

Indentation matters

A common mistake is incorrect indentation. Suppose we take the above and indent the last line. The program will still run, but it won't run as expected.

```
from random import randint

count = 0
for i in range(10000):
    num = randint(1, 100)
    if num%12==0:
        count=count+1
    print('Number of multiples of 12:', count)
```

When we run it, it outputs a whole bunch of numbers. The reason for this is that by indenting the print statement, we have made it a part of the for loop, so the print statement will be executed 10,000 times.

Suppose we indent the print statement one step further, like below.

```
from random import randint

count = 0
for i in range(10000):
    num = randint(1, 100)
    if num%12==0:
        count=count+1
        print('Number of multiples of 12:', count)
```

Now, not only is it part of the for loop, but it is also part of the if statement. What will happen is every time we find a new multiple of 12, we will print the count. Neither this, nor the previous example, is what we want. We just want to print the count once at the end of the program, so we don't want the print statement indented at all.

5.9 Exercises

1. Write a program that counts how many of the squares of the numbers from 1 to 100 end in a 1.
2. Write a program that counts how many of the squares of the numbers from 1 to 100 end in a 4 and how many end in a 9.
3. Write a program that asks the user to enter a value n , and then computes $(1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}) - \ln(n)$. The \ln function is `log` in the `math` module.
4. Write a program to compute the sum $1 - 2 + 3 - 4 + \dots + 1999 - 2000$.

5. Write a program that asks the user to enter a number and prints the sum of the divisors of that number. The sum of the divisors of a number is an important function in number theory.
6. A number is called a *perfect number* if it is equal to the sum of all of its divisors, not including the number itself. For instance, 6 is a perfect number because the divisors of 6 are 1, 2, 3, 6 and $6 = 1 + 2 + 3$. As another example, 28 is a perfect number because its divisors are 1, 2, 4, 7, 14, 28 and $28 = 1 + 2 + 4 + 7 + 14$. However, 15 is not a perfect number because its divisors are 1, 3, 5, 15 and $15 \neq 1 + 3 + 5$. Write a program that finds all four of the perfect numbers that are less than 10000.
7. An integer is called *squarefree* if it is not divisible by any perfect squares other than 1. For instance, 42 is squarefree because its divisors are 1, 2, 3, 6, 7, 21, and 42, and none of those numbers (except 1) is a perfect square. On the other hand, 45 is not squarefree because it is divisible by 9, which is a perfect square. Write a program that asks the user for an integer and tells them if it is squarefree or not.
8. Write a program that swaps the values of three variables x , y , and z , so that x gets the value of y , y gets the value of z , and z gets the value of x .
9. Write a program to count how many integers from 1 to 1000 are not perfect squares, perfect cubes, or perfect fifth powers.
10. Ask the user to enter 10 test scores. Write a program to do the following:
 - (a) Print out the highest and lowest scores.
 - (b) Print out the average of the scores.
 - (c) Print out the second largest score.
 - (d) If any of the scores is greater than 100, then after all the scores have been entered, print a message warning the user that a value over 100 has been entered.
 - (e) Drop the two lowest scores and print out the average of the rest of them.
11. Write a program that computes the factorial of a number. The factorial, $n!$, of a number n is the product of all the integers between 1 and n , including n . For instance, $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$. [Hint: Try using a multiplicative equivalent of the summing technique.]
12. Write a program that asks the user to guess a random number between 1 and 10. If they guess right, they get 10 points added to their score, and they lose 1 point for an incorrect guess. Give the user five numbers to guess and print their score after all the guessing is done.
13. In the last chapter there was an exercise that asked you to create a multiplication game for kids. Improve your program from that exercise to keep track of the number of right and wrong answers. At the end of the program, print a message that varies depending on how many questions the player got right.
14. This exercise is about the well-known Monty Hall problem. In the problem, you are a contestant on a game show. The host, Monty Hall, shows you three doors. Behind one of those doors is a prize, and behind the other two doors are goats. You pick a door. Monty Hall, who

knows behind which door the prize lies, then opens up one of the doors that doesn't contain the prize. There are now two doors left, and Monty gives you the opportunity to change your choice. Should you keep the same door, change doors, or does it not matter?

- (a) Write a program that simulates playing this game 10000 times and calculates what percentage of the time you would win if you switch and what percentage of the time you would win by not switching.
- (b) Try the above but with four doors instead of three. There is still only one prize, and Monty still opens up one door and then gives you the opportunity to switch.