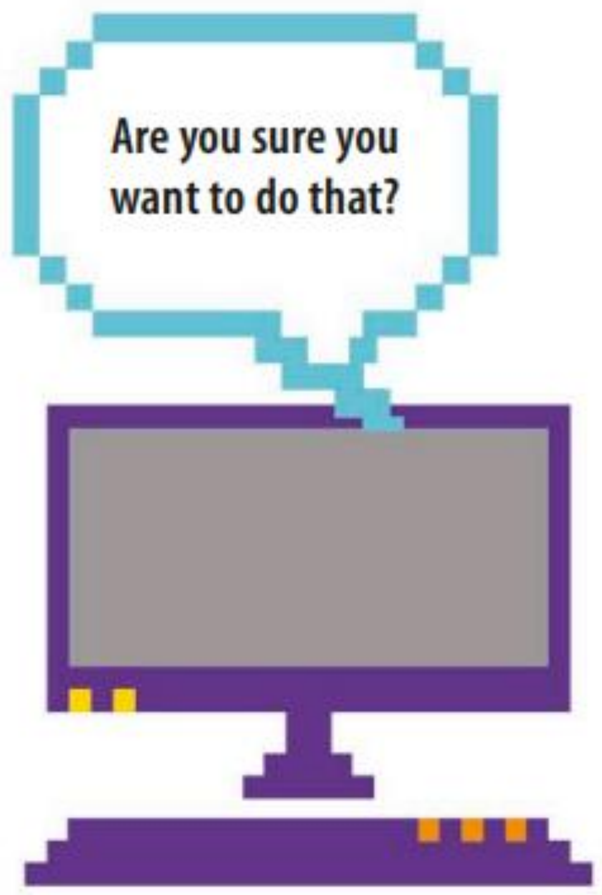
Conditional Statements

Making decisions

Every day you make decisions about what to do next, based on the answers to questions you ask yourself.

Computers also make decisions by asking questions.

The questions that computers ask themselves usually involve comparing one thing with another. For example, a computer might ask if one number is bigger than another. If it is, the computer might then decide to run a block of code that would otherwise be skipped.

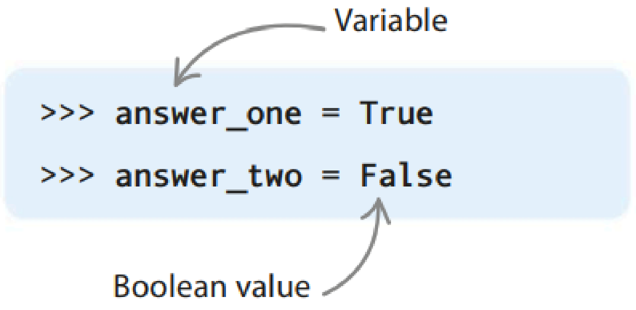


Making decisions

The answers to the questions computers ask have only two possible values: **True or False**.

Python calls these two values **Boolean values**, and they must always start with a capital letter.

You can store a Boolean value in a variable.

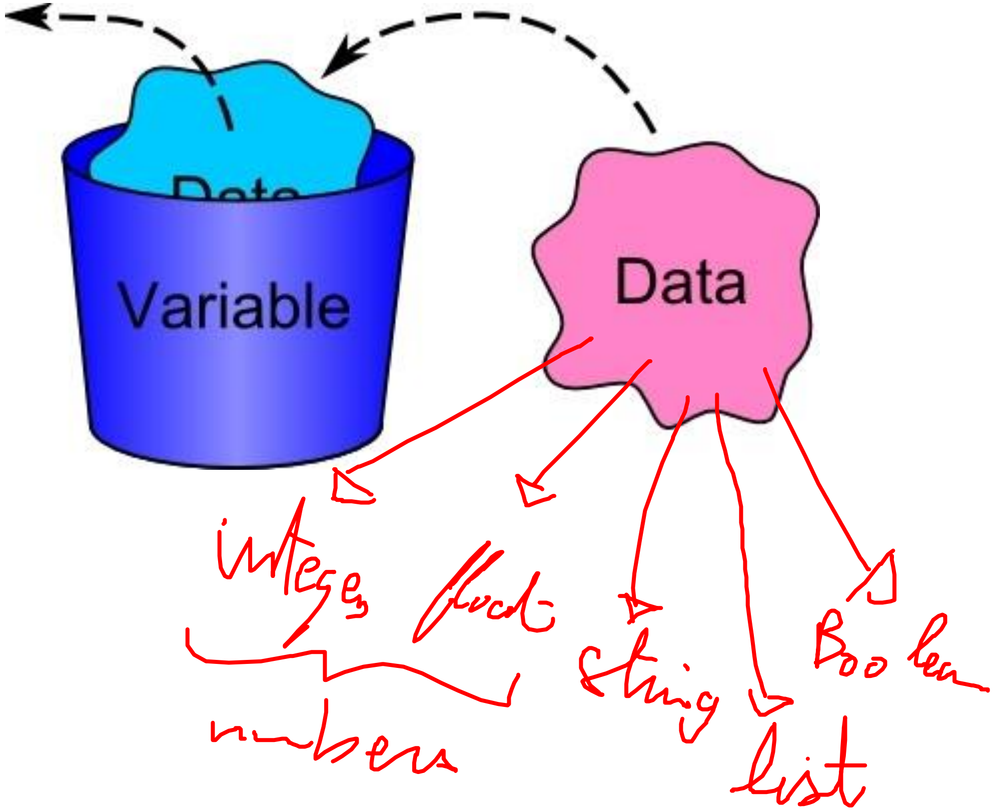


Variable

```
>>> answer_one = True
>>> answer_two = False
```

Boolean value

Data types


Data
Variable
Data
integer
float
string
list
Boolean
numbers

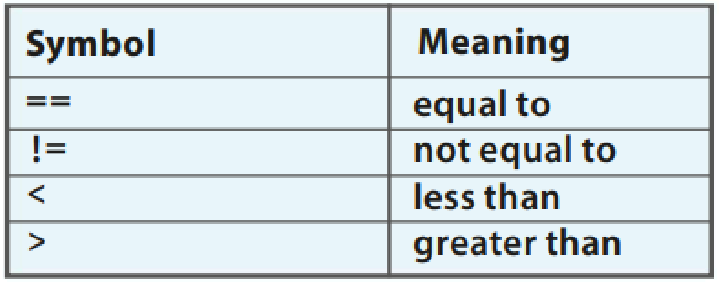
Boolean
(True or false)

Logical Operators

These symbols tell computers to make comparisons. Programmers call them logical operators.

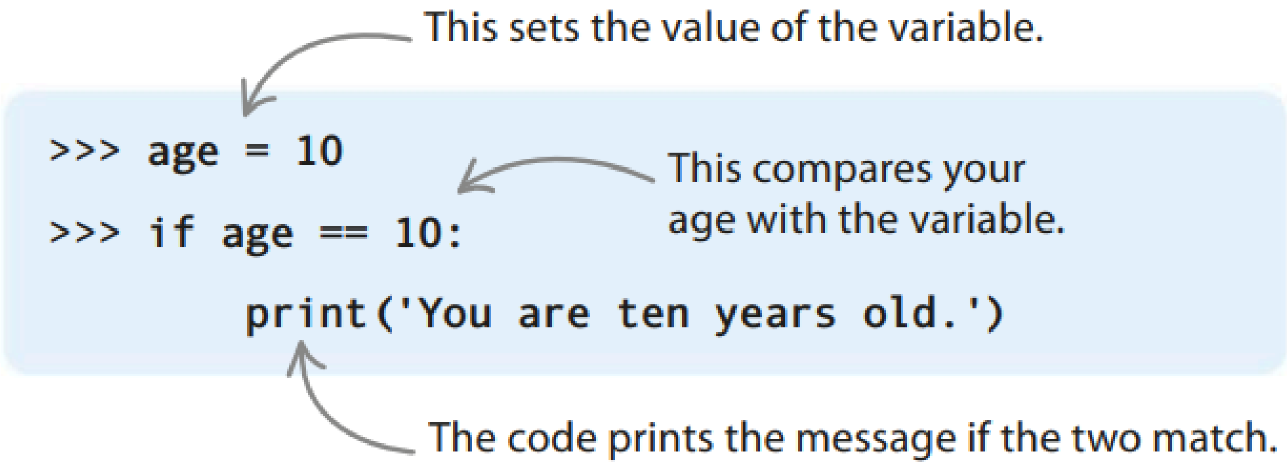
You may have used some of them in math.

The words “and” and ”or” can also be used as logical operators in computer code.



Symbol	Meaning
==	equal to
!=	not equal to
<	less than
>	greater than

Logical Operators



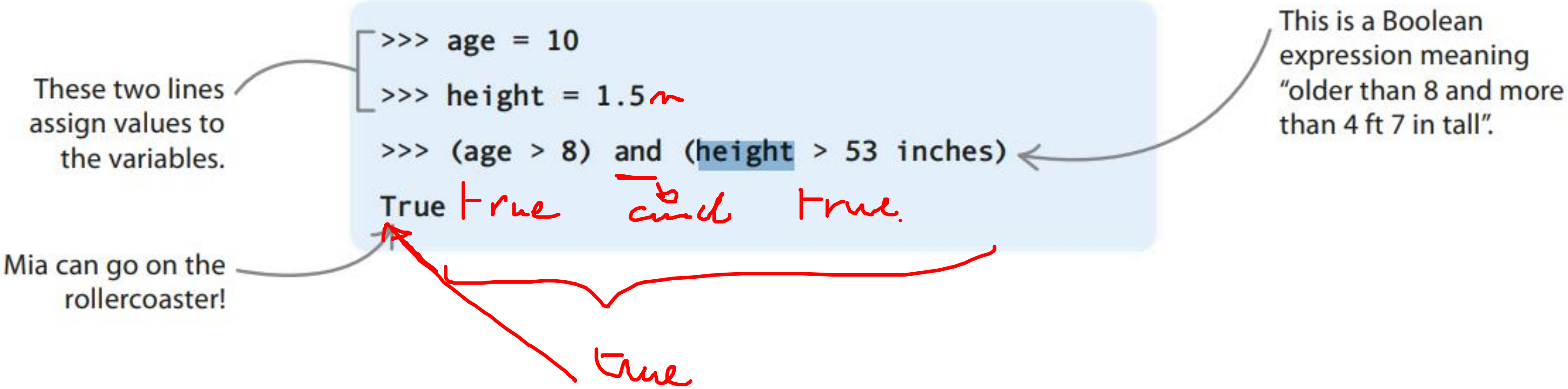
This sets the value of the variable.

```
>>> age = 10
>>> if age == 10:
        print('You are ten years old.')
```

This compares your age with the variable.

The code prints the message if the two match.

Logical Operators



```
>>> age = 10
>>> height = 1.5m
>>> (age > 8) and (height > 53 inches)
True true  and  true
```

true

These two lines assign values to the variables.

This is a Boolean expression meaning "older than 8 and more than 4 ft 7 in tall".

Mia can go on the rollercoaster!

Branching

Computers often need to make decisions about which parts of a program to run. This is because most programs are designed to do different things in different situations. The route through the program splits like a path branching off into side paths, each leading to a different place.

LINGO

Condition

A condition is a Boolean expression (a True-or-False comparison) that helps a computer decide which route to take when it reaches a branch in the code.

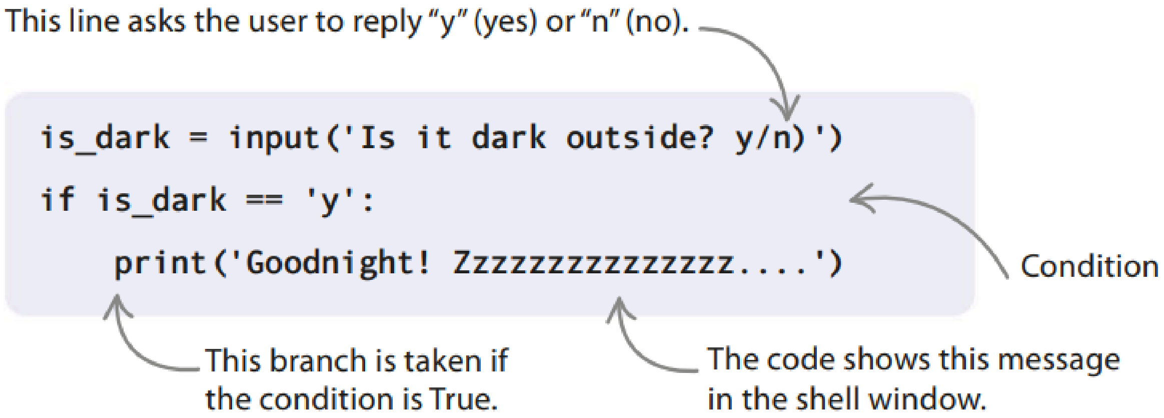
▷ **School or park?**

Imagine you have to decide what route to walk each day based on the answer to the question "Is today a weekday?" If it's a weekday, you take the route to school; if it's not, you take the route to the park. In Python, the different routes through a program lead to different blocks of code. A block can be one statement or several, all indented by four spaces. The computer uses a test called a condition to figure out which blocks it should run next.

Branching

▷ One branch

The simplest branching command is an `if` statement. It only has one branch, which the computer takes if the condition is True. This program asks the user to say if it's dark outside. If it is, the program pretends that the computer is going to sleep! If it's not dark, `is_dark == 'y'` is False, so the "Goodnight!" message isn't displayed.



This line asks the user to reply "y" (yes) or "n" (no).

```
is_dark = input('Is it dark outside? y/n)')
if is_dark == 'y':
    print('Goodnight! Zzzzzzzzzzzzzz....')
```

Condition

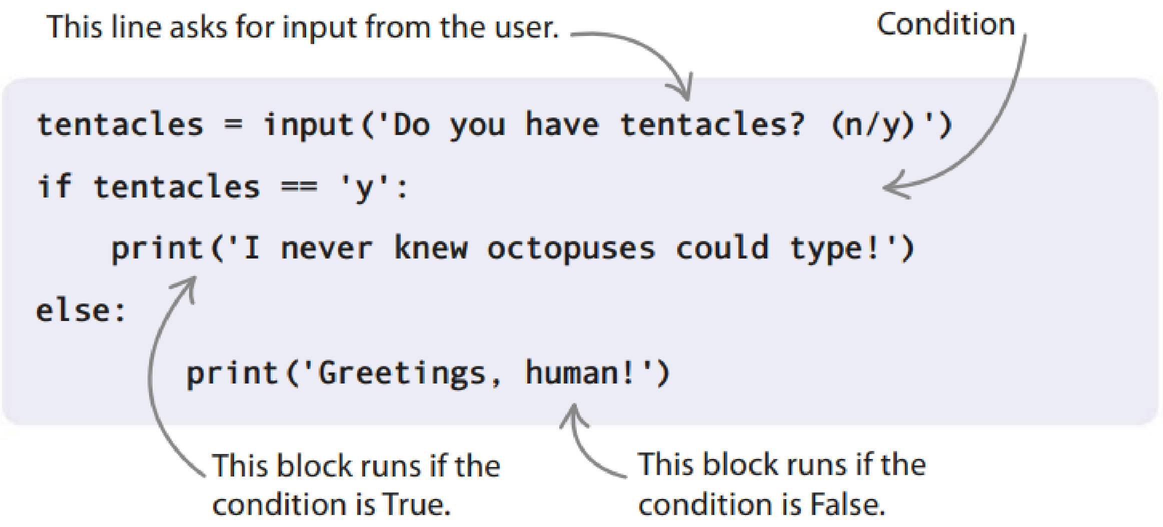
This branch is taken if the condition is True.

The code shows this message in the shell window.

Branching

▷ Two branches

Do you want a program to do one thing if a condition's True and another thing if it's False? If so, you need a command with two branches, called an `if-else` statement. This program asks if the user has tentacles. If they answer "Yes", it decides they must be an octopus! If they answer "No", it decides they're human. Each decision prints a different message.



This line asks for input from the user.

Condition

```
tentacles = input('Do you have tentacles? (n/y)')
if tentacles == 'y':
    print('I never knew octopuses could type!')
else:
        print('Greetings, human!')
```

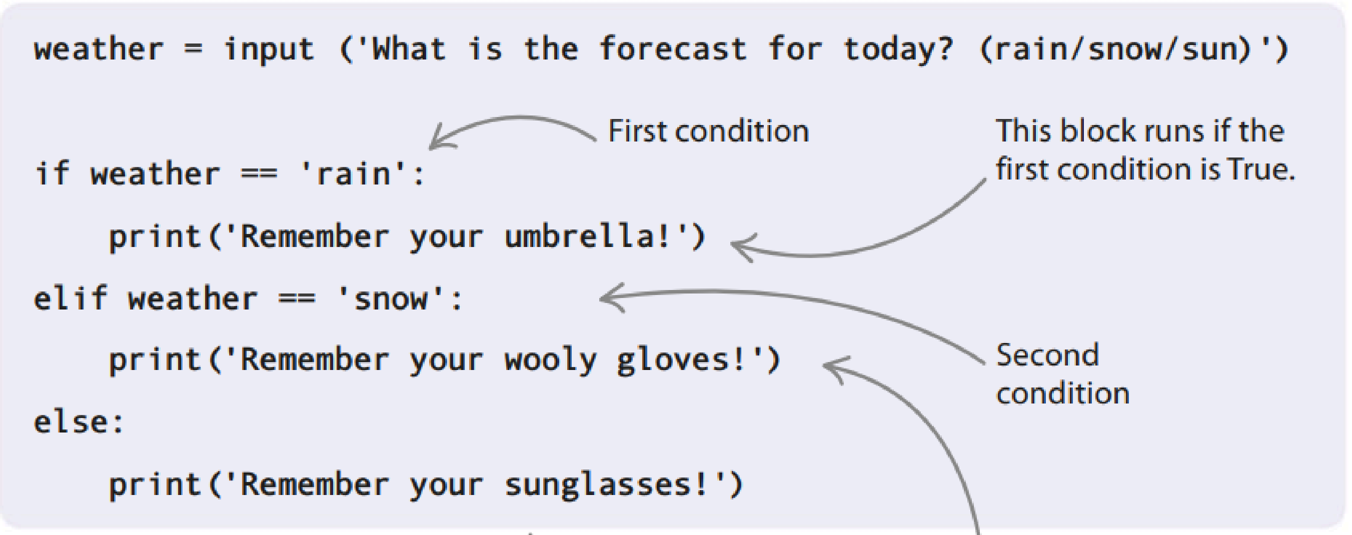
This block runs if the condition is True.

This block runs if the condition is False.

Branching

▷ Multiple branches

When there are more than two possible paths, the statement `elif` (short for “else-if”) comes in handy. This program asks the user to type in the weather forecast: either “rain”, “snow”, or “sun”. It then chooses one of three branches and weather conditions.



```
weather = input ('What is the forecast for today? (rain/snow/sun)')

if weather == 'rain':
    print('Remember your umbrella!')
elif weather == 'snow':
    print('Remember your wooly gloves!')
else:
    print('Remember your sunglasses!')
```

First condition

This block runs if the first condition is True.

Second condition

This block runs if the second condition is True.

This block runs if both conditions are False.

△ How it works

An `elif` statement must always come after `if` and before `else`. In this code, `elif` checks for snow only when the condition set by the `if` statement is False. You could insert additional `elif` statements to check for more types of weather.

Time to code

Primitive calculator

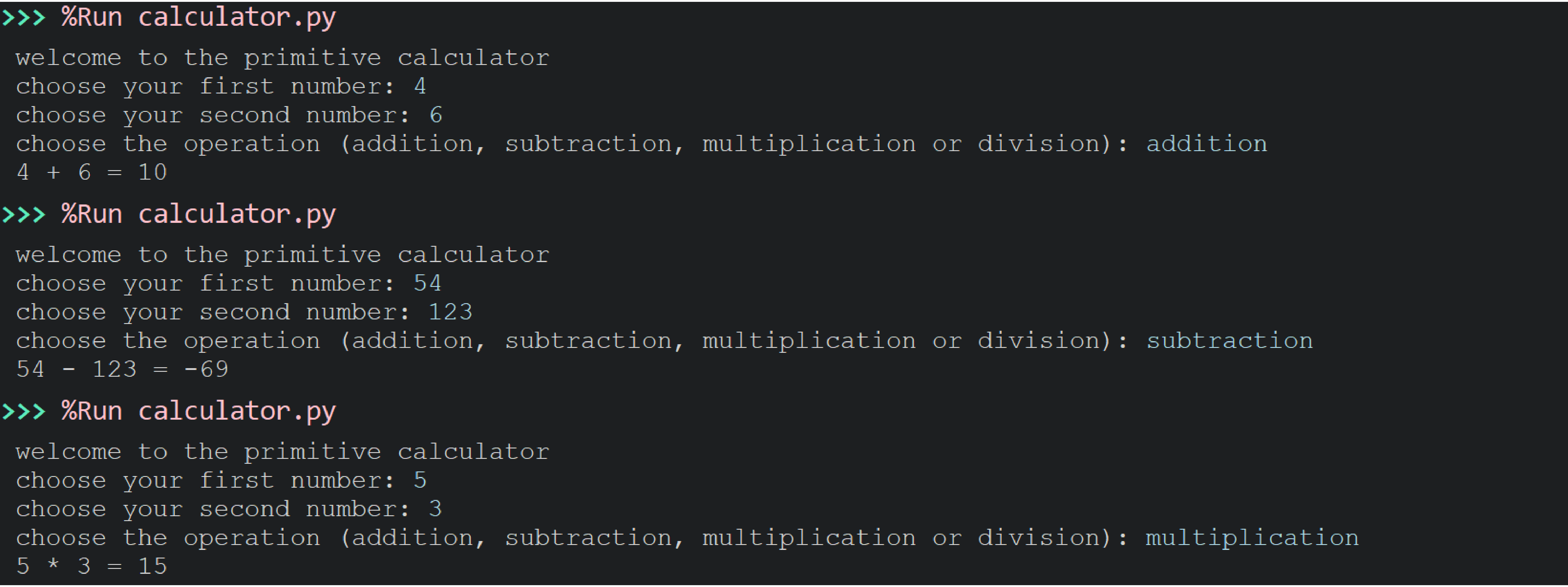
Ask the user to input two numbers. (x and y)

Ask the user to write the type of operation he wants to do. (addition, subtraction, multiplication, division)

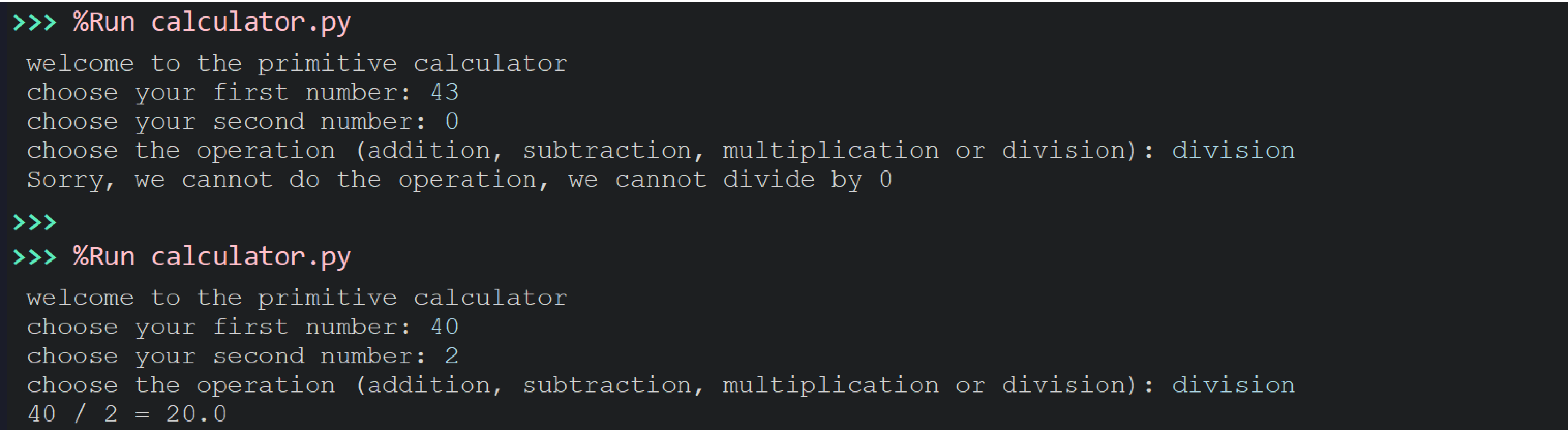
If the user asks for a division and y=0, tell the user that you cannot do the operation, we cannot divide by 0.

At the end print
“x [+, -, *, /] y : = result”

int() and str()



```
>>> %Run calculator.py
 welcome to the primitive calculator
 choose your first number: 4
 choose your second number: 6
 choose the operation (addition, subtraction, multiplication or division): addition
 4 + 6 = 10
>>> %Run calculator.py
 welcome to the primitive calculator
 choose your first number: 54
 choose your second number: 123
 choose the operation (addition, subtraction, multiplication or division): subtraction
 54 - 123 = -69
>>> %Run calculator.py
 welcome to the primitive calculator
 choose your first number: 5
 choose your second number: 3
 choose the operation (addition, subtraction, multiplication or division): multiplication
 5 * 3 = 15
```



```
>>> %Run calculator.py
 welcome to the primitive calculator
 choose your first number: 43
 choose your second number: 0
 choose the operation (addition, subtraction, multiplication or division): division
 Sorry, we cannot do the operation, we cannot divide by 0

>>>
>>> %Run calculator.py
 welcome to the primitive calculator
 choose your first number: 40
 choose your second number: 2
 choose the operation (addition, subtraction, multiplication or division): division
 40 / 2 = 20.0
```