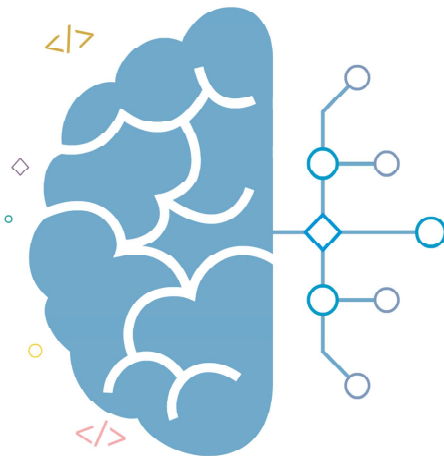




UNIT 2

Conditional statements
and flow control





Overview

Computer programs are useful because of their ability to make decisions while the program is running. In this unit, you will learn how to write conditional statements and use them to run or skip code blocks.

You will also learn the different ways to repeat the same code blocks based on set conditions or a specific number of times.

Key terms

Table 4

relational operators	operators used to compare values such as: ==, >, <, <=, >=, !=
logical operators	operators used to evaluate expressions which result in either true or false such as: or, and, not
simple conditional statement	an expression that uses relational operators to compare values; they evaluate to either true or false
compound conditional statement	an expression that combines multiple simple conditional statements using logical operators
if-else statement	selection/decision statement used to choose the code blocks to run
for loop	repetition statement that will repeat a code block a known number of times
while loop	repetition statement that will repeat a code block based on the result of a conditional statement

Learning outcomes

After completing this unit, you will be able to:

- ① describe sequencing.
- ① explain how to control the flow of a program.
- ① design computer programs using arithmetic and logical expressions.
- ① design computer programs using appropriate input/output.
- ① test computer programs.
- ① design computer programs using sequencing, selection and repetition statements.
- ① design computer programs to solve real-world problems.

Student digital competency framework

This unit also gives you the opportunity to improve the following digital competencies:

- 📶 Collaborate using digital tools.
- 📶 Leverage technology.
- 📶 Plan, manage and solve problems using digital tools.
- 📶 Integrate simulations.
- 📶 Coding.
- 📶 Lead digital changes.

Conditional statements

Conditional statements are expressions that give either a true or false result. You can write simple conditional statements using variables, numbers, and relational operators. Suppose you stored the number 12 in a variable called ageEman.

ageEman = 12

You can write a simple conditional statement to compare Eman's age as follows:

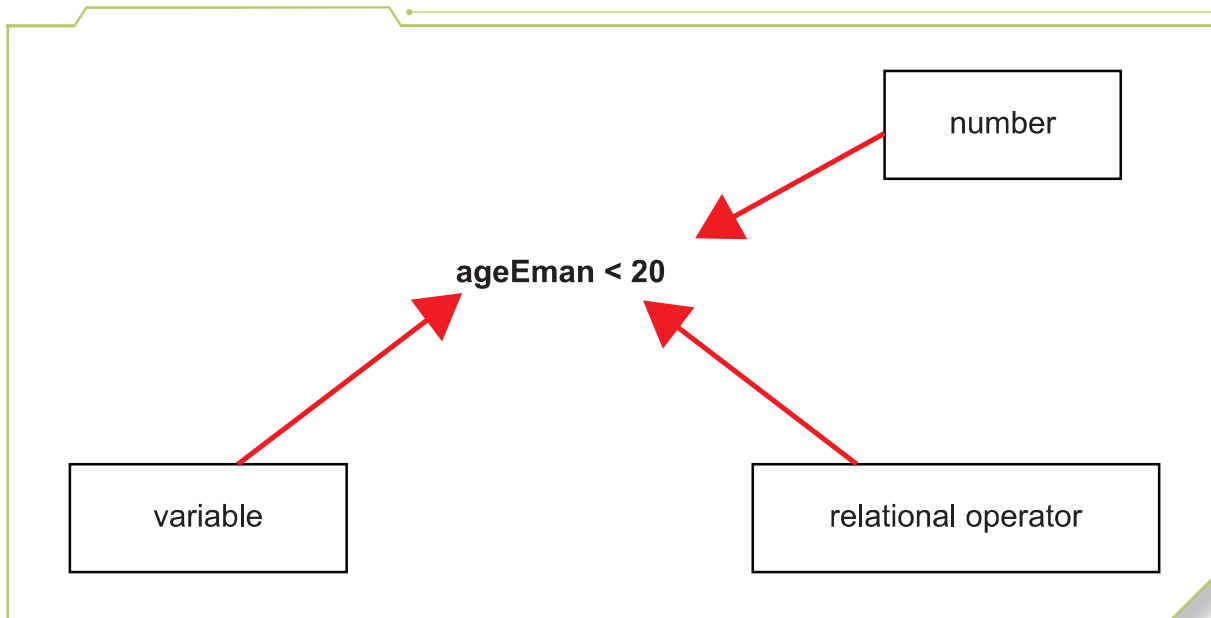


Figure 34

Since ageEman is smaller than 20, the result of this conditional statement is true.

What is the result of the conditional statement below?

ageEman > 20

You will use relational operators to write simple conditional statements. Simple conditional statements are used to compare the values stored in variables and numbers.

Here are a few values and variables you could compare:

Is a bus bigger than a car?



Figure 35

Using the variables:

```
sizeBus = 14
```

and

```
sizeCar = 4.5
```

You can compare the areas in Python using the simple conditional statement:

```
sizeBus > sizeCar
```

The result of this statement is true because of the values stored in the variables.

What about a baby and a grade 10 student, who is older?



Figure 36

Using the two variables:

yearsBaby = 1

and

yearsGrade10 = 7

You can write a simple conditional statement in Python as follows:

yearsBaby > yearsGrade10

Based on the values stored in the variables, the statement will give a false result.

The greater than (>) relational operator will check if the value on the left is bigger than the value on the right. We only want to know if the condition is true or false. The code will decide what to run based on the result of the condition.

More relational operators used to make different types of comparisons are listed below with examples.

The variables: a = 105.2, b = 0, c = 100, and d = 2 are used in the examples.

Table 5 - Relational operators

Relational Operator	Description	Example	True or False
>	greater than	a > 29.9	false
<	less than	b < -1	false
==	equal to	d == 2	true
!=	not equal to	b != c	true

Relational operators compare two values.

You will use two equal signs == to check if two values are equal.

Use = to set a variable to a value that you want. The = is called the assignment operator. It is not a relational operator.

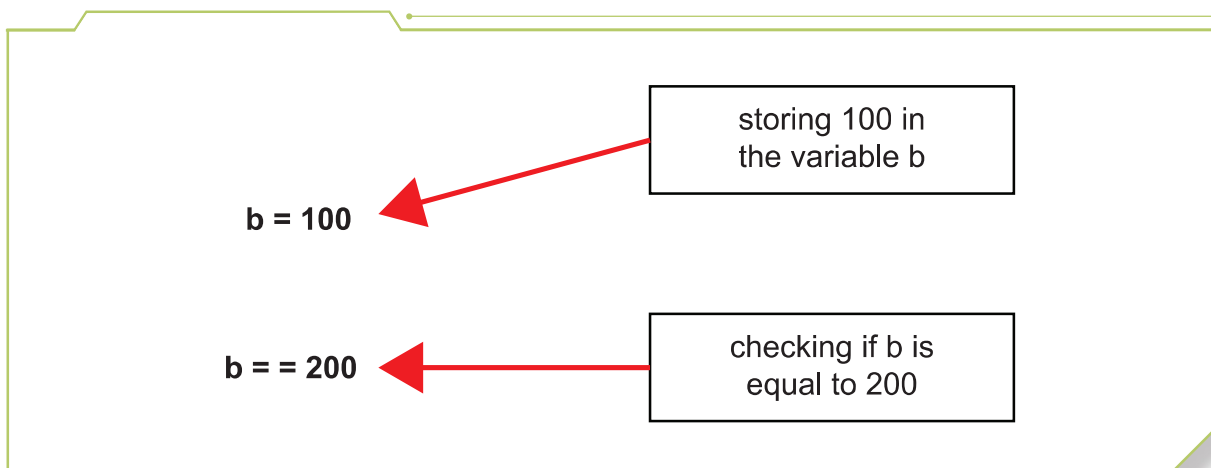


Figure 37



Designing simple conditional statements



Questions for simple conditional statements



Logical operators are used to join simple conditional statements to create compound conditional statements.

The logical operators are; and, or, and not. The result of logical operations is true or false.

Below, you can see how to write a compound conditional statement by joining simple conditional statements.

Look at the two fruits in the pictures.



Figure 38

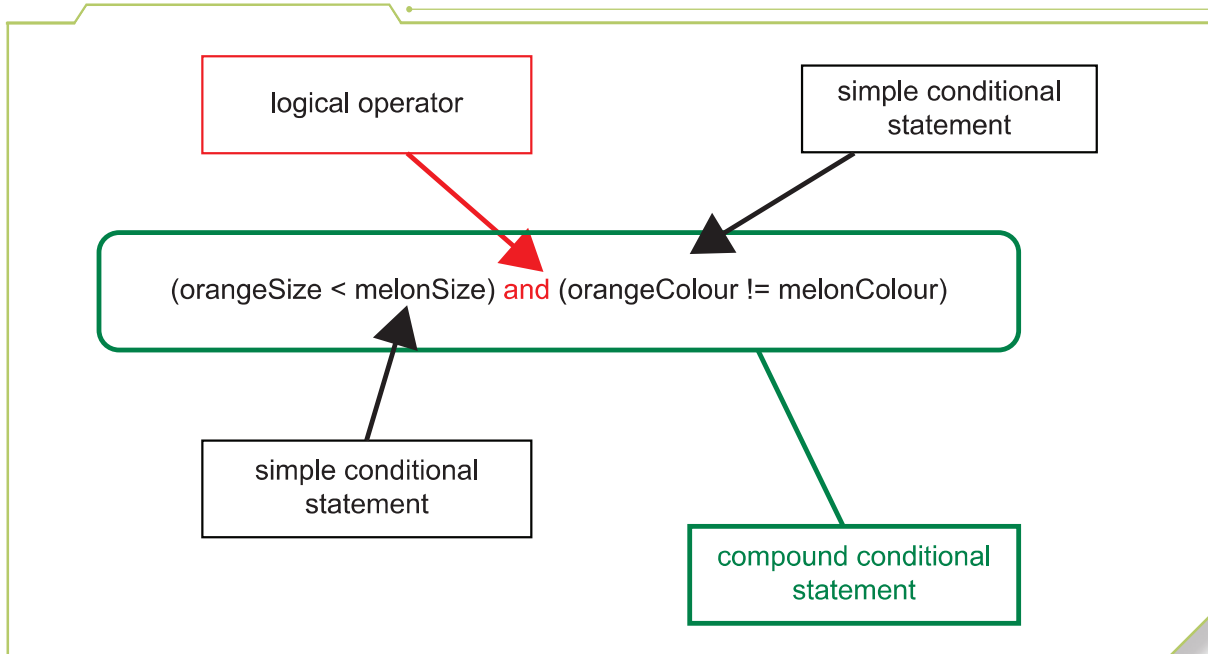


Figure 39

Each logical operator has a truth table. Truth tables give all the possible answers for any inputs for each logical operator.

Suppose you have 2 simple conditional statements: simple conditional statement A and simple conditional statement B. You can write a compound conditional statement for the and logical operator as follows:

(simple conditional statementA) and (simple conditional statementB)

Table 6

Result of simple conditional statement A	Result of simple conditional statement B	Result of compound conditional statement for 'and'
False	False	False
False	True	False
True	False	False
True	True	True

The truth table for the or is shown below. Only one side needs to have a true result for the result to be true.

Table 7

Result of simple conditional statement A	Result of simple conditional statement B	Result of compound conditional statement for 'or'
False	False	False
False	True	True
True	False	True
True	True	True

The not logical operator flips the result of the simple conditional statements. It switches the result of the conditional statement from false to true, or true to false.

The truth table for the not is as follows:

Table 8

SCSA	not (SCSA)
False	True
True	False

Simple conditional statements are used to compare values using relational operators.

Compound conditional statements use logical operators (and, or, not) to join simple conditional statements.

Simple and compound conditional statements give a true or false result.



Writing compound conditional statements



Results of compound conditional statements



Flow control

In this section, you will learn about how to control the flow of logic in computer programs. You learn how to write code that will let the program make decisions without any help.

Selection.

Conditional statements are used in if-statements to let the computer make decisions by itself. It uses the results of conditional statements to run a specific code block or to skip it.

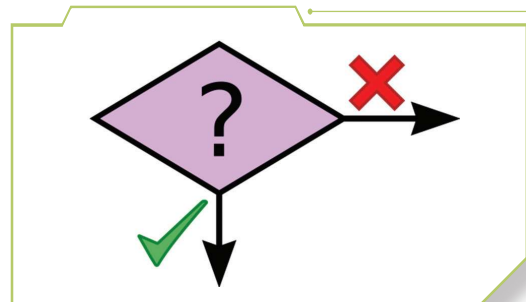


Figure 40

The if statement

The if statement is used together with conditional statements to decide whether to run a block of code or skip it.

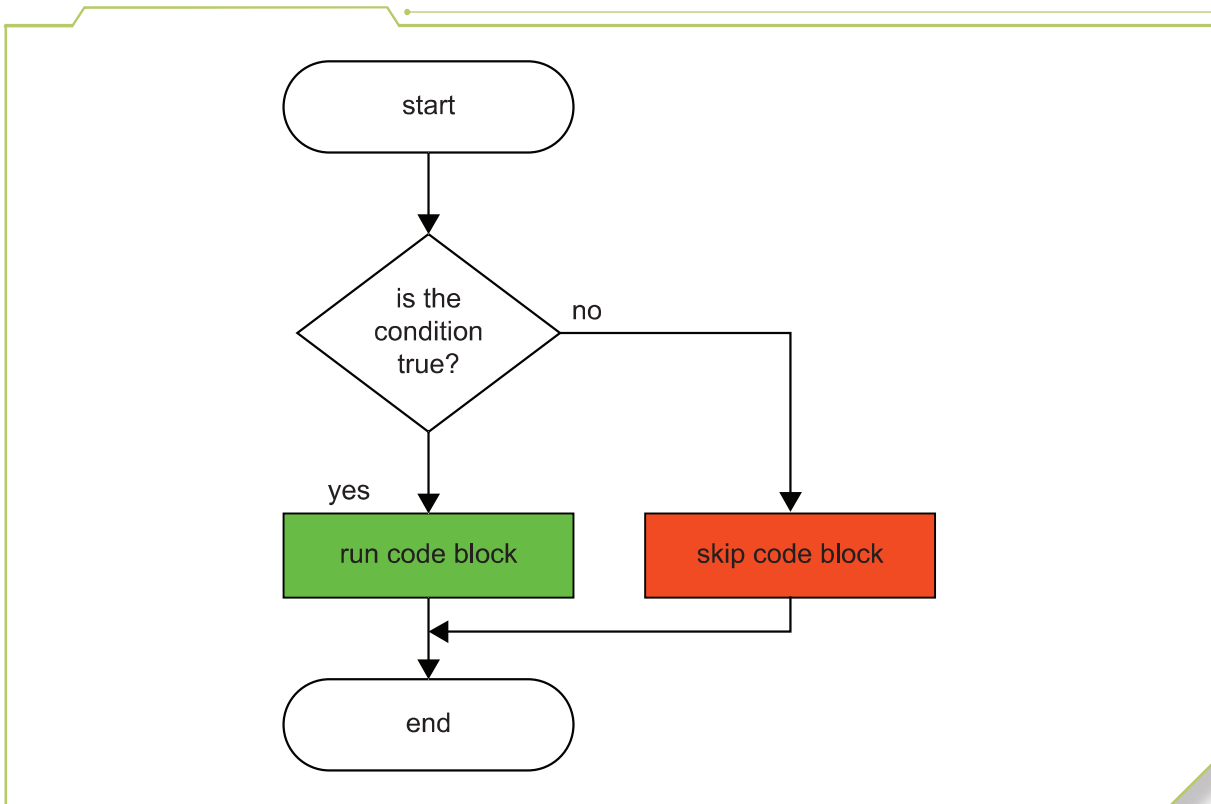


Figure 41

The flowchart below shows an algorithm that checks the temperature of water. When the temperature of the water reaches 100 degrees Celsius, the code block will print a warning message.

The if statement will skip the code block if the temperature is less than 100 degrees Celsius.



Figure 42

In an if statement, the result of the conditional statement determines whether a block of code is run

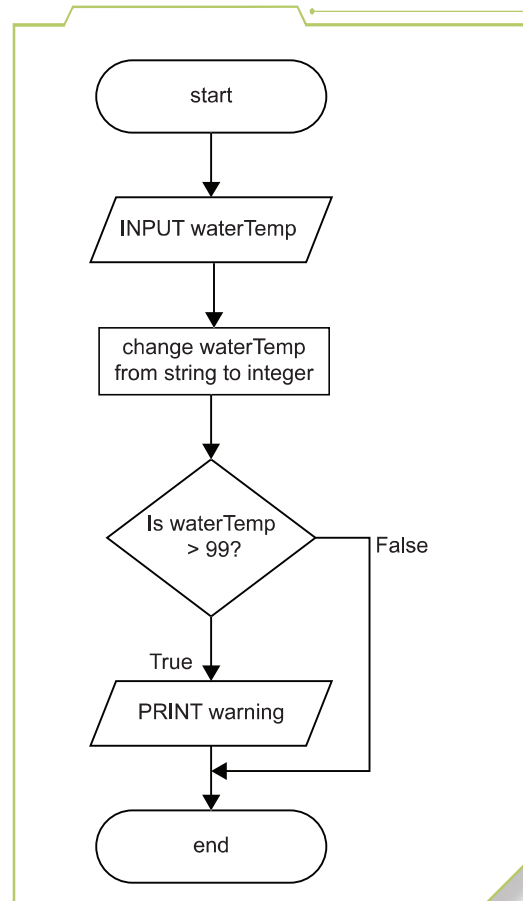


Figure 43

```
temp = input("Enter the water temperature:")
temp = float(temp)

if temp > 99.9:
    print("Warning: too hot!")
```

simple conditional statement

The warning message will only be printed if the user enters a value that is greater than 99.9




Figure 44

To write an if statement, you need to write a conditional statement. Put the if keyword before it and a colon after it.

The print statements will be run if the simple conditional statement, `age > 17`, is true. It will be skipped if the result is false.

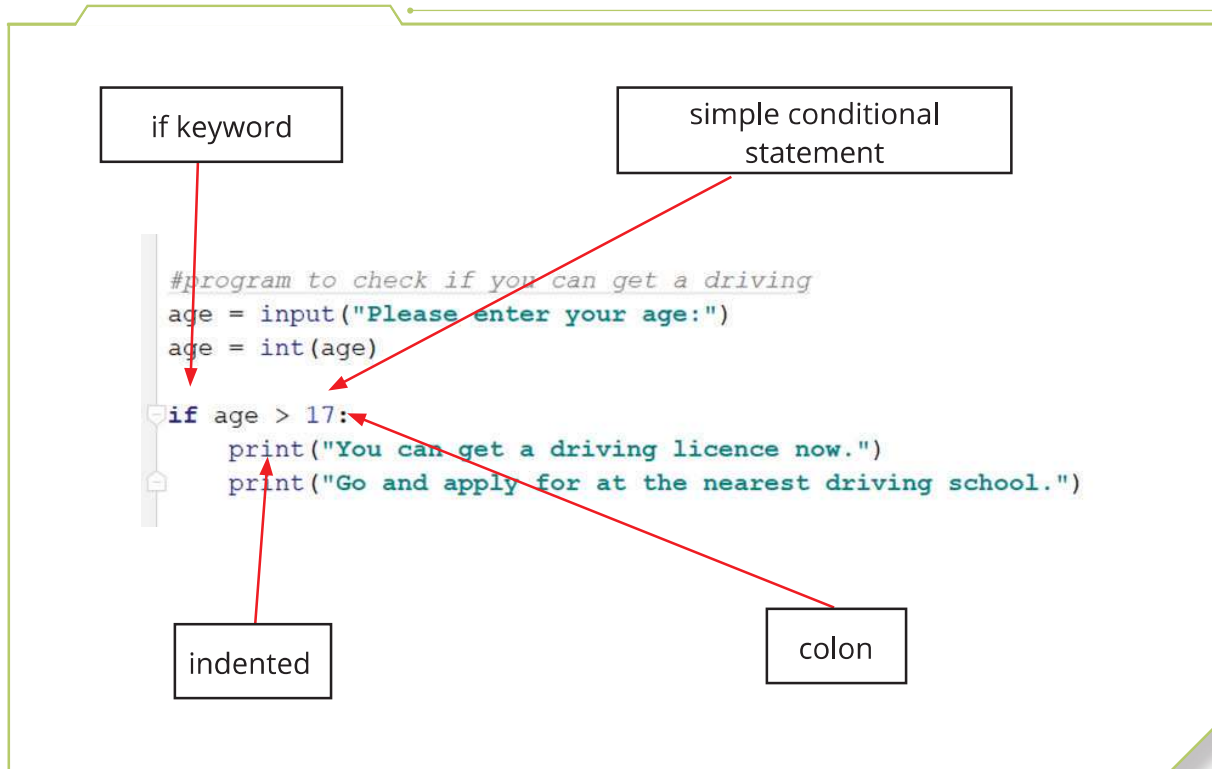


Figure 45

Indent the code that you want the computer to run when the condition is true. Any statement that is not indented is not part of the if statement.



The if statement



Using conditional statements



Using if statements



The if-else statement

The if statement will only run a code block if the result of the conditional statement is true. The code block will be skipped when the condition is false.

But the if-else statement runs one code block when the condition is true and a different code block when the condition is false.



Figure 46

If you store two numbers (e.g. the volumes of bottles of water) in two variables, called `bottleA` and `bottleB`. You can then write a conditional statement to tell which number is bigger and use an if statement to run one code block when `bottleA` is bigger. Adding an else statement will run another code block if `bottleB` is bigger.



Figure 47

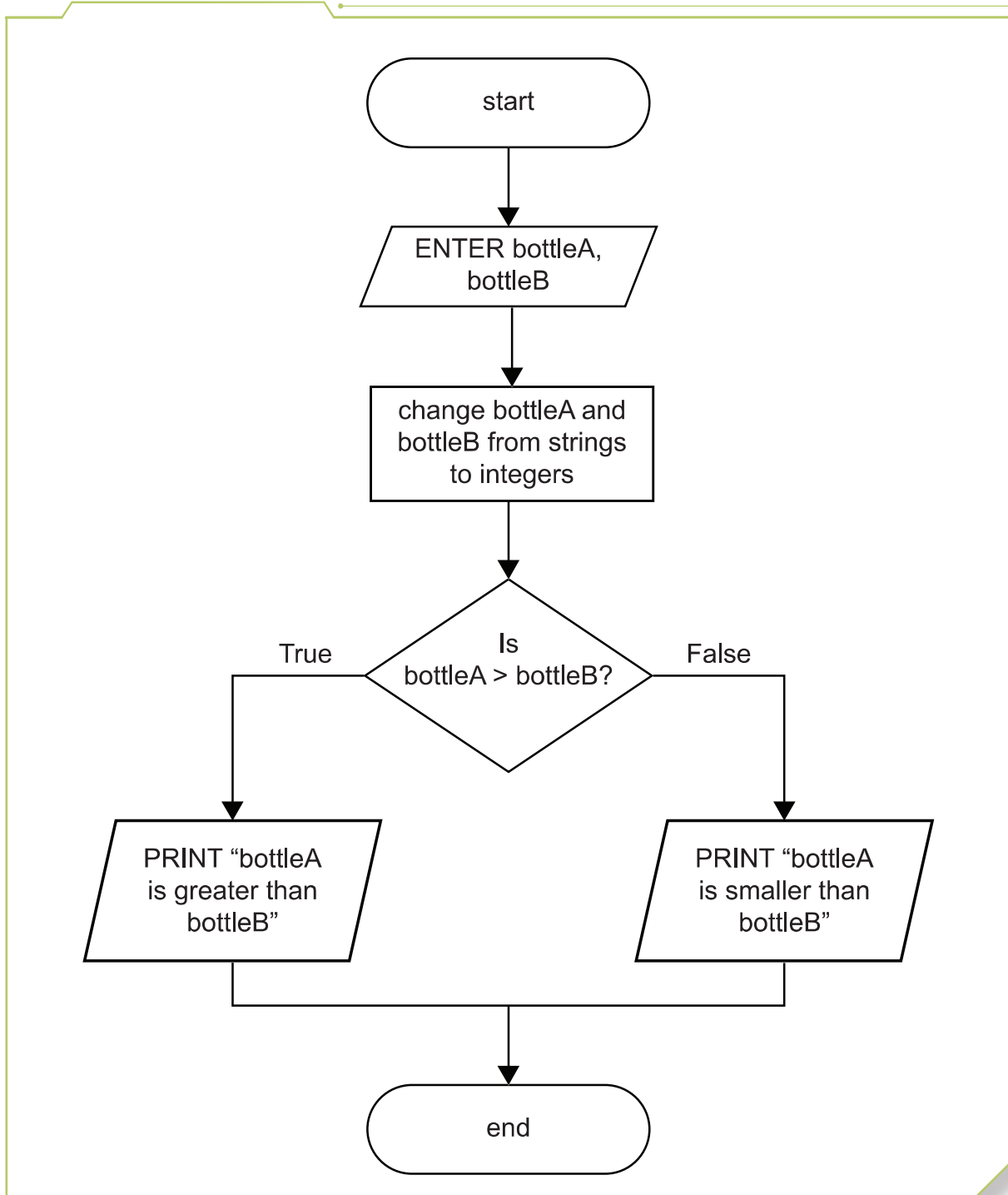


Figure 48

The Python code for this algorithm is:

```
1  ▶ #program to check if bottle A is bigger
2  bottleA = input("Enter the volume of bottle A:")
3  bottleA = int(bottleA)
4
5  bottleB = input("Enter the volume of bottle B:")
6  bottleB = int(bottleB)
7
8  if bottleA > bottleB:
9      print("Bottle A is bigger.")
10 else:
11     print("Bottle A is smaller.")
```

Figure 49

The if-else statement runs one of two blocks of code. The code block after the if or the one after the else.

You do not need to give a conditional statement for the else part. The computer will run the code block under the else if the conditional statement is false.



The difference between the if and the if-else statements



Using the if-else statement



The elif statement

The elif statement combines the else and if and allows you to add more than one conditional statement.

The algorithm for the boiling water could be changed to tell if the water is: frozen, cold, warm, hot or boiling depending on the value of the temperature.

You need the elif statement to add the other conditions. Look at the new flowchart and Python code.



Figure 50

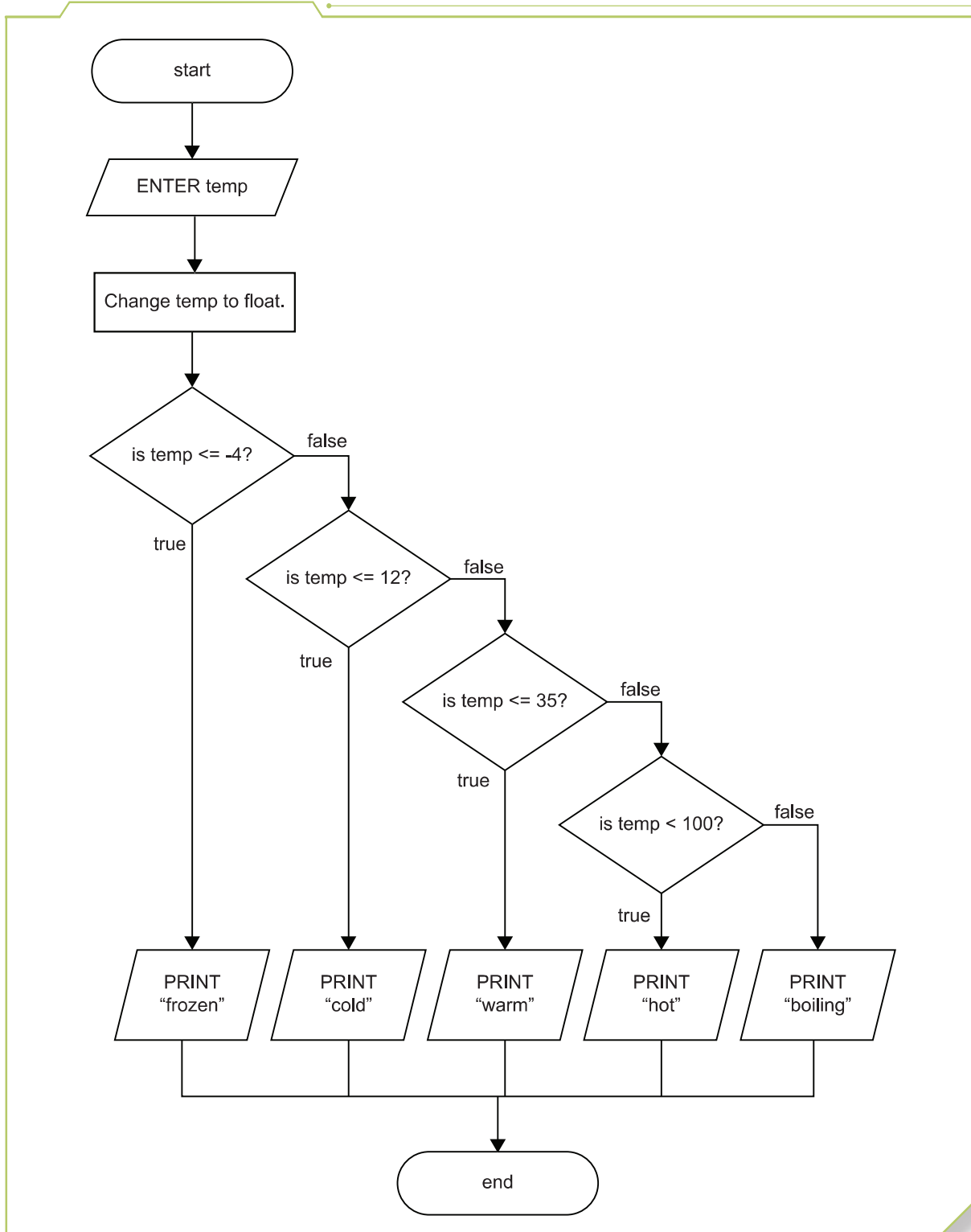


Figure 51


```
temp = input("Enter the temperature: ")
temp = float(temp)

if temp < 1:
    print("Frozen.")
elif temp < 13:
    print("Cold.")
elif temp < 26:
    print("Warm")
elif temp < 51:
    print("Hot")
elif temp < 100:
    print("Very hot.")
else:
    print("Boiling.")
```

Figure 52

The elif statement begins with an if and ends with an else. For each different condition, you must add an elif statement.

Remember the problem from activity 17(a) where you designed a program to say if a number was either positive or negative?

```
1 ▶ # Program to check if the number entered is positive
2   number = input("Please enter a number: ")
3   number = int(number)
4   if number > 0:
5       print("Number ", number, " is positive")
6   else:
7       print("Number ", number, " is negative")
8
```

Figure 53

What happens when a user enters 0? The program prints the message that 'Number 0 is positive.'

This is not true. You were limited to one condition with the if-else statement.

You can now use the elif statement to cover the conditions where the number is positive, negative or zero.

```
1 # Program to check if the number entered is positive
2 number = input("Please enter a number: ")
3 number = int(number)
4 if number > 0:
5     print("Number ", number, " is positive.")
6 elif number < 0:
7     print("Number ", number, " is negative.")
8 else:
9     print("Number ", number, " is 0.")
10
```

Figure 54

You control the flow of your program by setting all the needed conditions using the elif statement. You can use simple conditional statements and compound conditional statements in elif statements.



Using selection - the elif statement



Repetition

Computers can repeat the same code block for as long as you want. This is known as a loop. You will learn about while and for loops in this section:

While loops

while loops repeat the same code block for as long as the result of a conditional statement is true. The number of times the code block will be repeated is unknown.



Figure 55

Look at the flowchart for a while loop below.

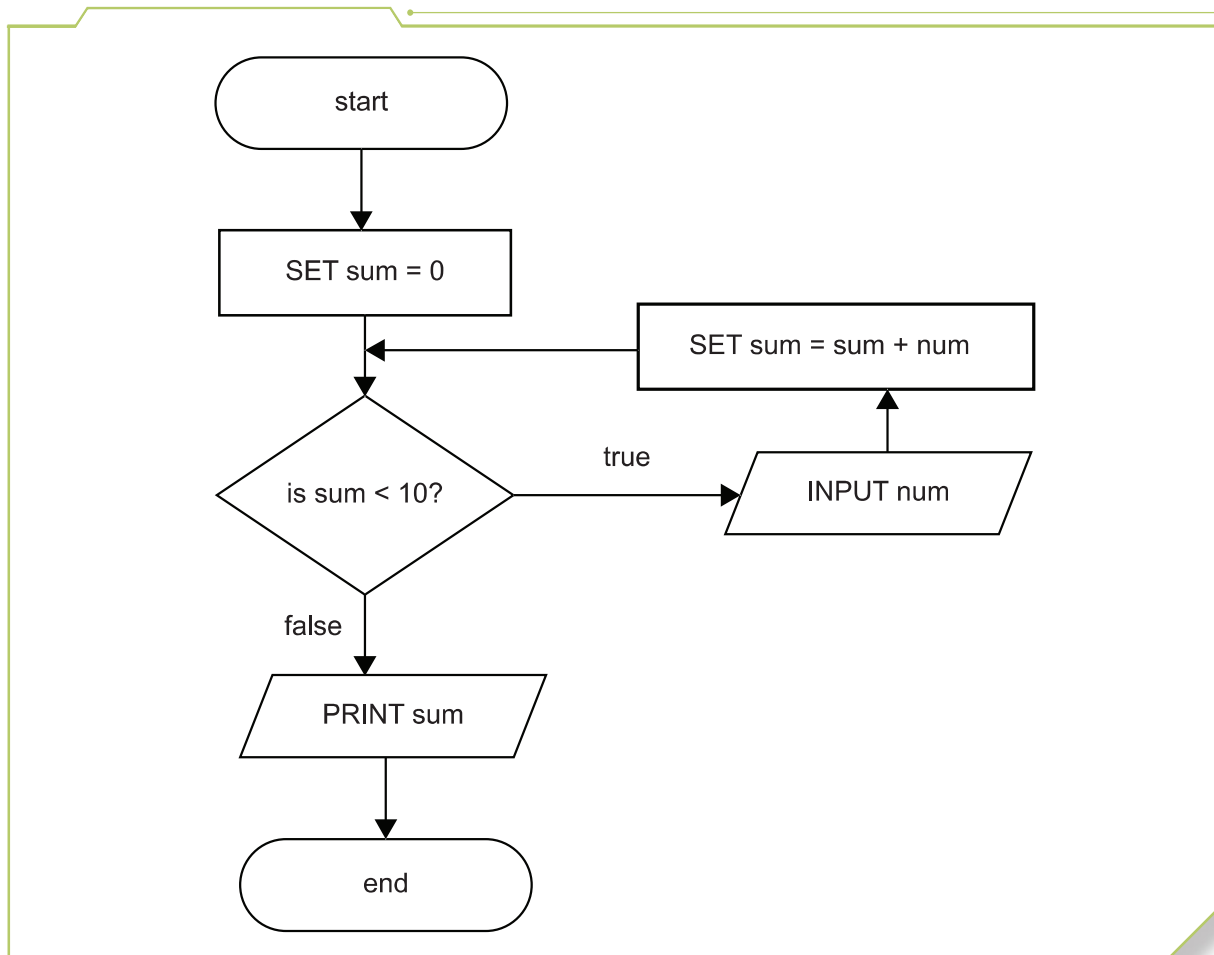


Figure 56

The Python code for the while loop is shown in the following diagram.

You need to start with the while keyword. Then add a conditional statement followed by a colon. Then list the indented code block you want to repeat.

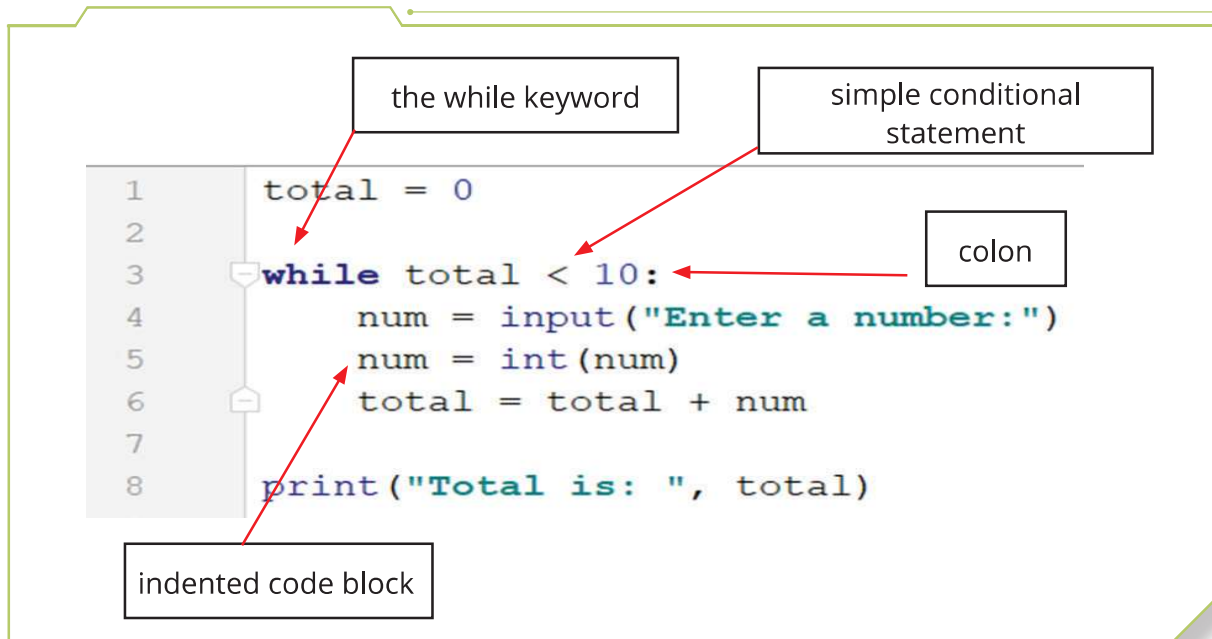


Figure 57

Indent all the code blocks that you want to run in a loop. Use tabs. You will get an error if you mix tabs and spaces.

All indented statements below the while are part of the loop. That code block will be repeated when the condition is true.

You will use conditional statements to control while loops.

Unit 2 Conditional statements and flow control



Use the code below to answer the questions



Conditional statements are very important when you design while loops. The code inside the while loop will repeat for as long as the condition is true.



Using while loops



For loops

You will use the for loop to repeat a code block a set number of times. For example, you go to school for five days every week. You repeat the same actions of getting up and going to school five times every week. You could represent these activities in a flowchart as follows:

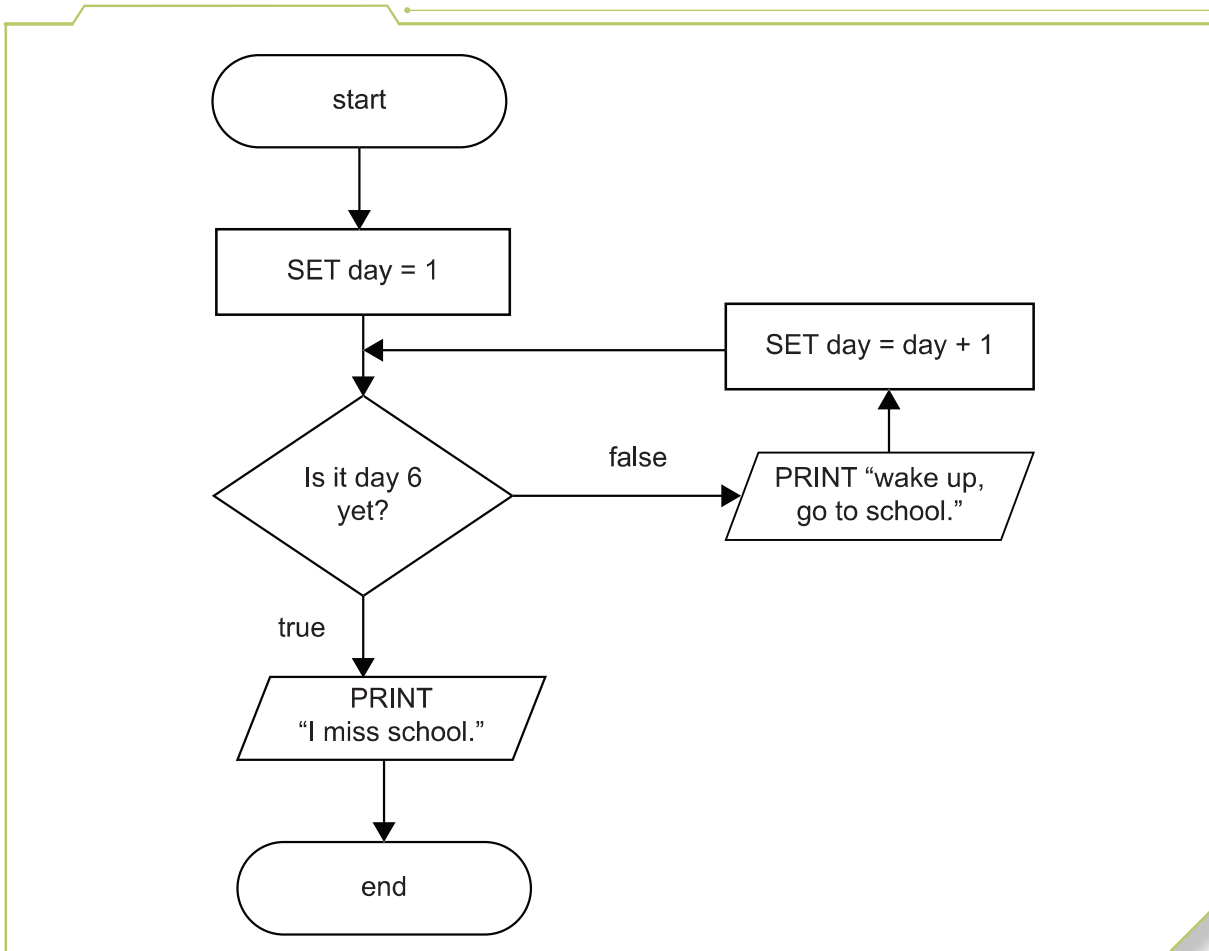


Figure 58

We can then write this for loop in Python as follows:

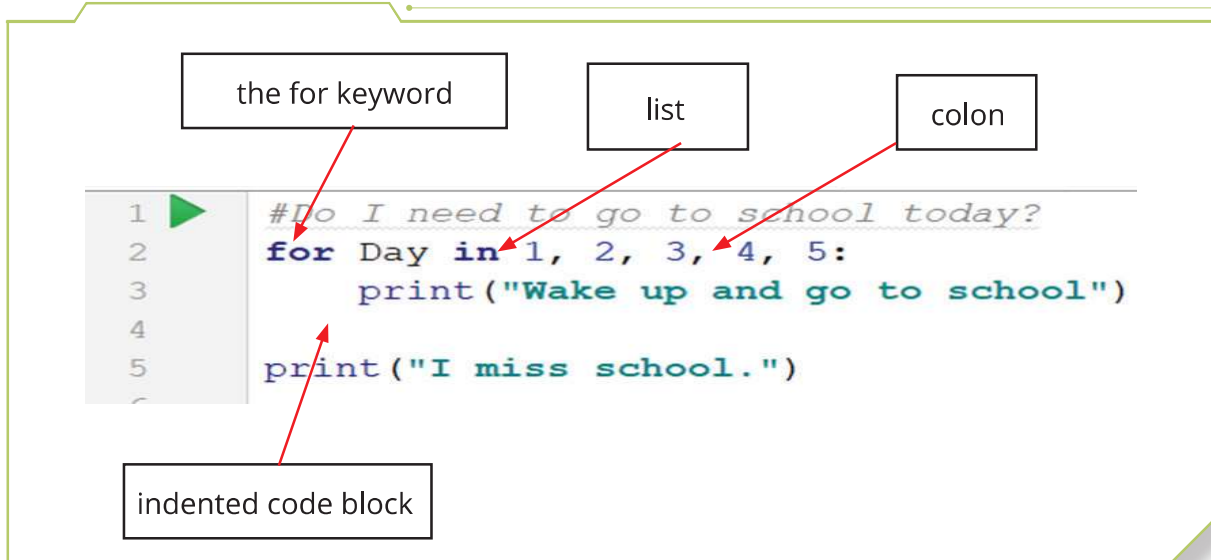


Figure 59

Indent the code you need to repeat under the for loop.

Statements that are not indented are not part of the for loop. The `print('I miss school')` statement is NOT part of the for loop.



Using for loops



If you need to repeat a block of code 1000 times. You would need to write a long list. The `range()` function makes this easy to do.

All you need is to give the `range()` function the start, end and step-size values.

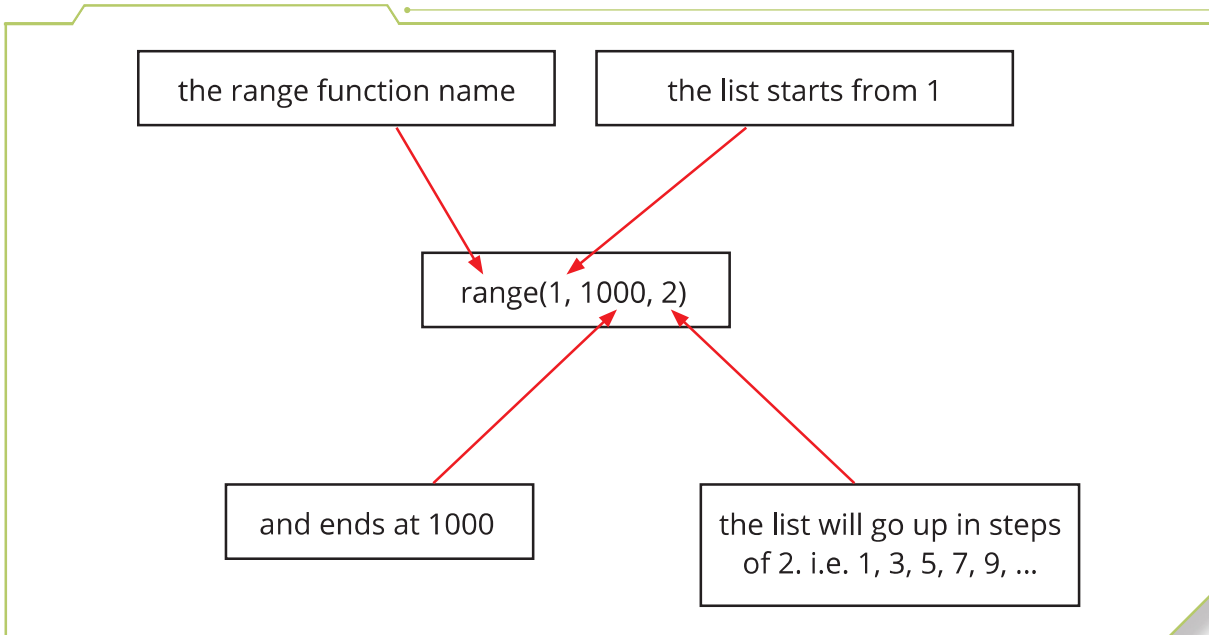


Figure 60



Using range()



State the output



Unit 2 Conditional statements and flow control



Khalifa Fund for Enterprise Development – Main menu



Khalifa Fund for Enterprise Development – Main menu



Completed unit objectives

You have completed the introduction to computer programming. You should now be able to do the following:

- ❖ design computer programs using arithmetic and logical expressions.
- ❖ design computer programs using appropriate input/output.
- ❖ design computer programs using sequencing, selection and repetition statements.

Unit 2

Conditional statements and flow control



Student reflection



End of units activities