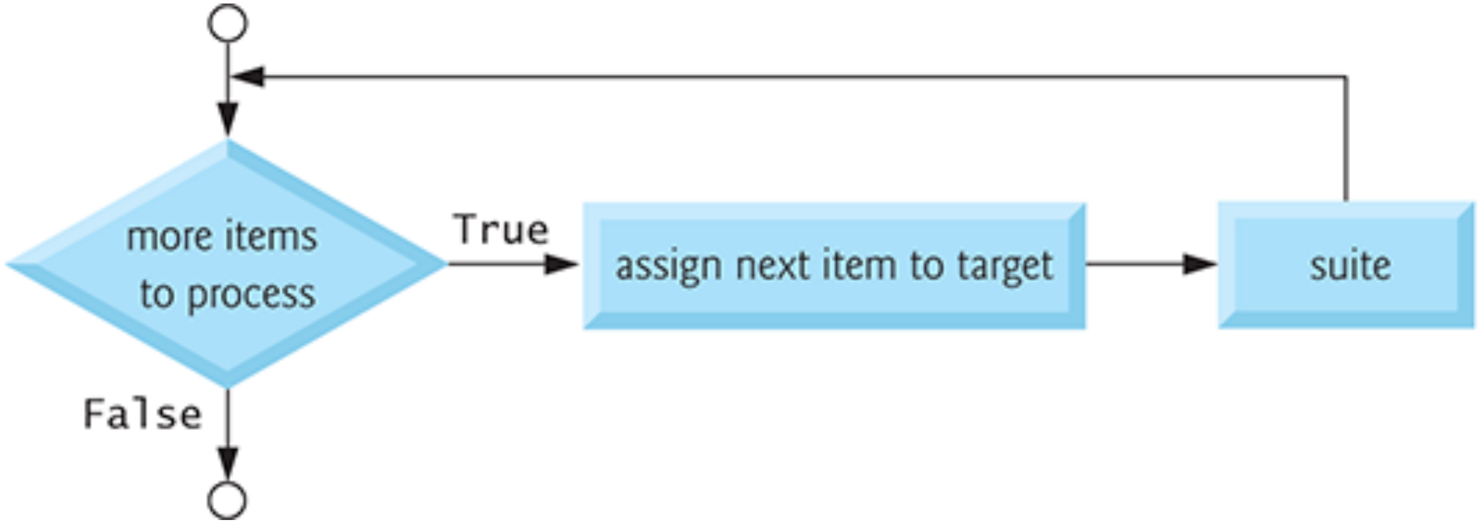
```python
product = 3
```

```python
while product <= 50:
    product = product * 3
    print(product)
```

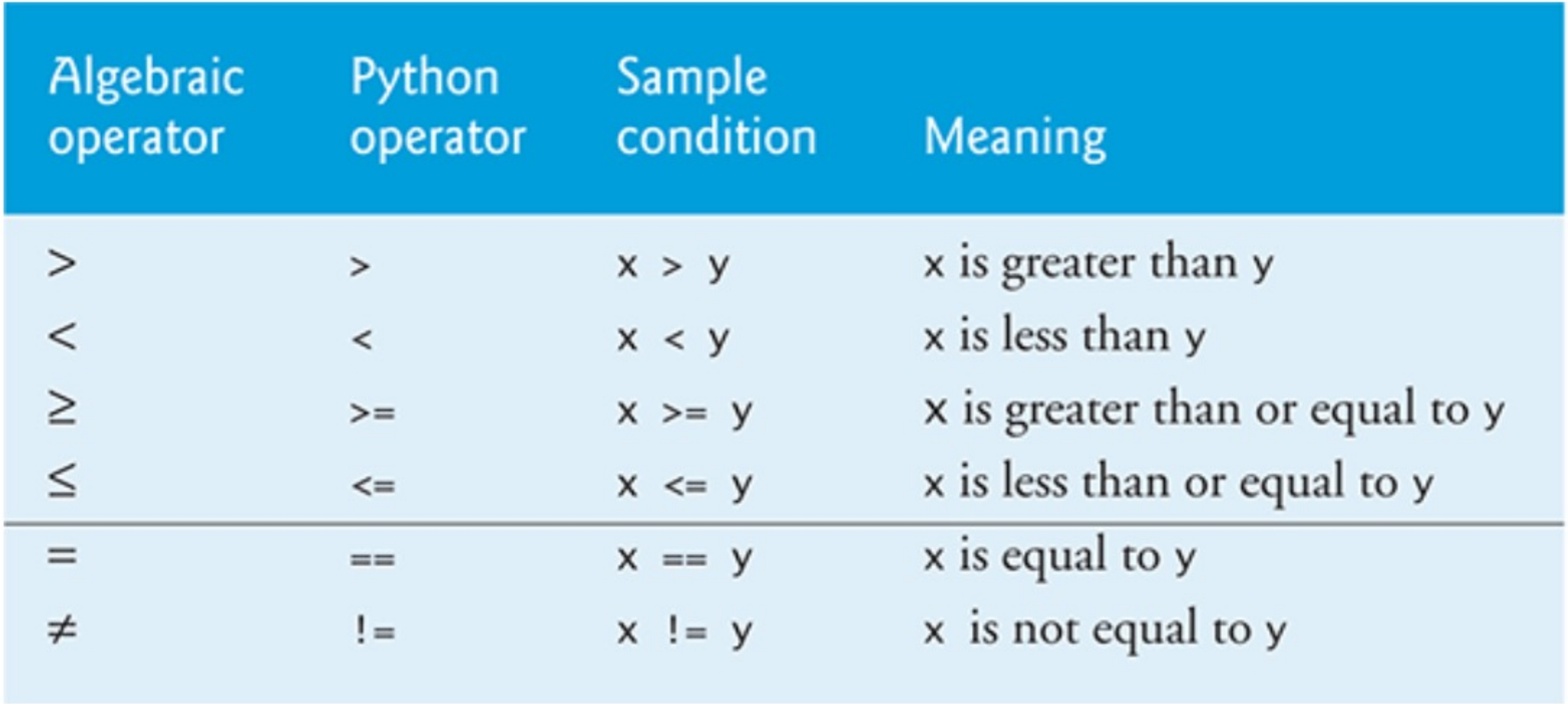
```
9
27
81
```



for

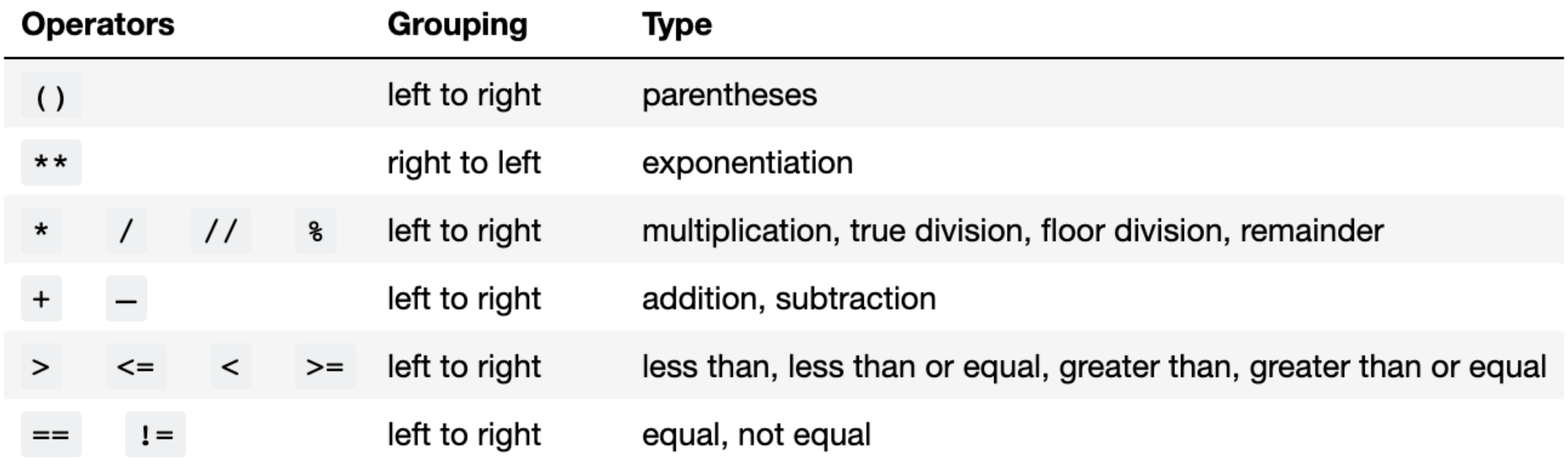

more items to process
True
assign next item to target
suite
False

Comparison operators



Algebraic operator	Python operator	Sample condition	Meaning
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

Operator precedence



Operators	Grouping	Type
( )	left to right	parentheses
**	right to left	exponentiation
* / // %	left to right	multiplication, true division, floor division, remainder
+ –	left to right	addition, subtraction
> <= < >=	left to right	less than, less than or equal, greater than, greater than or equal
== !=	left to right	equal, not equal

2 ** 3 / 4 - 1

8 / 4 - 1

2.0 - 1

1.0