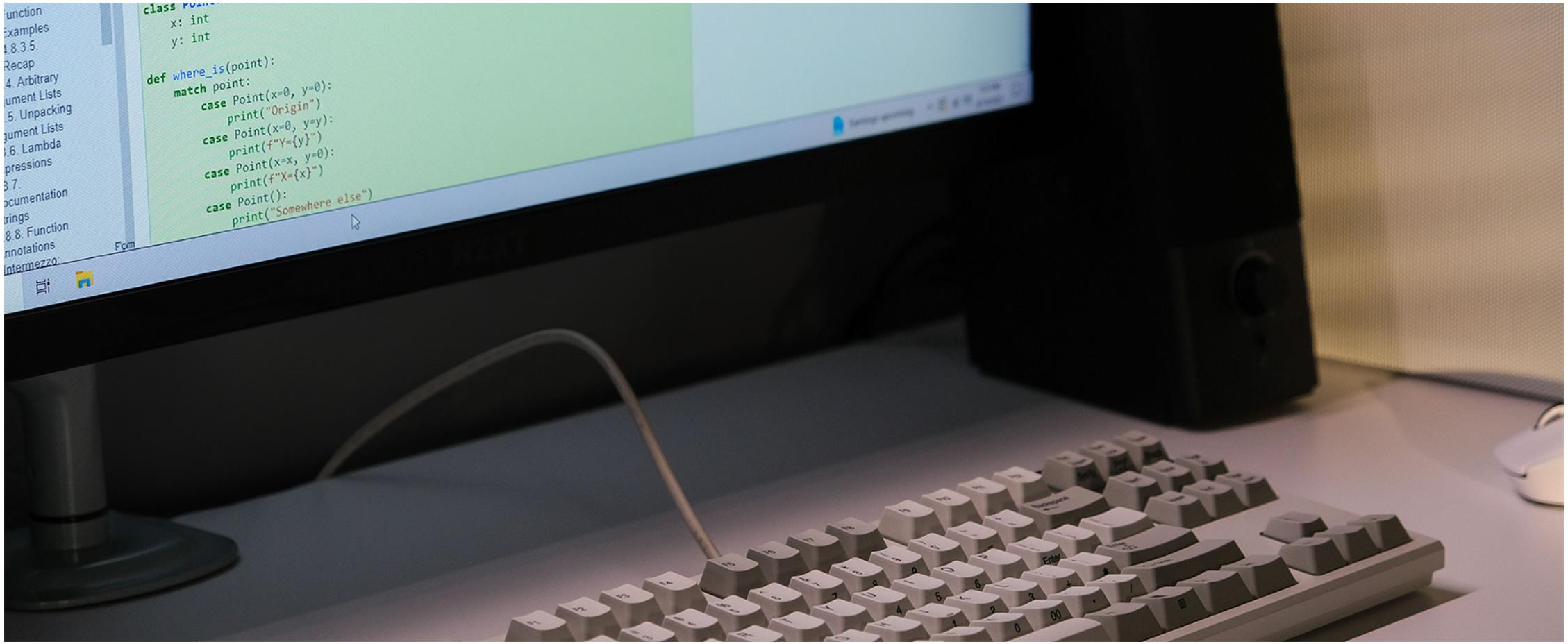
DS-011 Foundations for Data Science in Python

Module 1 - Statements



Credit: Larissa Chu, CC BY 4.0

1.1 Background

Learning Objectives

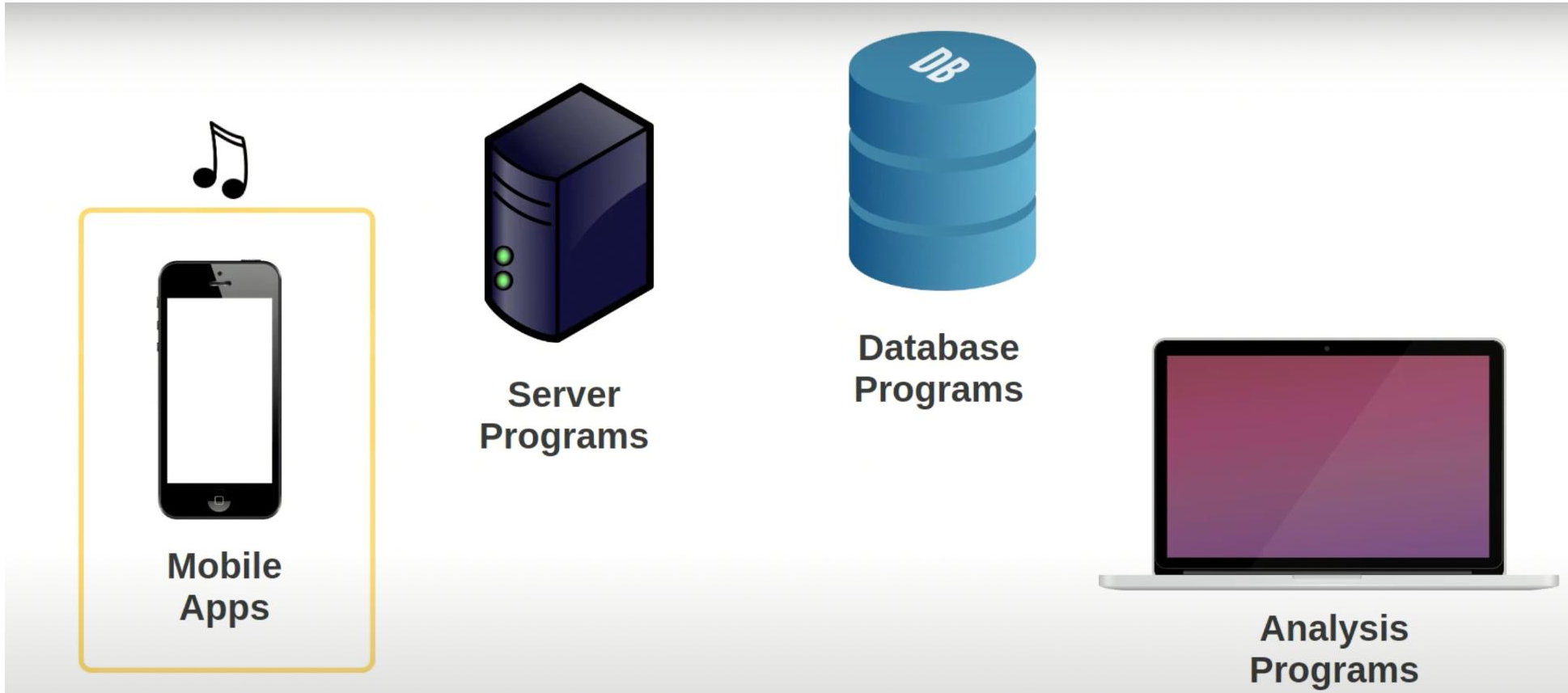
By the end of this section you should be able to

- Name two examples of computer programs in everyday life.
- Explain why Python is a good programming language to learn.

Computer programs

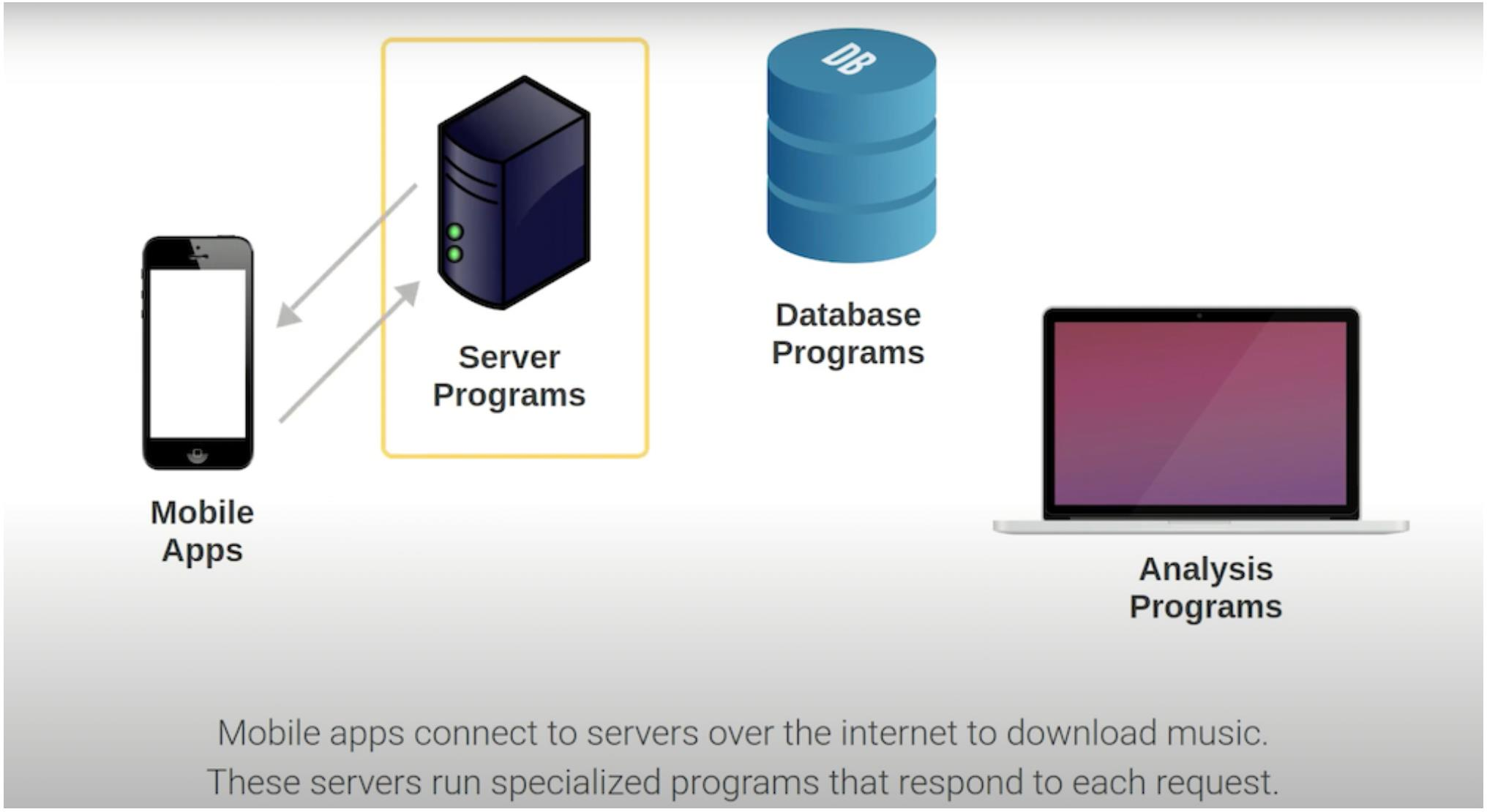
- A **computer** is an electronic device that stores and processes information. Examples of computers include smartphones, tablets, laptops, desktops, and servers.
- A **program** is a sequence of instructions that a computer can run. Programs help people accomplish everyday tasks, create new technology, and have fun.

Online music streaming

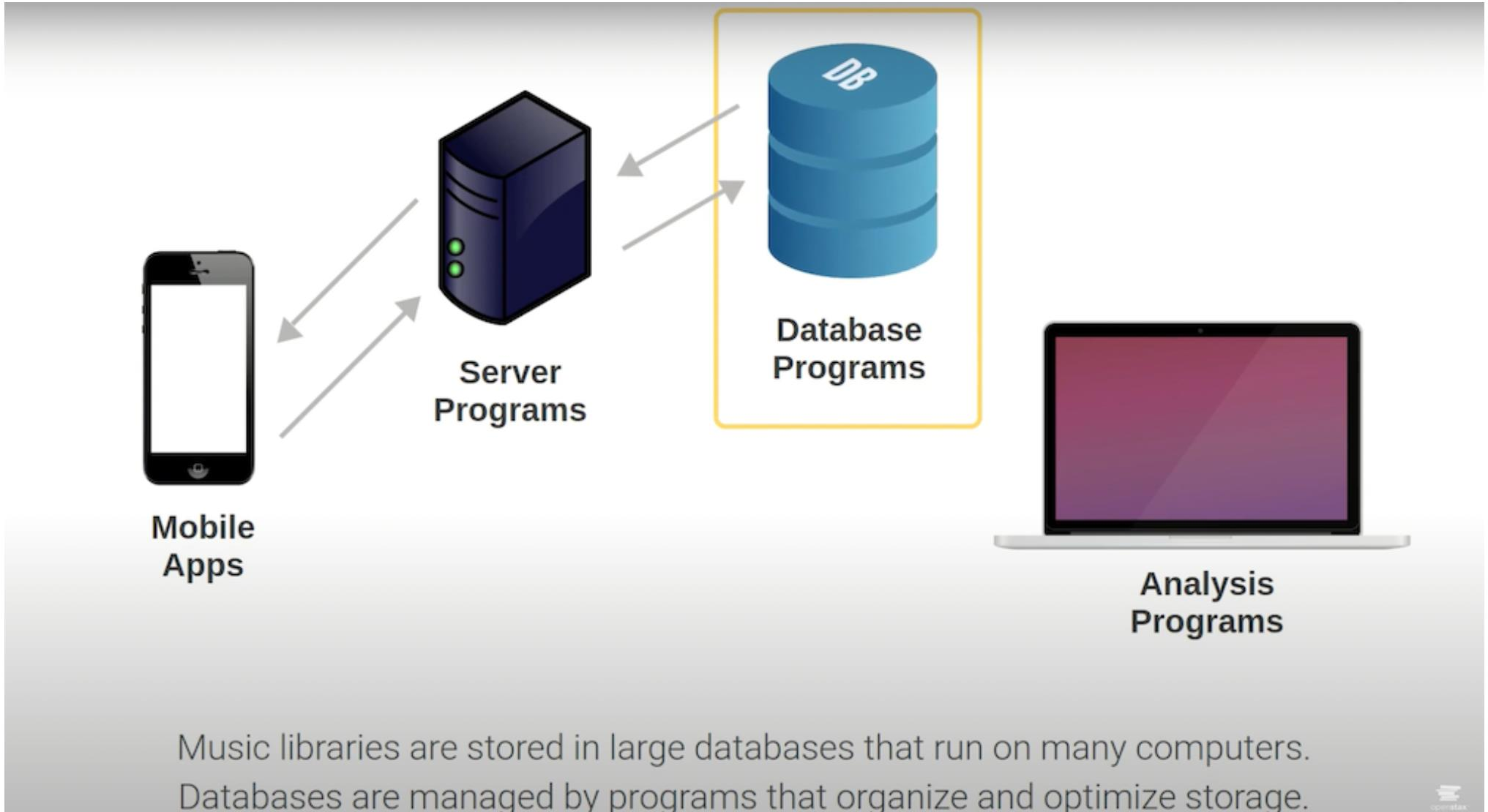


A common example of a computer program is an interactive mobile app. Some apps play music and allow people to manage their music collections.

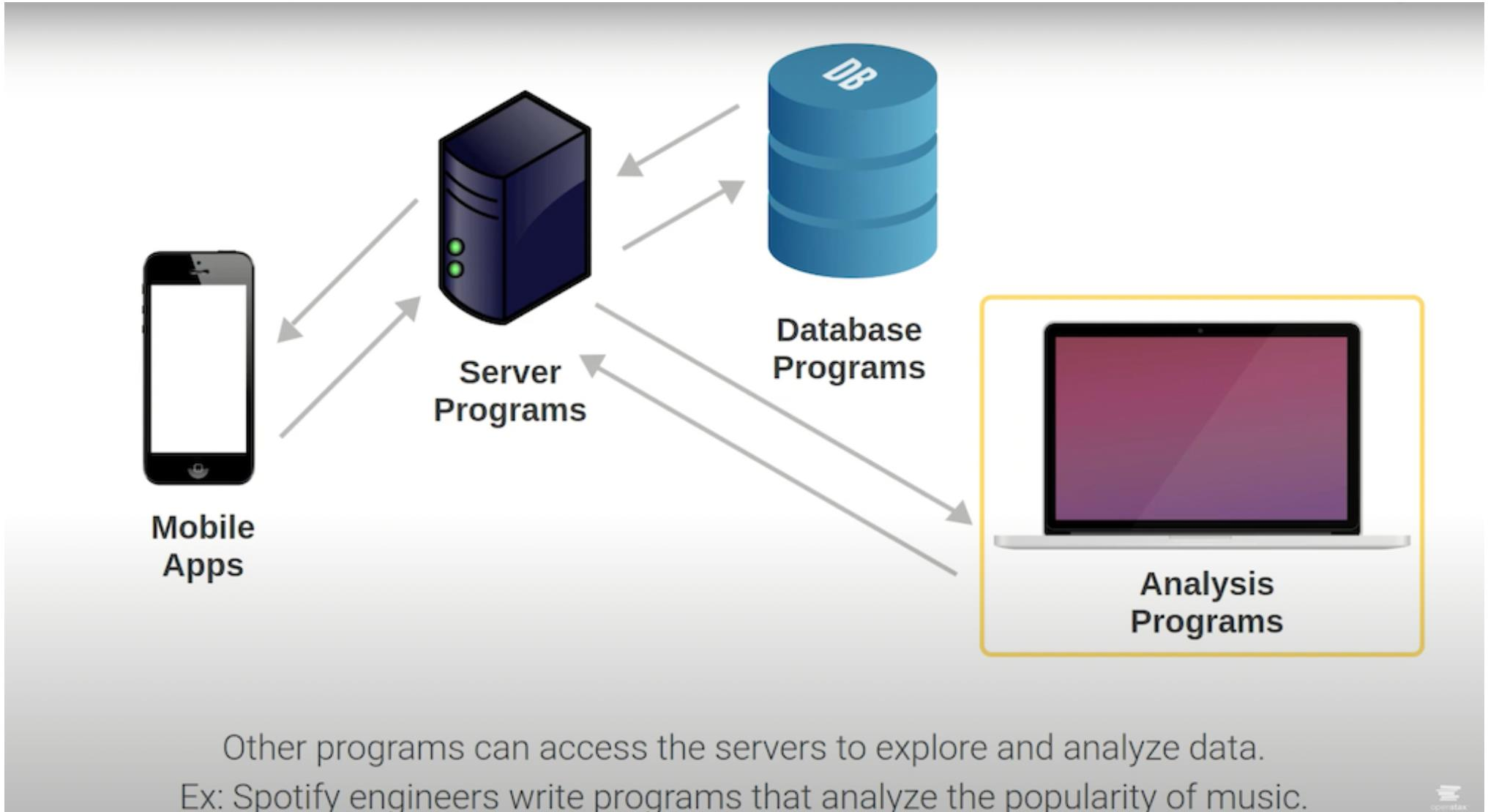
Online music streaming



Online music streaming



Online music streaming



The Python language

- Python is one of the top programming languages today. Leading tech giants like Google, Apple, NASA, Instagram, Pixar, and others use Python extensively.
- Many Python libraries exist for doing real work. A library is a collection of code that can be used in other programs. Ex: Pandas is a widely used library for data analysis.
- Python syntax is concise and straightforward. The syntax of a language defines how code must be structured including the keywords, symbols, and formatting used in programs.

Example 1.1: Hello world in Python and Java

The hello world program is only one line in Python:

```
print("Hello, World!")
```

In contrast, the hello world program is five lines in Java:

```
public class Hello {
  public static void main(String[] args) {
   System.out.println("Hello, World!");
  }
}
```

Counting lines in a file (Python code)

This program requires four lines of Python code.

```
data = open("file.txt")
lines = data.readlines()
count = len(lines)
print("The file has," count, "lines. ")
```

Counting lines in a file (Java code)

In java, this program would require many more lines.

```
import java.io.*;
import java.util.*;
public class Count{
  public static void main(String[] args){
    try{
        File file = new File("file.txt");
        Scanner in = new Scanner(file);
        int count = 0;
        while(in.hasNextLine()){
          in.nextLine();
          count++;
        }
    } catch(FileNotFoundException e)
        e.printStackTrace();
  }
}
```

Quiz

1. What type of program will this course explain how to write?

- [ ] a tool that summarizes an individual's music preferences
- [ ] a mobile app for managing and sharing playlists of songs
- [ ] a website that shows the top artists for the past five years

2. What type of program will this course explain how to write?

- [ ] a tool that summarizes an individual's music preferences
- [ ] a mobile app for managing and sharing playlists of songs
- [x] a website that shows the top artists for the past five years

3. Which of the following devices is an example of a computer?

- [ ] wired headphones that plug into a smartphone
- [ ] remote control that pauses or skips the current song
- [ ] wi-fi speaker that streams music from Amazon

4. Which of the following devices is an example of a computer?

- [ ] wired headphones that plug into a smartphone
- [ ] remote control that pauses or skips the current song
- [x] wi-fi speaker that streams music from Amazon

5. In general, Python programs are _____ than Java programs.

- [ ] faster
- [ ] longer
- [ ] shorter

6. In general, Python programs are _____ than Java programs.

- [ ] faster
- [ ] longer
- [x] shorter

7. In the example programs above, what syntax required by Java is not required by Python?

- [ ] semicolons
- [ ] parentheses
- [ ] quote marks

8. In the example programs above, what syntax required by Java is not required by Python?

- [ ] semicolons
- [ ] parentheses
- [x] quote marks

TRY IT

1.1 Favorite song

1.2 Input/output

Learning objectives

By the end of this section you should be able to:

- Display output using the `print()` function.
- Obtain user input using the `input()` function.

Basic output

- The `print()` function displays output to the user. **Output** is the information or result produced by a program. The `sep` and `end` options can be used to customize the output.
- Multiple values, separated by commas, can be printed in the same statement. By default, each value is separated by a space character in the output. The `sep` option can be used to change this behavior.
- By default, the `print()` function adds a newline character at the end of the output. A newline character tells the display to move to the next line. The `end` option can be used to continue printing on the same line.

Basic output

Example uses of `print()`

Code	Output
`print("Today is Monday.")` `print("I like string beans.")`	`Today is Monday.` `I like string beans.`
`print("Today", "is", "Monday")` `print("Today", "is", "Monday", sep="...")`	`Today is Monday` `Today...is...Monday`
`print("Today is Monday, ", end="")` `print("I like string beans.")`	`Today is Monday, I like string beans.`
`print("Today", "is", "Monday", sep="? ", end="!!")` `print("I like string beans.")`	`Today? is? Monday!!I like string beans.`

Quiz

1. Which line of code prints Hello world! as one line of output?

- `print(Hello world!)`
- `print("Hello", "world", "!")`
- `print("Hello world!")`

1. Which line of code prints Hello world! as one line of output?

- [ ] `print(Hello world!)`
- [ ] `print("Hello", "world", "!")`
- [x] `print("Hello world!")`

2. Which line of code prints Hello world! as one line of output?

1. `print("Hello")`
2. `print(" world!")`
3. `print("Hello")`
4. `print(" world!", end="")`
5. `print("Hello", end="")`
6. `print(" world!")`

3. What output is produced by the following statement?

```
print("555", "0123", sep="-")
```

1. 555 0123
2. 5550123-
3. 555-0123

3. What output is produced by the following statement?

```
print("555", "0123", sep="-")
```

- [ ] 1. 555 0123
- [ ] 2. 5550123-
- [x] 3. 555-0123

Basic input

- Computer programs often receive input from the user. Input is what a user enters into a program. An input statement, `variable = input("prompt")`, has three parts:
    - A **variable** refers to a value stored in memory. In the statement above, variable can be replaced with any name the programmer chooses.
    - The **input()** function reads one line of input from the user. A function is a named, reusable block of code that performs a task when called. The input is stored in the computer's memory and can be accessed later using the variable.
    - A **prompt** is a short message that indicates the program is waiting for input. In the statement above, "prompt" can be omitted or replaced with any message.

Quiz

4. Which line of code correctly obtains and stores user input?

1. `input()`
2. `today_is = input`
3. `today_is = input()`

4. Which line of code correctly obtains and stores user input?

- [ ] 1. `input()`
- [ ] 2. `today_is = input`
- [x] 3. `today_is = input()`

5. Someone named Sophia enters their name when prompted with

```
print("Please enter your name: ")
name = input()
```

6. What is displayed by `print("You entered:", name)`?

- [ ] 1. You entered: name
- [ ] 2. You entered: Sophia
- [ ] 3. You entered:, Sophia

5. Someone named Sophia enters their name when prompted with

```
print("Please enter your name: ")
name = input()
```

6. What is displayed by `print("You entered:", name)`?

- [ ] 1. You entered: name
- [x] 2. You entered: Sophia
- [ ] 3. You entered:, Sophia

7. What is the output if the user enters "six" as the input?

```
print("Please enter a number: ")
number = input()
print("Value =", number)
```

- 1. Value = six
- 2. Value = 6
- 3. Value = number

7. What is the output if the user enters "six" as the input?

```
print("Please enter a number: ")
number = input()
print("Value =", number)
```

- [x] 1. Value = six
- [ ] 2. Value = 6
- [ ] 3. Value = number

TRY IT

1.2 Frost poem

TRY IT

1.2 Name and likes

1.3 Variables

Learning objectives

By the end of this section you should be able to

- Assign variables and print variables.
- Explain rules for naming variables.

Assignment statement

- Variables allow programs to refer to values using names rather than memory locations. Ex: age refers to a person's age, and birth refers to a person's date of birth.
- A statement can set a variable to a value using the assignment operator (=). Note that this is different from the equal sign of mathematics. Ex: `age = 6` or `birth = "May 15"`. The left side of the assignment statement is a variable, and the right side is the value the variable is assigned.

Quiz

1. Which line of code correctly retrieves the value of the variable, city, after the following assignment?

```
city = "Chicago"
```

1. `print("In which city do you live?")`
2. `city = "London"`
3. `print("The city where you live is", city)`

1. Which line of code correctly retrieves the value of the variable, city, after the following assignment?

```
city = "Chicago"
```

- [ ] 1. `print("In which city do you live?")`
- [ ] 2. `city = "London"`
- [x] 3. `print("The city where you live is", city)`

2. Which program stores and retrieves a variable correctly?

- [ ] 1. `print("Total =", total)`

```
total = 6
```

- [ ] 2. `total = 6`

```
print("Total =", total)
```

- [ ] 3. `print("Total =", total)`

```
total = input()
```

2. Which program stores and retrieves a variable correctly?

⬜ 1. `print("Total =", total)`

```
total = 6
```

✅ 2. `total = 6`

```
print("Total =", total)
```

⬜ 3. `print("Total =", total)`

```
total = input()
```

3. Which is the assignment operator?

1. :
2. ==
3. =

3. Which is the assignment operator?

1. :
2. ==
   ✅ 3. =

4. **Which is a valid assignment?**

    1. `temperature = 98.5`
    2. `98.5 = temperature`
    3. `temperature - 23.2`

Variable naming rules

- A variable name can consist of letters, digits, and underscores and be of any length.
- The name cannot start with a digit. Ex: **101class** is invalid.
- Letter case matters. Ex: **Total** is different from **total**.
- Python's style guide recommends writing variable names in snake case, which is all lowercase with underscores in between each word, such as **first_name** or **total_price**.
- A name should be short and descriptive, so words are preferred over single characters in programs for readability. Ex: A variable named **count** indicates the variable's purpose better than a variable named **c**.
- Python has reserved words, known as keywords, which have special functions and cannot be used as names for variables (or other objects).

Python Keywords

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
asynch	elif	if	or	yield

Quiz

5. Which can be used as a variable name?

- [ ] 1. median
- [ ] 2. class
- [ ] 3. import

5. Which can be used as a variable name?

- [x] 1. median
- [ ] 2. class
- [ ] 3. import

6. Why is the name, 2nd_input, not a valid variable name?

- [ ] 1. contains an underscore
- [ ] 2. starts with a digit
- [ ] 3. is a keyword

6. Why is the name, 2nd_input, not a valid variable name?

- [ ] 1. contains an underscore
- [x] 2. starts with a digit
- [ ] 3. is a keyword

7. Which would be a good name for a variable storing a zip code?

- [ ] 1. z
- [ ] 2. var_2
- [ ] 3. zip_code

7. Which would be a good name for a variable storing a zip code?

- [ ] 1. z
- [ ] 2. var_2
- [x] 3. zip_code

8. Given the variable name, DogBreed, which improvement conforms to Python's style guide?

- [ ] 1. `dog_breed`
- [ ] 2. `dogBreed`
- [ ] 3. `dog-breed`

8. Given the variable name, DogBreed, which improvement conforms to Python's style guide?

- [x] 1. `dog_breed`
- [ ] 2. `dogBreed`
- [ ] 3. `dog-breed`

TRY IT

1.3 Final score

1.4 String basics

Learning objectives

By the end of this section you should be able to

- Use the built-in len() function to get a string's length.
- Concatenate string literals and variables using the + operator.

Quote marks

- A string is a sequence of characters enclosed by matching single (') or double (") quotes. Ex: `"Happy birthday!"` and `'21'` are both strings.
- To include a single quote (') in a string, enclose the string with matching double quotes ("). Ex: `"Won't this work?"` To include a double quote ("), enclose the string with matching single quotes ('). Ex: `'They said "Try it!", so I did'`.

String rules

Valid string	Invalid string
`"17" or '17'`	`17`
`"seventeen" or 'seventeen'`	`seventeen`
`"Where?" or 'Where?'`	`"Where?'`
`"I hope you aren't sad."`	`'I hope you aren't sad.'`
`'The teacher said "Correct!" '`	`"The teacher said "Correct!" "`

Quiz

1. Which of the following is a string?

- [ ] 1. Hello!
- [ ] 2. 29
- [ ] 3. "7 days"

1. Which of the following is a string?

- [ ] 1. Hello!
- [ ] 2. 29
- [x] 3. "7 days"

2. Which line of code assigns a string to the variable email?

1. `"fred78@gmail.com"`
2. `"email = fred78@gmail.com"`
3. `email = "fred78@gmail.com"`

2. Which line of code assigns a string to the variable email?

- [ ] 1. `"fred78@gmail.com"`
- [ ] 2. `"email = fred78@gmail.com"`
- [x] 3. `email = "fred78@gmail.com"`

3. Which is a valid string?

1. `I know you'll answer correctly!`
2. `'I know you'll answer correctly!'`
3. `"I know you'll answer correctly!"`

3. Which is a valid string?

- [ ] 1. `I know you'll answer correctly!`
- [ ] 2. `'I know you'll answer correctly!'`
- [x] 3. `"I know you'll answer correctly!"`

4. Which is a valid string?

1. `You say "Please" to be polite`
2. `"You say "Please" to be polite"`
3. `'You say "Please" to be polite'`

4. Which is a valid string?

- [ ] 1. `You say "Please" to be polite`
- [ ] 2. `"You say "Please" to be polite"`
- [x] 3. `'You say "Please" to be polite'`

len() function

A common operation on a string object is to get the string length, or the number of characters in the string. The `len()` function, when called on a string value, returns the string length.

Quiz

5. What is the return value for len("Hi Ali")?

- [ ] - 2
- [ ] - 5
- [ ] - 6

5. What is the return value for len("Hi Ali")?

- [ ] - 2
- [ ] - 5
- [x] - 6

6. What is the length of an empty string variable ("")?

- [ ] - undefined
- [ ] - 0
- [ ] - 2

6. What is the length of an empty string variable ("")?

- [ ] - undefined
- [x] - 0
- [ ] - 2

7. What is the output of the following code?

```
number = "12"
number_of_digits = len(number)
print("Number", number, "has", number_of_digits, "digits.")
```

- 1. Number 12 has 12 digits.
- 2. Number 12 has 2 digits.
- 3. Number 12 has number_of_digits digits.

7. What is the output of the following code?

```
number = "12"
number_of_digits = len(number)
print("Number", number, "has", number_of_digits, "digits.")
```

- [ ] 1. Number 12 has 12 digits.
- [x] 2. Number 12 has 2 digits.
- [ ] 3. Number 12 has number_of_digits digits.

Concatenation

Concatenation is an operation that combines two or more strings sequentially with the concatenation operator (+). Ex: `"A" + "part"` produces the string `"Apart"`.

Quiz

8. Which produces the string "10"?

- [ ] 1. `1 + 0`
- [ ] 2. `"1 + 0"`
- [ ] 3. `"1" + "0"`

8. Which produces the string "10"?

- [ ] 1. `1 + 0`
- [ ] 2. `"1 + 0"`
- [x] 3. `"1" + "0"`

9. Which produces the string "Awake"?

- [ ] 1. `"wake" + "A"`
- [ ] 2. `"A + wake"`
- [ ] 3. `"A" + "wake"`

9. Which produces the string "Awake"?

- [ ] 1. `"wake" + "A"`
- [ ] 2. `"A + wake"`
- [x] 3. `"A" + "wake"`

10. A user enters "red" after the following line of code.

```
color = input("What is your favorite color?")
```

Which produces the output "Your favorite color is red!"?

1. `print("Your favorite color is " + color + !)`
2. `print("Your favorite color is " + "color" + "!")`
3. `print("Your favorite color is " + color + "!")`

10. A user enters "red" after the following line of code.

```
color = input("What is your favorite color?")
```

Which produces the output "Your favorite color is red!"?

- [ ] 1. `print("Your favorite color is " + color + !)`
- [ ] 2. `print("Your favorite color is " + "color" + "!")`
- [x] 3. `print("Your favorite color is " + color + "!")`

11. Which of the following assigns "one-sided" to the variable holiday?

1. `holiday = "one" + "sided"`
2. `holiday = one-sided`
3. `holiday = "one-" + "sided"`

11. Which of the following assigns "one-sided" to the variable holiday?

- [ ] 1. `holiday = "one" + "sided"`
- [ ] 2. `holiday = one-sided`
- [x] 3. `holiday = "one-" + "sided"`

TRY IT

1.4 Name length

TRY IT

1.4 Punctuation

1.5 Number basics

Learning objectives

By the end of this section you should be able to

- Use arithmetic operators to perform calculations.
- Explain the precedence of arithmetic operators.

Numeric data types

- Python supports two basic number formats, integer and floating-point.
- An integer represents a whole number, and a floating-point format represents a decimal number.
- The format a language uses to represent data is called a **data type**.
- In addition to integer and floating-point types, programming languages typically have a string type for representing text.

Quiz

1. What is the output of the following code?

```
x, y, s = 1, 2.0, "32"

print(x, type(x))
```

- 1 'int'.
- 1.0 <class 'float'>.
- 1 <class 'int'>.

1. What is the output of the following code?

```
x, y, s = 1, 2.0, "32"

print(x, type(x))
```

- [ ] 1 'int'.
- [ ] 1.0 <class 'float'>.
- [x] 1 <class 'int'>.

2. What is the output of the following code?

```
x, y, s = 1, 2.0, "32"

print(y, type(y))
```

2.0 <class 'int'>

2.0 <class 'float'>

2 <class 'int'>

2. What is the output of the following code?

```
x, y, s = 1, 2.0, "32"

print(y, type(y))
```

- [ ] 2.0 <class 'int'>
- [x] 2.0 <class 'float'>
- [ ] 2 <class 'int'>

3. What is the type of the following value?

```
"12.0"
```

1. string
2. int
3. float

3. What is the type of the following value?

```
"12.0"
```

- [x] 1. string
- [ ] 2. int
- [ ] 3. float

Basic arithmetic

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, and division.

Four basic arithmetic operators exist in Python:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)

Quiz

4. Given the following lines of code, what is the output of the code?

```
x, y, z = 7, 20, 2

c = 0
c = x - z
c = c + 1
print(c)
```

1

5

6

5. What is the value of a?

```
x, y, z = 7, 20, 2

a = 3.5 - 1.5
```

2

2.0

2.5

6. What is the printed output?

```
x, y, z = 7, 20, 2

print(x / z)
```

3

3.0

3.5

7. What is the printed output?

```
x, y, z = 7, 20, 2

print(y / z)
```

0

10

10.0

8. What is the printed output?

```
x, y, z = 7, 20, 2

print(z * 1.5)
```

2

3

3.0

Operator precedence

When a calculation has multiple operators, each operator is evaluated in order of **precedence**. Ex: 1 + 2 * 3 is 7 because multiplication takes precedence over addition. However, (1 + 2) * 3 is 9 because parentheses take precedence over multiplication.

Operator precedence

Operator	Description	Example	Result
`()`	Parentheses	`(1 + 2) * 3`	`9`
`**`	Exponentiation	`2 ** 4`	`16`
`+, -`	Positive, negative	`-math.pi`	`-3.14159`
`*, /`	Multiplication, division	`2 * 3`	`6`
`+, -`	Addition, subtraction	`1 + 2`	`3`

Quiz

9. What is the value of `4 * 3 ** 2 + 1`?

37

40

145

10. Which part of `(1 + 3) ** 2 / 4` evaluates first?

4 ** 2

1 + 3

2 / 4

11. What is the value of `-4 ** 2`?

-16

16

12. How many operators are in the following statement?

```
result = -2 ** 3
```

1

2

3

TRY IT

1.5 Values and types

TRY IT

1.5 Meters to feet

1.6 Error messages

Learning objectives

By the end of this section you should be able to

- Identify the error type and line number in error messages.
- Correct syntax errors, name errors, and indentation errors.

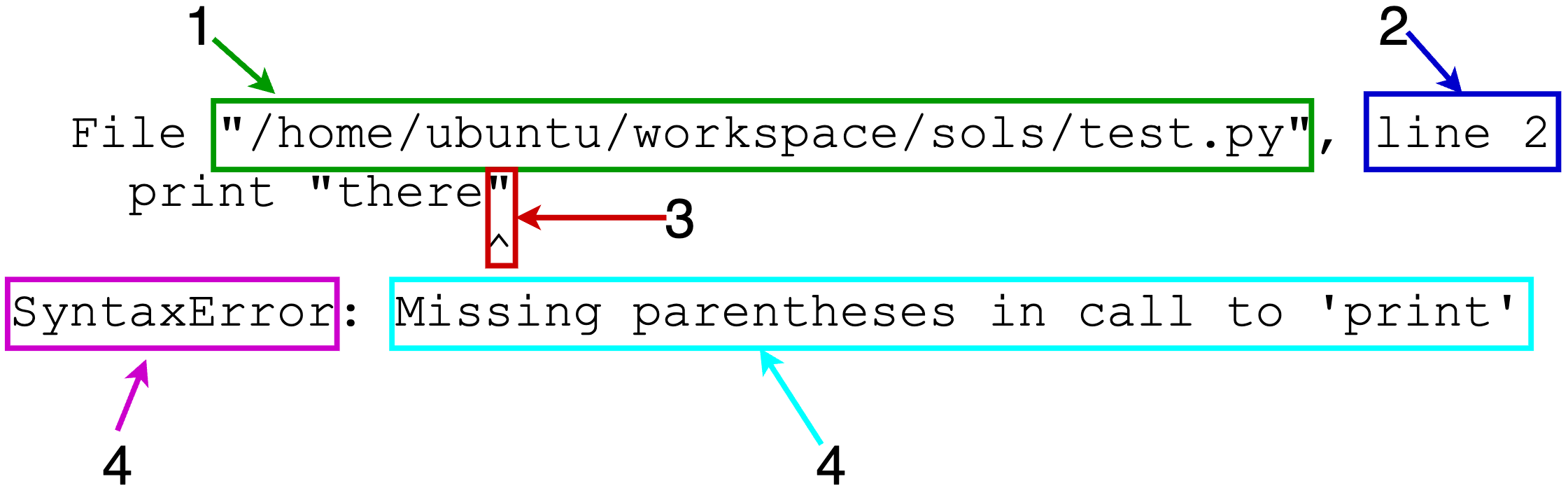
How to read errors

When an error occurs, Python displays a message with the following information:

- The line number of the error.
- The type of error (Ex: SyntaxError).
- Additional details about the error.

```
print "Hello!"
```

```
Traceback (most recent call last):
  File "/home/student/Desktop/example.py", line 1
    print "Hello"
                ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("Hello")?
```


1
2
File "/home/ubuntu/workspace/sols/test.py", line 2
print "there"
^
3
SyntaxError: Missing parentheses in call to 'print'
4
4

Quiz

1. Given the following error message:

```
Traceback (most recent call last):
  File "/home/student/Desktop/example.py", line 2
    print "test"
          ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("test")?
```

2. What is the filename of the program?

1. Desktop
2. example.py
3. test

3. Given the following error message:

```
Traceback (most recent call last):
  File "/home/student/Desktop/example.py", line 2
    print "test"
          ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("test")?
```

4. On which line was the error found?

1

2

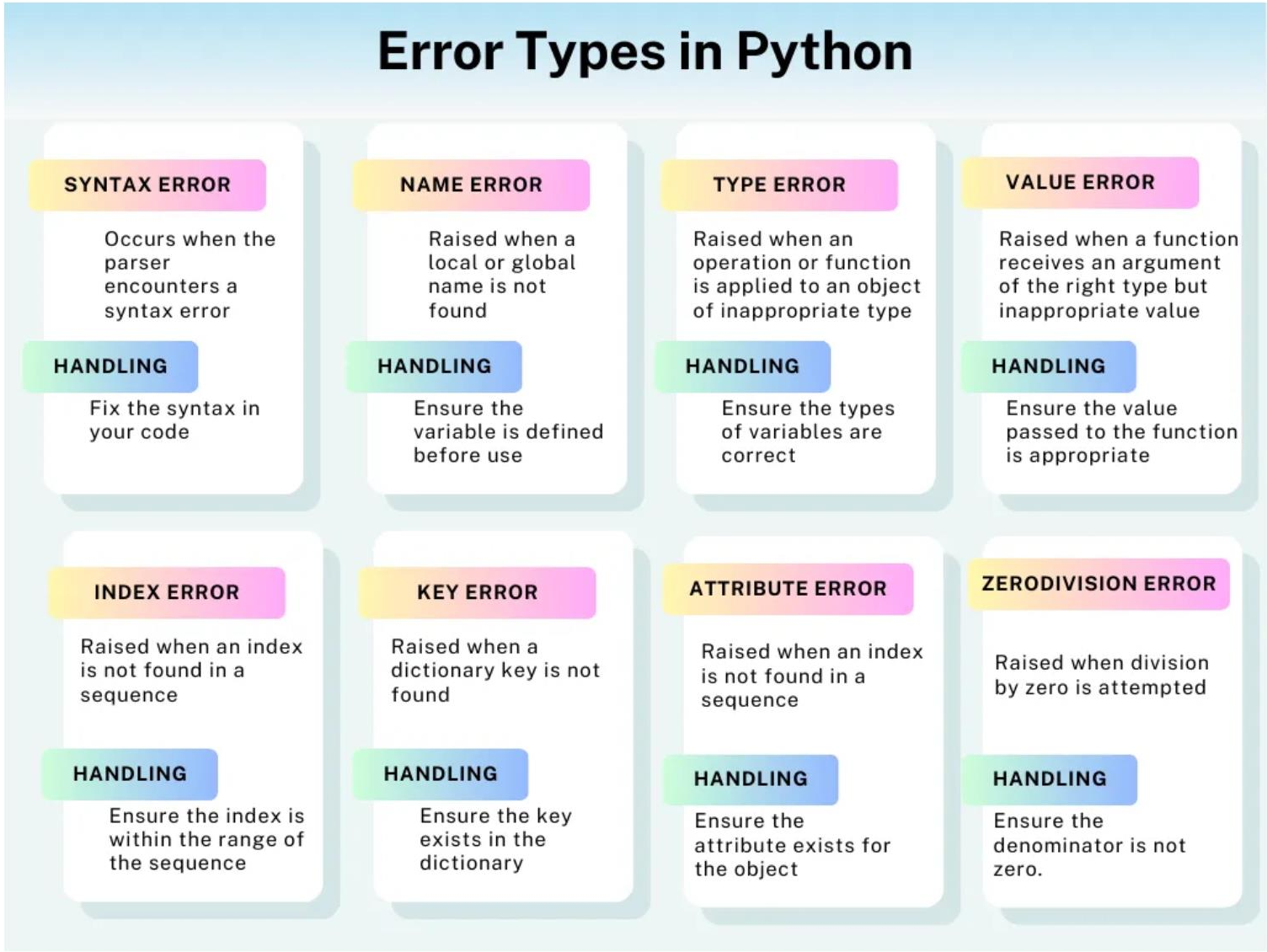
3

5. Given the following error message:

```
Traceback (most recent call last):
  File "/home/student/Desktop/example.py", line 2
    print "test"
          ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("test")?
```

6. What type of error was found?

1. missing parentheses
2. SyntaxError
3. traceback


Error Types in Python
SYNTAX ERROR
Occurs when the parser encounters a syntax error
HANDLING
Fix the syntax in your code
NAME ERROR
Raised when a local or global name is not found
HANDLING
Ensure the variable is defined before use
TYPE ERROR
Raised when an operation or function is applied to an object of inappropriate type
HANDLING
Ensure the types of variables are correct
VALUE ERROR
Raised when a function receives an argument of the right type but inappropriate value
HANDLING
Ensure the value passed to the function is appropriate
INDEX ERROR
Raised when an index is not found in a sequence
HANDLING
Ensure the index is within the range of the sequence
KEY ERROR
Raised when a dictionary key is not found
HANDLING
Ensure the key exists in the dictionary
ATTRIBUTE ERROR
Raised when an index is not found in a sequence
HANDLING
Ensure the attribute exists for the object
ZERODIVISION ERROR
Raised when division by zero is attempted
HANDLING
Ensure the denominator is not zero.

Common types of errors

Mistake	Error message	Explanation
`print("Have a nice day!"`	`SyntaxError: unexpected EOF while parsing`	The closing parenthesis is missing. Python is surprised to reach the end of file (EOF) before this line is complete.
`word = input("Type a word: )`	`SyntaxError: EOL while scanning string literal`	The closing quote marks are missing. As a result, the string does not terminate before the end of line (EOL).
`print("You typed:", wird)`	`NameError: name 'wird' is not defined`	The spelling of `word` is incorrect. The programmer accidentally typed the wrong key.
`prints("You typed:", word)`	`NameError: name 'prints' is not defined`	The spelling of `print` is incorrect. The programmer accidentally typed an extra letter.
`  print("Hello")`	`IndentationError: unexpected indent`	The programmer accidentally typed a space at the start of the line.
`    print("Goodbye")`	`IndentationError: unexpected indent`	The programmer accidentally pressed the Tab key at the start of the line.

Quiz

7. What type of error will occur in the below program?

```
print("Breakfast options:")
 print("A. Cereal")
 print("B. Eggs")
 print("C. Yogurt")
choice = input("What would you like? ")
```

1. IndentationError
2. NameError
3. SyntaxError

7. What type of error will occur in the below program?

```
print("Breakfast options:")
 print("A. Cereal")
 print("B. Eggs")
 print("C. Yogurt")
choice = input("What would you like? ")
```

- [x] 1. IndentationError
- [ ] 2. NameError
- [ ] 3. SyntaxError

8. What type of error will occur in the below program?

```
birth = input("Enter your birth date: )
print("Happy birthday on ", birth)
```

1. IndentationError
2. NameError
3. SyntaxError

8. What type of error will occur in the below program?

```
birth = input("Enter your birth date: )
print("Happy birthday on ", birth)
```

- 1. IndentationError
- 2. NameError
- ✅ 3. SyntaxError

9. What type of error will occur in the below program?

```
print("Breakfast options:")
print(" A. Cereal")
print(" B. Eggs")
print(" C. Yogurt")
choice = intput("What would you like? ")
```

1. IndentationError
2. NameError
3. SyntaxError

9. What type of error will occur in the below program?

```
print("Breakfast options:")
print(" A. Cereal")
print(" B. Eggs")
print(" C. Yogurt")
choice = intput("What would you like? ")
```

- [ ] 1. IndentationError
- [x] 2. NameError
- [ ] 3. SyntaxError

TRY IT

1.6 Three errors

TRY IT

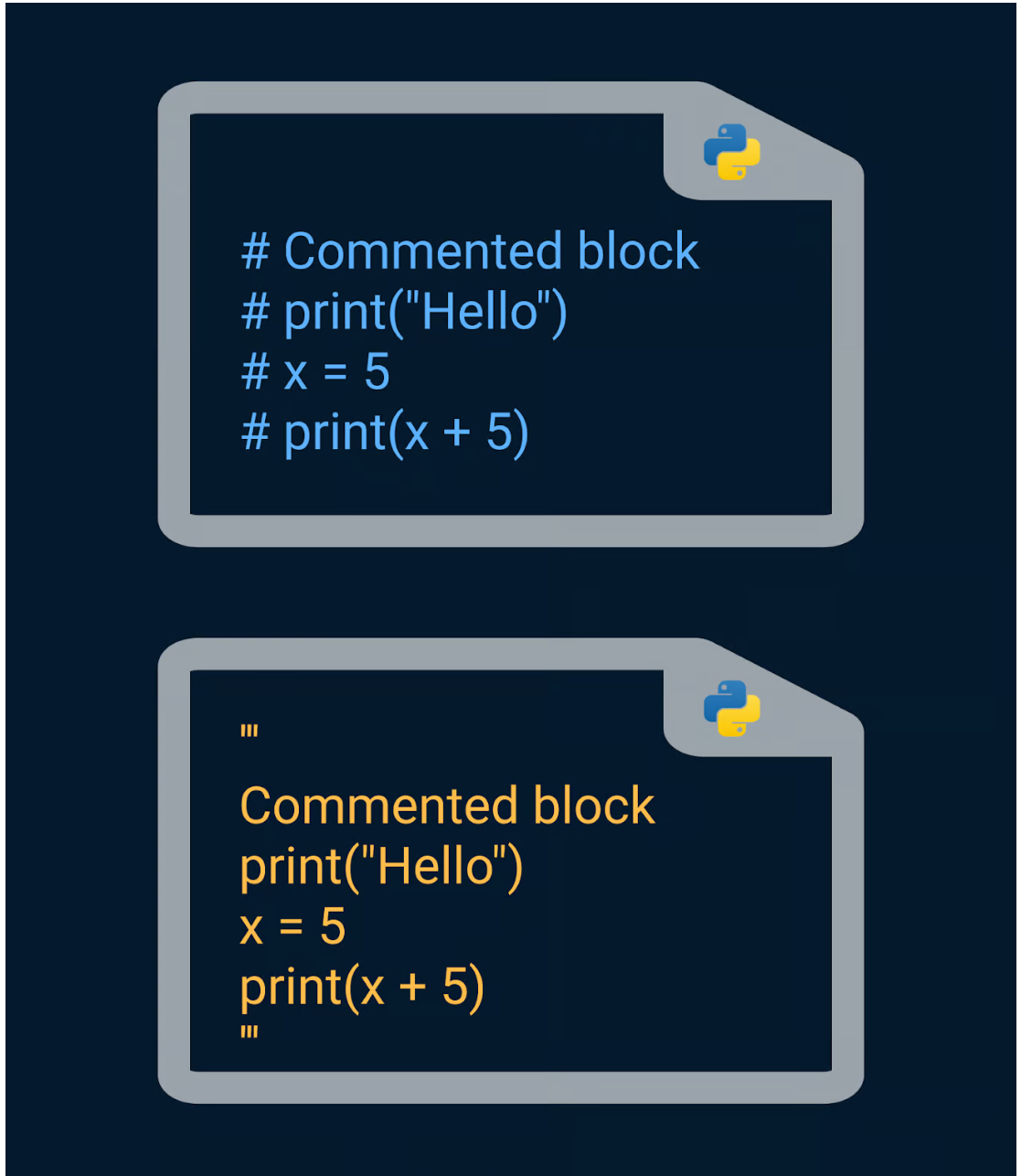
1.6 Wrong symbols

1.7 Comments

Learning objectives

By the end of this section you should be able to

- Write concise, meaningful comments that explain intended functionality of the code.
- Write a docstring (more verbose comment) that describes the program functionality.


Commented block
print("Hello")
x = 5
print(x + 5)
'''
Commented block
print("Hello")
x = 5
print(x + 5)
'''

The hash character

- **Comments** are short phrases that explain what the code is doing.
  - Ex: Lines 1, 8, and 10 in the following program contain comments.
- Each comment begins with a hash character (#). All text from the hash character to the end of the line is ignored when running the program.
- In contrast, hash characters inside of strings are treated as regular text.
  - Ex: The string "Item #1: " does not contain a comment.

The hash character

When writing comments:

- The # character should be followed by a single space.
  - Ex: # End of menu is easier to read than #End of menu.
- Comments should explain the purpose of the code, not just repeat the code itself.
  - Ex: # Get the user's preferences is more descriptive than # Input item1 and item2.

Example 1.2 Program with three comments

```
# Display the menu options
print("Lunch Menu")
print("----------")
print("Burrito")
print("Enchilada")
print("Taco")
print("Salad")
print() # End of menu

# Get the user's preferences
item1 = input("Item #1: ")
item2 = input("Item #2: ")
```

Quiz

1. **The main purpose of writing comments is to _____.**
    1. avoid writing syntax errors
    2. explain what the code does
    3. make the code run faster

2. Which symbol is used for comments in Python?

1. #
2. /*
3. //

3. Which comment is formatted correctly?

- 0 spaces:

  #Get the user input

- 1 space:

  # Get the user input

- 2 spaces:

  # Get the user input

Code quality

- The blank lines and comments show the overall structure of the program above. It has two parts:
    i. display the menu options, and
    ii. get the user's preferences.
- Programmers spend more time reading code than writing code. Therefore, making code easier for others to read and understand is important. Two ways to improve code quality include:
    - Separate each part (lines that have a similar purpose) with a blank line.
    - Write a comment before each part. Not every line needs a comment.

Comments in a program

Program without comments

```
adj1 = input("Adjective: ")
adj2 = input("Adjective: ")
noun1 = input("Noun: ")
noun2 = input("Noun: ")
print()
print("A vacation is when you take")
print("a trip to some", adj1, "place")
print("with your", adj2, "family.")
print("Usually you go to some place")
print("that is near a/an", noun1)
print("or up on a/an", noun2 + ".")
```

Program with comments

```
# Input the words missing from the story
adj1 = input("Adjective: ")
adj2 = input("Adjective: ")
noun1 = input("Noun: ")
noun2 = input("Noun: ")

# Blank line to separate input from output
print()

# Print the story with the words filled in
print("A vacation is when you take")
print("a trip to some", adj1, "place")
print("with your", adj2, "family.")
print("Usually you go to some place")
print("that is near a/an", noun1)
print("or up on a/an", noun2 + ".")
```

Adding comments before each part makes the code easier to understand. The comments explain the purpose of each part.

Quiz

4. Which comment is most useful for the following code?

```
print("You said:", adj1 + " " + noun1)
```

- # Append adj1 and noun1
- # Print out a bunch of stuff
- # Show the resulting phrase

5. Where should a blank line be inserted?

```
1 | name = input("Whose birthday is today? ")
2 | print("Happy birthday to", name)
3 | print("Everyone cheer for", name)
```

1. After line 1
2. After line 2
3. After line 3

6. To temporarily prevent a line from being run, one might . . .

1. introduce a syntax error in the line.
2. remove the line from the program.
3. insert a # at the beginning of the line.

Documentation

- Python programs may optionally begin with a string known as a docstring.
- A **docstring** is documentation written for others who will use the program but not necessarily read the source code.
- Most of the official documentation at docs.python.org is generated from docstrings.
- Documentation can be long, so docstrings are written as multi-line strings (""").
- Common elements of a docstring include a one-line summary, a blank line, and a longer description.

Vacations docstring

```
"""Vacations Madlib.

This program asks the user for two adjectives and two nouns,
which are then used to print a funny story about a vacation.
"""
```

Summary line

Blank line

Description

```
# Input the words missing from the story
adj1 = input("Adjective: ")
adj2 = input("Adjective: ")
noun1 = input("Noun: ")
noun2 = input("Noun: ")

...
```

Docstrings typically have three parts: a summary line, a blank line, and a longer description. The summary should be on the same line as the opening quote marks.

Quiz

7. The main purpose of writing docstrings is to . . .

1. summarize the program's purpose or usage.
2. explain how each part of the code works.
3. maintain a list of ideas for new features.

8. Which of the following is NOT a docstring?

1. """Vacations Madlib."""
2. """Vacations Madlib.
   This program asks the user for two adjectives
   and two nouns, which are then used to print
   a funny story about a vacation.
   """
3. # Vacations Madlib.
   #
   # This program asks the user for two adjectives
   # and two nouns, which are then used to print
   # a funny story about a vacation.

9. Which docstring is most useful for this program?

1. """Vacations Madlib."""
2. """Vacations Madlib.
   This program asks the user for two adjectives and two nouns, which are then used to print a funny story about a vacation.
   """
3. """Vacations Madlib.
   This program asks the user for two adjectives and two nouns, which are then used to print a funny story about a vacation. The code uses four variables to store the user input: two for the adjectives, and two for the nouns. The output is displayed on seven lines, beginning with a blank line after the input.
   """

TRY IT

1.7 Whose birthday

TRY IT

1.7 Gravity calculation

1.8 Why Python?

Learning objectives

By the end of this section you should be able to

- Name two historical facts about how Python was first created.
- Describe two ways Python is considered a popular language.

Historical background

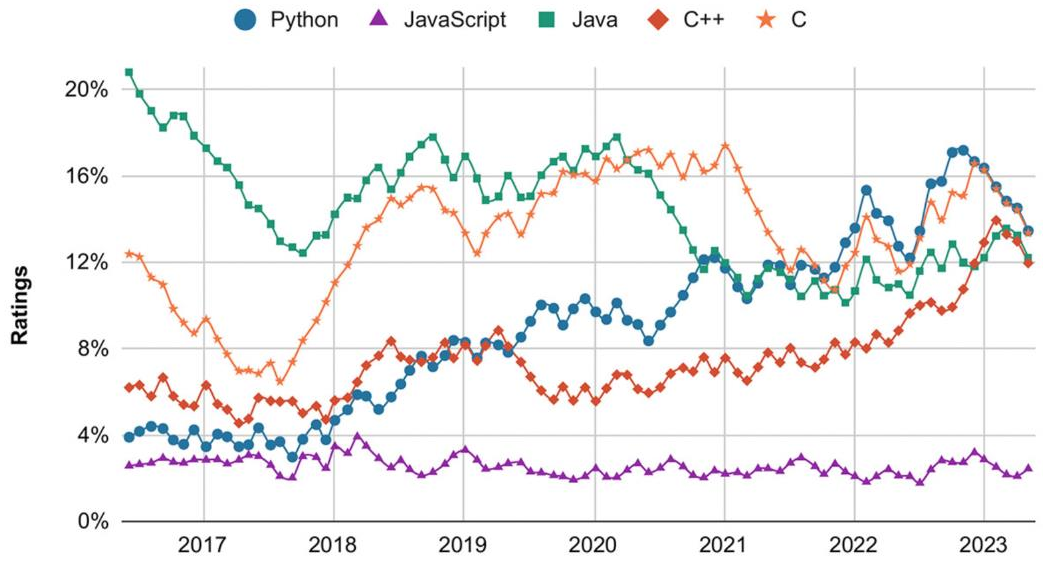
- Python has an interesting history. In 1982, Guido van Rossum, the creator of Python, started working at CWI, a Dutch national research institute. He joined a team that was designing a new programming language, named ABC, for teaching and prototyping. ABC's simplicity was ideal for beginners, but the language lacked features required to write advanced programs.
- Several years later, van Rossum joined a different team at CWI working on an operating system. The team needed an easier way to write programs for monitoring computers and analyzing data. Languages common in the 1980's were (and still are) difficult to use for these kinds of programs. van Rossum envisioned a new language that would have a simple syntax, like ABC, but also provide advanced features that professionals would need.

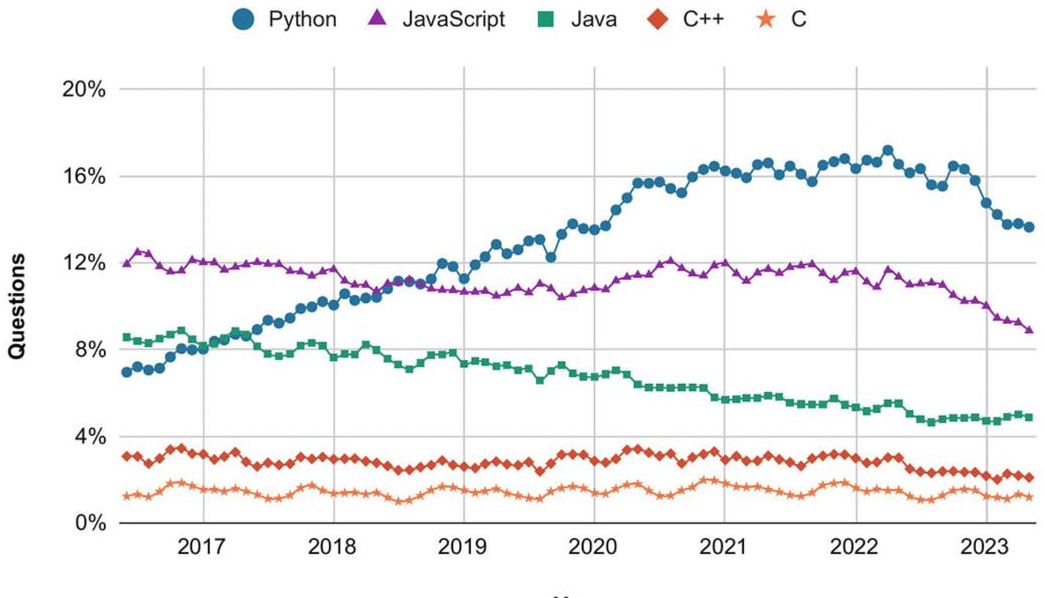
Historical background

- At first, van Rossum started working on this new language as a hobby during his free time. He named the language Python because he was a fan of the British comedy group Monty Python. Over the next year, he and his colleagues successfully used Python many times for real work. van Rossum eventually decided to share Python with the broader programming community online. He freely shared Python's entire source code so that anyone could write and run Python programs.
- Python's first release, known as Version 0.9.0, appeared in 1991, about six years after C++ and four years before Java. van Rossum's decisions to make the language simple yet advanced, suitable for everyday tasks, and freely available online contributed to Python's long-term success.

Popularity of Python

- Millions of programmers around the world use Python for all kinds of projects.
- Hundreds of thousands of Python libraries have been released as open source.
- In October 2021, Python became the #1 language on the TIOBE index. Since 2001, TIOBE has tracked the popularity of programming languages and posted the results online.





Quiz

1. The Python programming language was named after a _____.

- [ ] `British comedy group
- [ ] `Dutch programmer
- [ ] `non-venomous snake

2. Which programming language came first?

- [ ] `Java
- [ ] `Python

3. Which sentence best describes the beginning of Python?

- [ ] `CWI hired Guido van Rossum to design a new programming language to compete with C++.
- [ ] `Python started as a hobby and became open source after years of development.
- [ ] `van Rossum posted Python's source code online after working on the language for one year.

4. According to the TIOBE Index, which two languages were most popular from 2001 to 2021?

- [ ] `C and C++
- [ ] `Java and C
- [ ] `Python and JavaScript

5. In what year did Python become the most asked about language on Stack Overflow?

- [ ] `2018
- [ ] `2019
- [ ] `2020

6. **How long has TIOBE been tracking programming language popularity?**

- [ ] `since 1991
- [ ] `since 2001
- [ ] `since 2015

1.9 Summary

This module introduced the basics of programming in Python, including:

- `print()` and `input()`.
- Variables and assignment.
- Strings, integers, and floats.
- Arithmetic, concatenation.
- Common error messages.
- Comments and docstrings.

At this point, you should be able to write programs that ask for input, perform simple calculations, and output the results. The programming practice below ties together most topics presented in the module.

TRY IT

1.9 Fun facts

TRY IT

1.9 Mad lib