You can put any conditional test in the first line and just about any action in the indented block following the test. If the conditional test evaluates to True, Python executes the code following the if statement. If the test evaluates to False, Python ignores the code following the if statement.

Let's say we have a variable representing a person's age, and we want to know if that person is old enough to vote. The following code tests whether the person can vote:

voting.py

```
age = 19
❶ if age >= 18:
❷     print("You are old enough to vote!")
```

At ❶ Python checks to see whether the value of age is greater than or equal to 18. It is, so Python executes the indented print() call at ❷:

```
You are old enough to vote!
```

Indentation plays the same role in if statements as it did in for loops. All indented lines after an if statement will be executed if the test passes, and the entire block of indented lines will be ignored if the test does not pass.

You can have as many lines of code as you want in the block following the if statement. Let's add another line of output if the person is old enough to vote, asking if the individual has registered to vote yet:

```
age = 19
if age >= 18:
    print("You are old enough to vote!")
    print("Have you registered to vote yet?")
```

The conditional test passes, and both print() calls are indented, so both lines are printed:

```
You are old enough to vote!
Have you registered to vote yet?
```

If the value of age is less than 18, this program would produce no output.

if-else Statements

Often, you'll want to take one action when a conditional test passes and a different action in all other cases. Python's if-else syntax makes this possible. An if-else block is similar to a simple if statement, but the else statement allows you to define an action or set of actions that are executed when the conditional test fails.

We'll display the same message we had previously if the person is old enough to vote, but this time we'll add a message for anyone who is not old enough to vote:

```
age = 17
❶ if age >= 18:
        print("You are old enough to vote!")
        print("Have you registered to vote yet?")
❷ else:
        print("Sorry, you are too young to vote.")
        print("Please register to vote as soon as you turn 18!")
```

If the conditional test at ❶ passes, the first block of indented `print()` calls is executed. If the test evaluates to False, the else block at ❷ is executed. Because age is less than 18 this time, the conditional test fails and the code in the else block is executed:

```
Sorry, you are too young to vote.
Please register to vote as soon as you turn 18!
```

This code works because it has only two possible situations to evaluate: a person is either old enough to vote or not old enough to vote. The if-else structure works well in situations in which you want Python to always execute one of two possible actions. In a simple if-else chain like this, one of the two actions will always be executed.

The if-elif-else Chain

Often, you'll need to test more than two possible situations, and to evaluate these you can use Python's if-elif-else syntax. Python executes only one block in an if-elif-else chain. It runs each conditional test in order until one passes. When a test passes, the code following that test is executed and Python skips the rest of the tests.

Many real-world situations involve more than two possible conditions. For example, consider an amusement park that charges different rates for different age groups:

- Admission for anyone under age 4 is free.
- Admission for anyone between the ages of 4 and 18 is $25.
- Admission for anyone age 18 or older is $40.

How can we use an if statement to determine a person's admission rate? The following code tests for the age group of a person and then prints an admission price message:

amusement _park.py
```
age = 12

❶ if age < 4:
        print("Your admission cost is $0.")
```

```
❷ elif age < 18:
       print("Your admission cost is $25.")
❸ else:
       print("Your admission cost is $40.")
```

The if test at ❶ tests whether a person is under 4 years old. If the test passes, an appropriate message is printed and Python skips the rest of the tests. The elif line at ❷ is really another if test, which runs only if the previous test failed. At this point in the chain, we know the person is at least 4 years old because the first test failed. If the person is under 18, an appropriate message is printed and Python skips the else block. If both the if and elif tests fail, Python runs the code in the else block at ❸.

In this example the test at ❶ evaluates to False, so its code block is not executed. However, the second test evaluates to True (12 is less than 18) so its code is executed. The output is one sentence, informing the user of the admission cost:

```
Your admission cost is $25.
```

Any age greater than 17 would cause the first two tests to fail. In these situations, the else block would be executed and the admission price would be $40.

Rather than printing the admission price within the if-elif-else block, it would be more concise to set just the price inside the if-elif-else chain and then have a simple print() call that runs after the chain has been evaluated:

```
age = 12

if age < 4:
❶     price = 0
elif age < 18:
❷     price = 25
else:
❸     price = 40

❹ print(f"Your admission cost is ${price}.")
```

The lines at ❶, ❷, and ❸ set the value of price according to the person's age, as in the previous example. After the price is set by the if-elif-else chain, a separate unindented print() call ❹ uses this value to display a message reporting the person's admission price.

This code produces the same output as the previous example, but the purpose of the if-elif-else chain is narrower. Instead of determining a price and displaying a message, it simply determines the admission price. In addition to being more efficient, this revised code is easier to modify than the original approach. To change the text of the output message, you would need to change only one print() call rather than three separate print() calls.