1. What does ‘print(”3” + ”4”)‘ show?

   _____

2. What does ‘print(3 >= 3 and not (2 > 5))‘ show?

   _____

3. How many times does the block of ‘for i in range(3, 8):‘ run?

   _____

4. Which contract lets another calculation use the fee-inclusive amount directly?

   ☐ Return the fee-inclusive amount

   ☐ Ask the caller to read the terminal

   ☐ Print the amount and return nothing explicitly

5. With ‘s = [10, 20, 30, 40, 50]‘, what is ‘s[1:4]‘?

   _____

6. A file holds three lines: ‘Ana,72‘, ‘Ben,55‘, ‘Cai,90‘. The loop adds ‘int(line.strip().split(”,”)[1])‘ for every line. What is the total?

   _____

7. Which split of a ”print a receipt” program is the best decomposition?

   ☐ One function per line of the receipt

   ☐ One function that reads, calculates and prints everything

   ☐ Two functions, the first half and the second half of the code

   ☐ read_items, line_total, order_total, format_receipt, each doing one job

8. What is the value of ‘17 % 5‘?

   _____

9. A line that is not indented, written straight after an ‘if‘ block, runs only when the condition is ‘True‘.

   ☐ True ☐ False

10. After ‘total = 0‘ and ‘for p in [4, 8, 15]: total += p‘, what is ‘total‘?

    _____

exercises.lol exercises.com.cn

1. **34**

   Both values are strings, so ‘+‘ joins them. Only numbers would give ‘7‘.

2. **True**

   ‘3 >= 3‘ is ‘True‘; ‘2 > 5‘ is ‘False‘, so ‘not (2 > 5)‘ is ‘True‘. Both sides are true, so ‘and‘ gives ‘True‘.

3. **5**

   ‘i‘ takes ‘3, 4, 5, 6, 7‘; the end value 8 is left out. ‘range(a, b)‘ runs $b - a = 8 - 3 = 5$ times.

4. **Return the fee-inclusive amount**

   A returned result can be used in an expression; printed output is a separate side effect.

5. **[20, 30, 40]**

   Position 1 holds 20 (counting from 0), and the slice stops before position 4, so it takes positions 1, 2 and 3.

6. **217**

   $72 + 55 + 90 = 217$. ‘strip()‘ removes the ‘”\n”‘, ‘split(”,”)‘ separates the name from the mark, and ‘[1]‘ takes the mark.

7. **read_items, line_total, order_total, format_receipt, each doing one job**

   Each piece has one job and a clear input and output, so each can be tested alone. Splitting by position in the code gives pieces with no clear contract.

8. **2**

   ‘%‘ gives the remainder: $17 = 3 \times 5 + 2$, so ‘17 % 5‘ is ‘2‘ and ‘17 // 5‘ is ‘3‘.

9. **False**

   Only indented lines belong to the ‘if‘. The first line back at the left edge is outside the block, so it runs either way.

10. **27**

    The total goes $0 \to 4 \to 12 \to 27$, one step for each item in the list.