Operators

6.5 · Problem solving with programming · revision notes

The operators Python uses to calculate, compare and combine conditions: arithmetic operators (including integer division, modulus and exponentiation), relational operators (such as == and !=) and logical operators (and, or, not). You need to work out the value of expressions and choose the right operator in your own code.

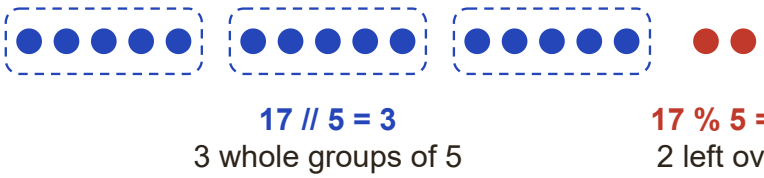
Key facts

- + addition, - subtraction, * multiplication, / division (always gives a real)
- // integer division: 17 // 5 = 3
- % modulus (remainder): 17 % 5 = 2
- ** exponentiation: 2 ** 3 = 8
- == equal to, != not equal to, < less than, > greater than, <= less than or equal to, >= greater than or equal to
- and: both True. or: at least one True. not: reverses True and False
- Order: brackets, **, then * / // %, then + -
- Even test: n % 2 == 0
- = assigns a value; == compares values

Notes

Arithmetic operators

- `+` addition, `-` subtraction, `*` multiplication, `/` division.
- `/` always gives a real (float): `7 / 2` is 3.5 and `8 / 2` is 4.0.
- `//` **integer division**: how many whole times one number goes into another. `17 // 5` is 3.
- `%` **modulus**: the remainder after integer division. `17 % 5` is 2.



17 shared into groups of 5: `//` counts the whole groups, `%` gives what is left over.

- `**` **exponentiation** (to the power of): `2 ** 5` is 32.
- Order: brackets, then `**`, then `*`, `/`, `//` and `%` (left to right), then `+` and `-`. So `3 + 4 * 2` is 11 but `(3 + 4) * 2` is 14.

Using // and %

- Even or odd: `n % 2 == 0` is True when `n` is even.
- Divisible: `n % 7 == 0` is True when `n` divides exactly by 7.
- Units: 200 minutes is `200 // 60` = 3 hours and `200 % 60` = 20 minutes.
- Digits: `n % 10` is the last digit of a whole number and `n // 10` removes it. For 472 these give 2 and 47. Grade 7+ only

Relational operators

- `==` equal to, `!=` not equal to, `<` less than, `>` greater than, `<=` less than or equal to, `>=` greater than or equal to.
- Every comparison gives a Boolean result: `5 >= 5` is True and `5 > 5` is False.
- `=` assigns a value; `==` compares two values. `if score = 10:` is a syntax error.
- Strings can be compared too, but it is case-sensitive: `"Yes" == "yes"` is False.

Logical operators

- `and`: True only if both conditions are True. `or`: True if at least one condition is True. `not`: reverses a Boolean, so `not True` is False.
- Inside a range: `mark >= 0 and mark <= 100`. Outside a range: `mark < 0 or mark > 100`.
- Write every comparison in full: `choice == "A" or choice == "B"`, not `choice == "A" or "B"`. Grade 7+ only
- Python applies `not` first, then `and`, then `or`. Use brackets to make the order you want clear, e.g. `(day == 6 or day == 7) and not raining`. Grade 8+ only

How to answer each type of question

State the value of arithmetic expressions 1 mark each **Grade 5**

1. Put the values of the variables into the expression.
2. Work out brackets first, then **, then *, /, // and %, then + and −.
3. Remember that / always gives a real, so write 4.0, not 4.

Example.

```
a = 23
b = 4
```

State the value of each expression.

(a) `a // b` (1)

(b) `a % b` (1)

(c) `a / 2` (1)

(d) `b ** 2 + 1` (1)

(e) `(a - 3) // b` (1)

Model answer.

(a) 5 (1)

(b) 3 (1)

(c) 11.5 (1)

(d) 17 (1)

(e) 5, because 20 // 4 = 5 (1)

State whether conditions are True or False

1 mark each | Grade 6

1. Work out each comparison on its own and write T or F above it.
2. Then apply not, then and, then or (brackets first).
3. Remember that comparing strings is case-sensitive.

Example.

```
x = 8
y = 3
name = "Ali"
```

State whether each condition is True or False.

(a) `x != y` (1)

(b) `x > 5 and y > 5` (1)

(c) `x > 5 or y > 5` (1)

(d) `not (x % y == 2)` (1)

(e) `name == "ali"` (1)

Model answer.

(a) True	(1)
(b) False	(1)
(c) True	(1)
(d) False, because 8 % 3 is 2, so the bracket is True	(1)
(e) False, because the comparison is case-sensitive	(1)

Write code using // and %

3 marks | Grade 6

1. Use // to find how many whole units fit.
2. Use % to find what is left over.
3. Output both parts in the form the question asks for.

Example. The integer variable `seconds` holds a length of time, e.g. 500.
Write code that outputs the time in minutes and seconds, e.g. `8 minutes 20 seconds`. (3)

Model answer.

```
minutes = seconds // 60
left = seconds % 60
print(minutes, "minutes", left, "seconds")
```

Uses `// 60` to find the whole minutes (1); uses `% 60` to find the seconds left over (1); outputs both values with the words minutes and seconds (1).

Write a condition using logical operators

3 to 4 marks | Grade 7

1. Write each part of the rule as a full comparison.
2. Join the parts with and (all must be true) or or (any can be true).
3. Add brackets so that or is worked out before and when the rule needs it.

Example. A student may enter a competition if they are in year 10 or year 11 and they have not entered before. The year is stored in the integer `year`, and `entered` is `True` if the student has entered before.
Write an `if` statement that outputs `You may enter` or `You cannot enter`. (4)

Model answer.

```
if (year == 10 or year == 11) and not entered:
    print("You may enter")
else:
    print("You cannot enter")
```

Checks the year is 10 or 11, with both comparisons written in full (1); checks the student has not entered, e.g. `not entered` or `entered == False` (1); brackets and `and` used so the year test is worked out first (1); correct message in each case using `else` (1).
Without the brackets, a year 10 student who has entered before would be allowed in.

Don’t lose marks

- Writing that 17 / 5 is 3. It is 3.4; use // for whole-number division.
- Using = instead of == in a condition.
- Writing choice == "A" or "B": this is always True, because a non-empty string on its own counts as True.
- Using and where you need or: mark < 0 and mark > 100 can never be True.
- Forgetting that comparing strings is case-sensitive.
- Ignoring the order of operations: 2 + 3 ** 2 is 11, not 25.

More tips

Memory tricks

- // asks 'how many whole times?'; % asks 'how much is left over?'
- Check: for positive whole numbers, (a // b) * b + a % b always equals a. For 17 and 5: 3 × 5 + 2 = 17.
- and is fussy (needs both); or is easy-going (needs just one).
- Inside a range, join with and; outside a range, join with or.
- One equals sign sets a value; two equals signs ask a question.

Exam technique

- When stating output, remember that / always gives a real: write 4.0, not 4.
- For True or False questions, work out each comparison separately before applying not, and, or.
- Name operators with the specification's terms: modulus, integer division, exponentiation.
- Test every condition you write with a value on each side of the boundary.

What each grade needs

What you need to be able to do, from the first marks up to the top grade.

Grade 3 — **Use +, −, * and / correctly**
Know that / always gives a real result, e.g. 8 / 2 gives 4.0.

Grade 4 — **Use the six relational operators**
Compare values with ==, !=, <, >, <= and >= to get True or False.

Grade 5 — **Use integer division and modulus**
17 // 5 gives 3 (the whole number part) and 17 % 5 gives 2 (the remainder).

Grade 5 — **Combine conditions with and, or, not**
E.g. age >= 13 and age <= 19 is True only for ages 13 to 19.

Grade 6 — **Apply the order of operations**
Brackets first, then exponentiation, then *, /, // and %, then + and −.

Grade 7 — **Use modulus to test divisibility**
n % 2 == 0 tests for an even number; n % 10 gives the last digit.

Grade 8 — **Evaluate complex Boolean expressions**
Know that Python applies not first, then and, then or, and use brackets to control the order.

Free notes, Full packs, practice papers and mock exams at chhetriacademy.com. Written for this site; not taken from past papers. Stuck? 1-to-1 tutoring with a free 30-minute trial lesson.