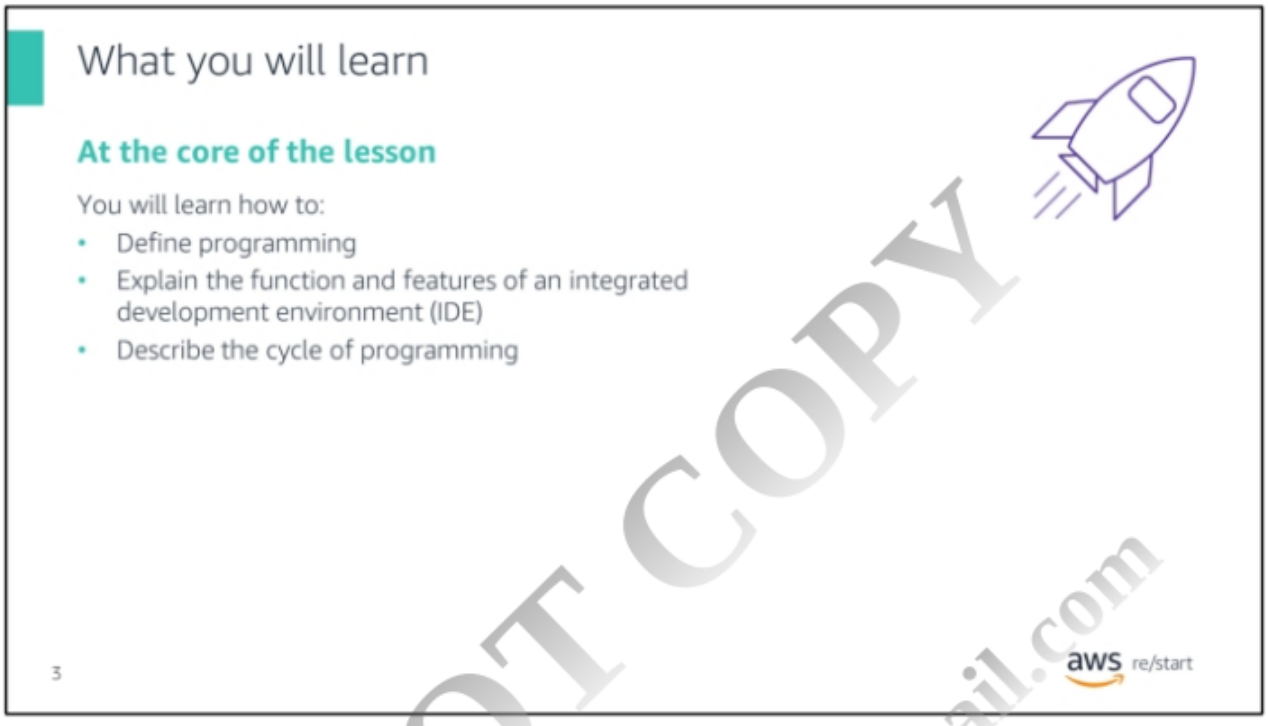
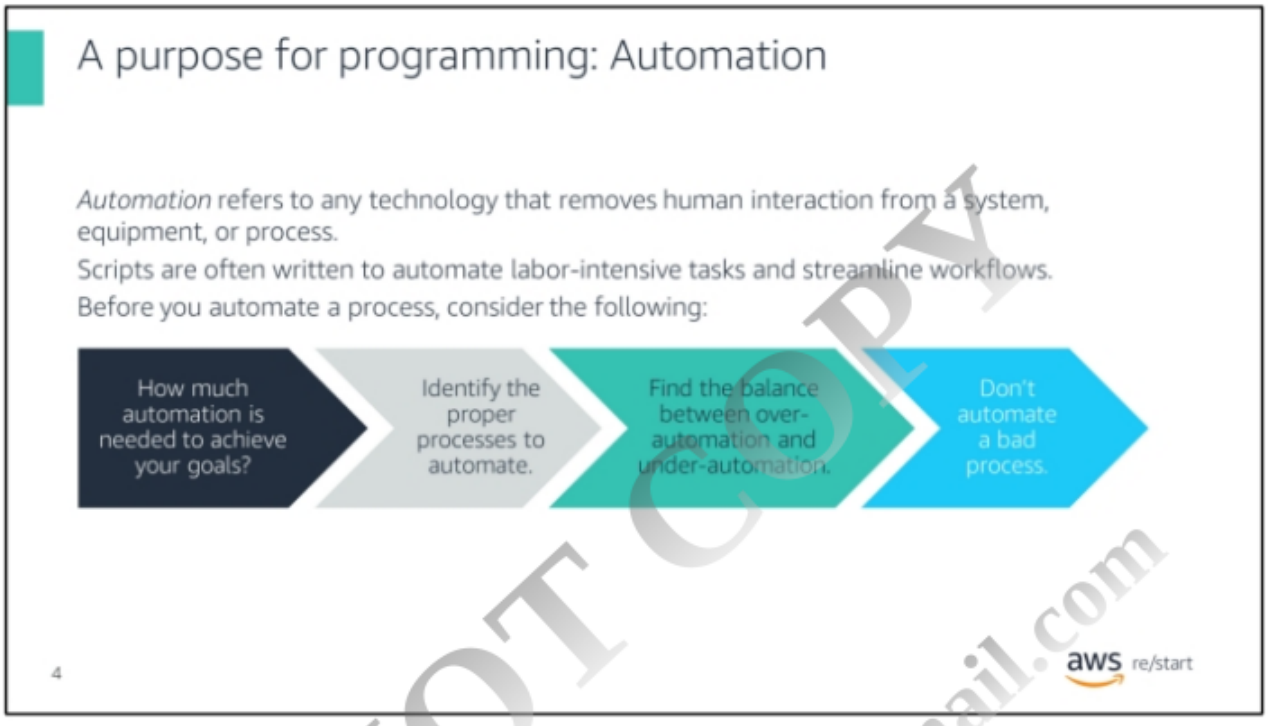
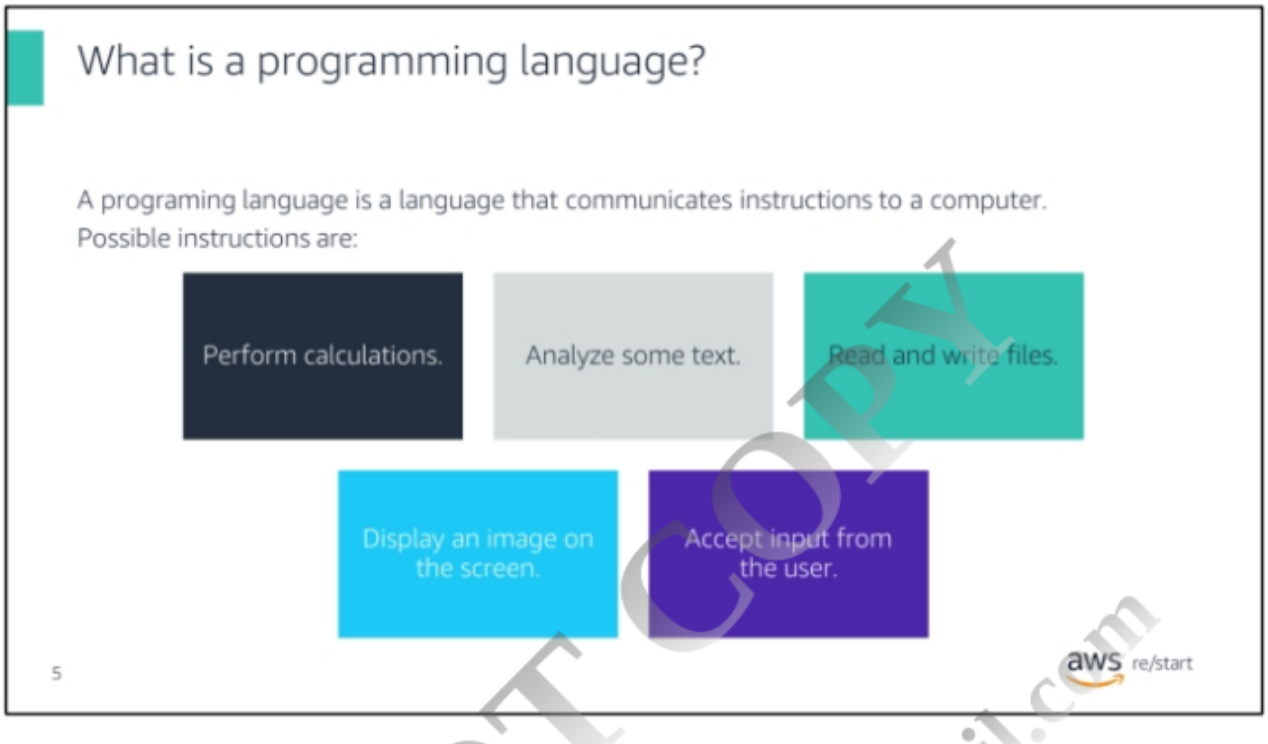
Welcome to Introduction to Programming.

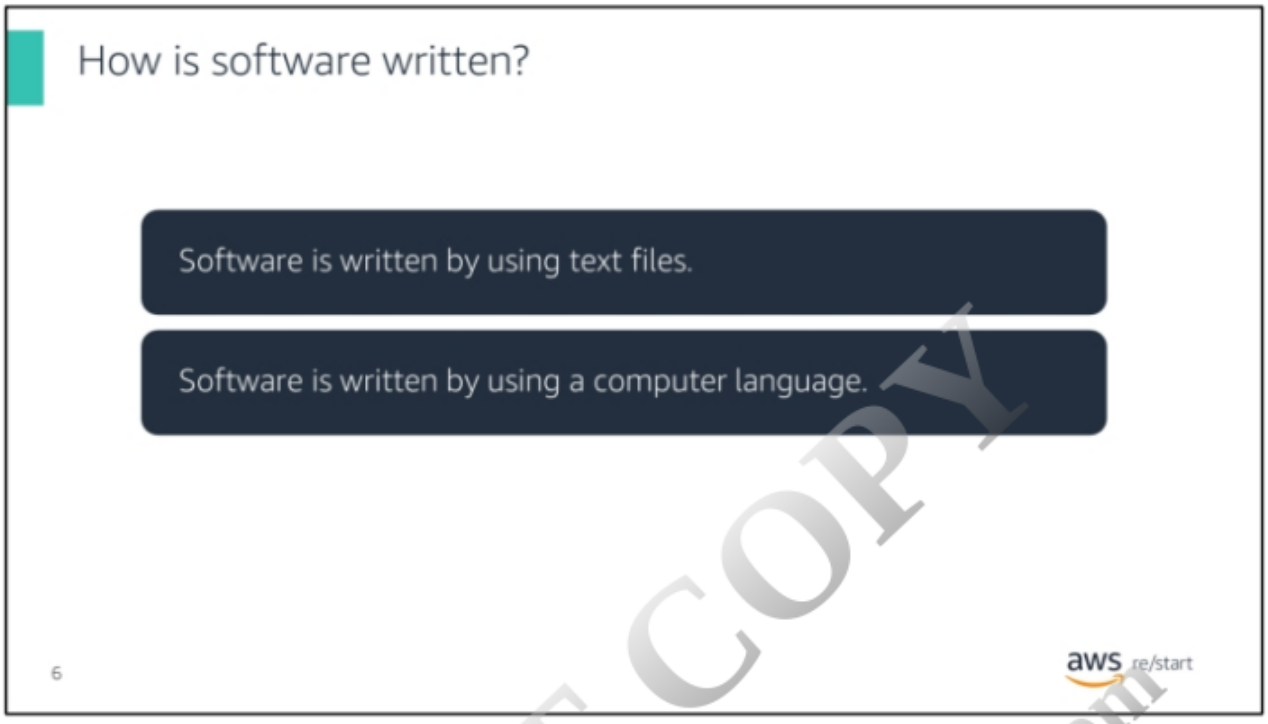


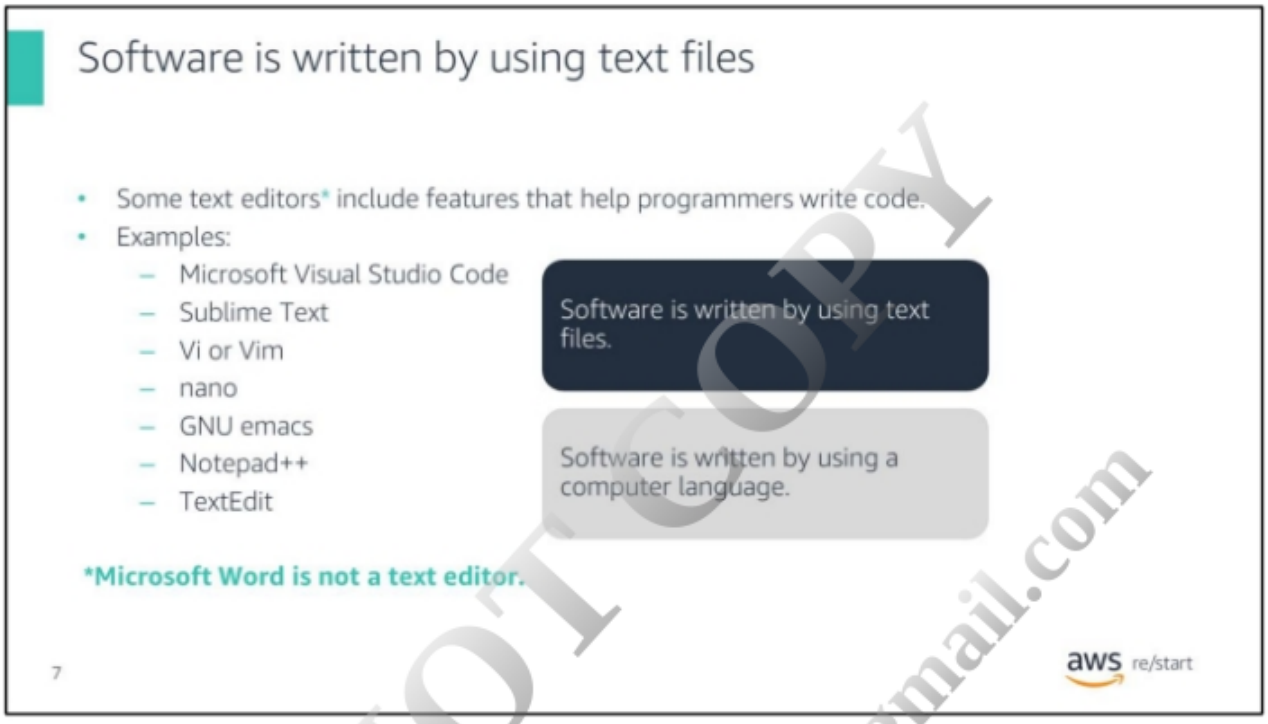
In this lesson, you will learn how to:

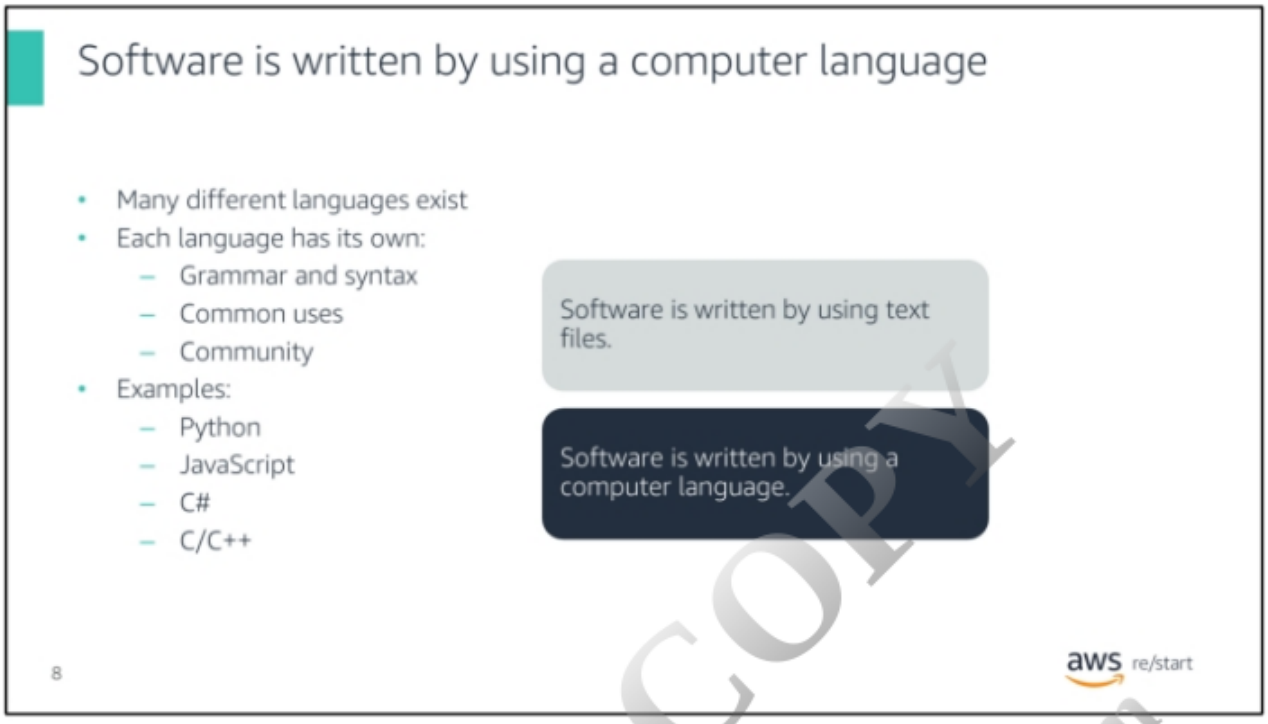
- Define programming
- Explain the function and features of an integrated development environment (IDE)
- Describe the cycle of programming





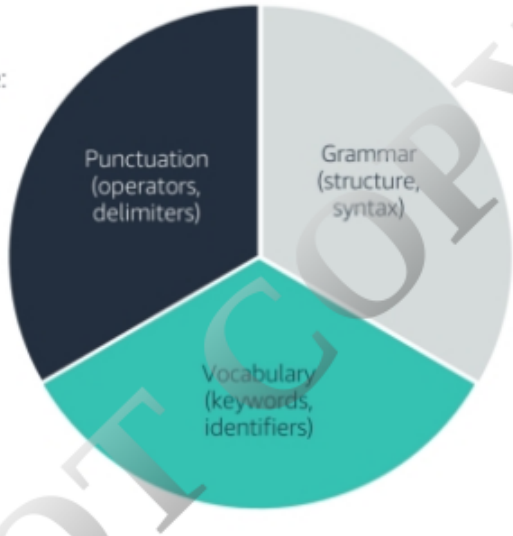






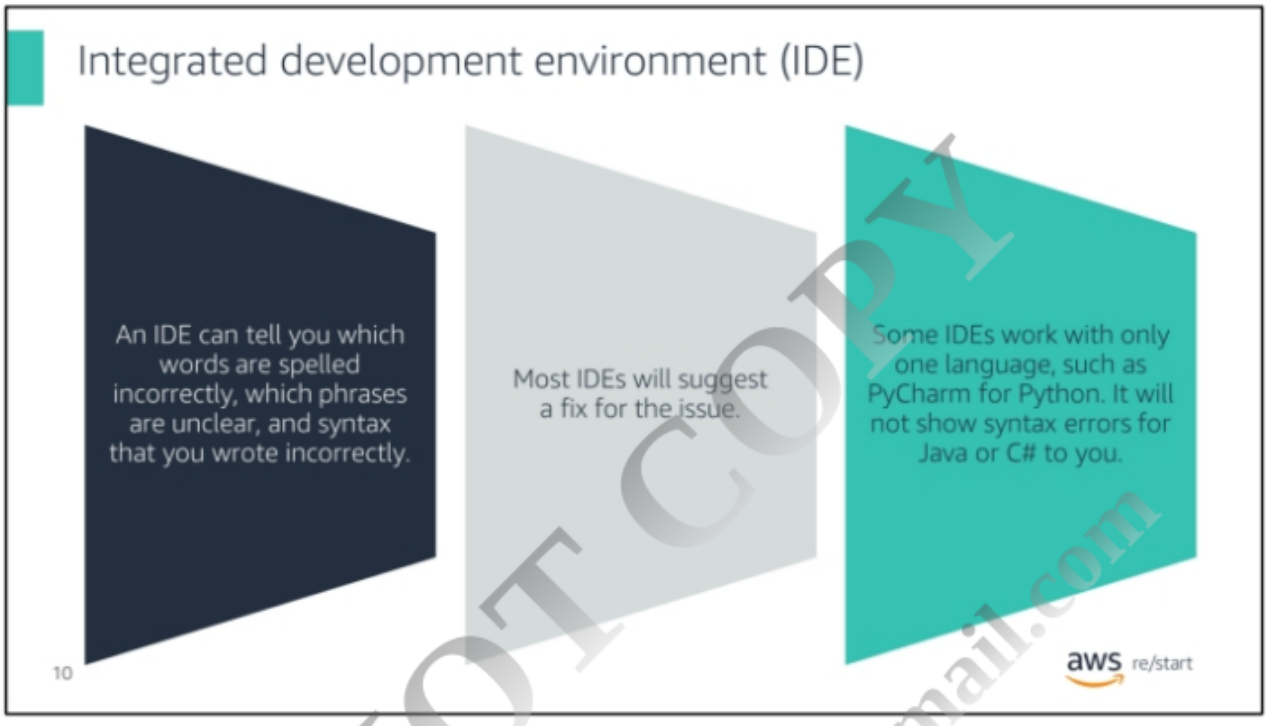
Programming language elements

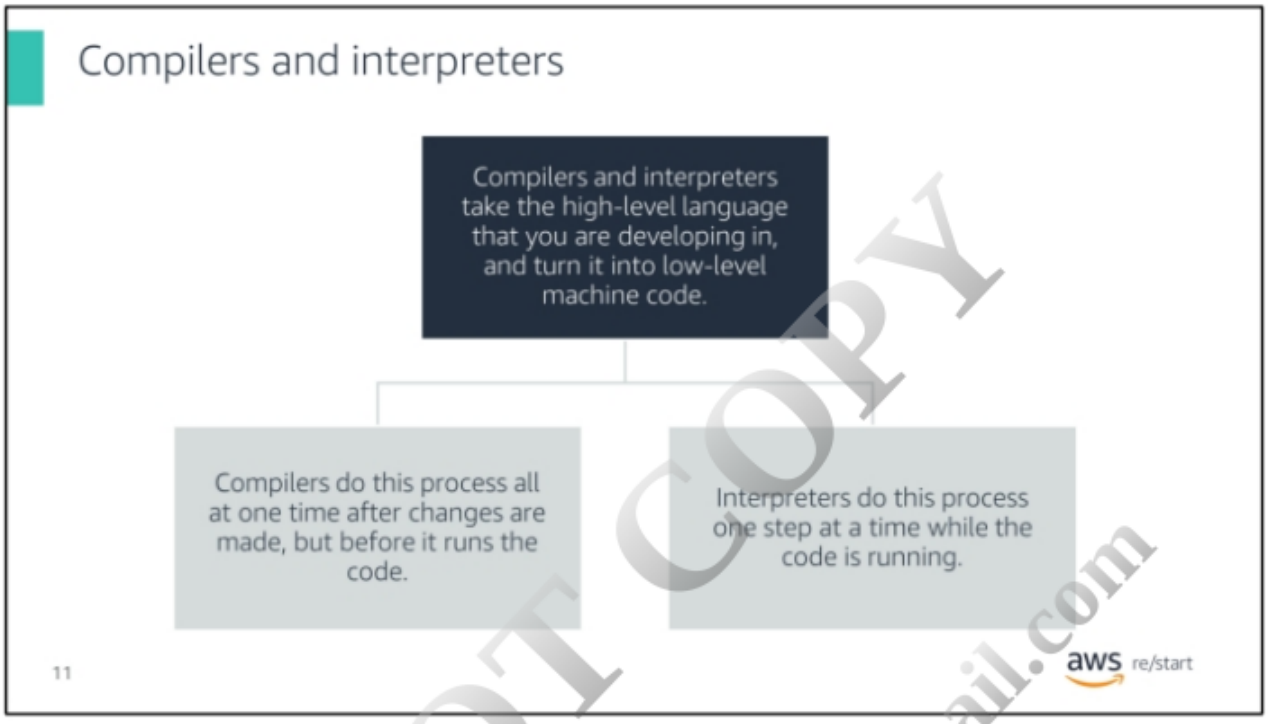
Like human languages, programming languages have:

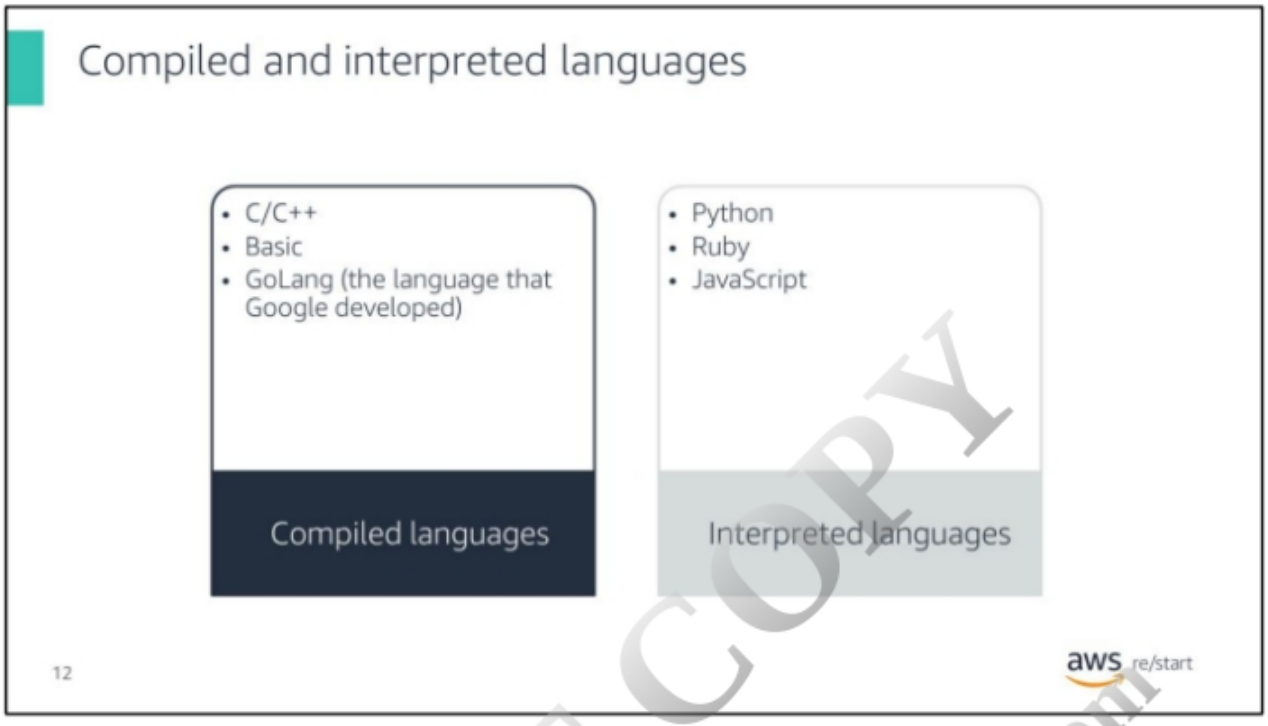


9

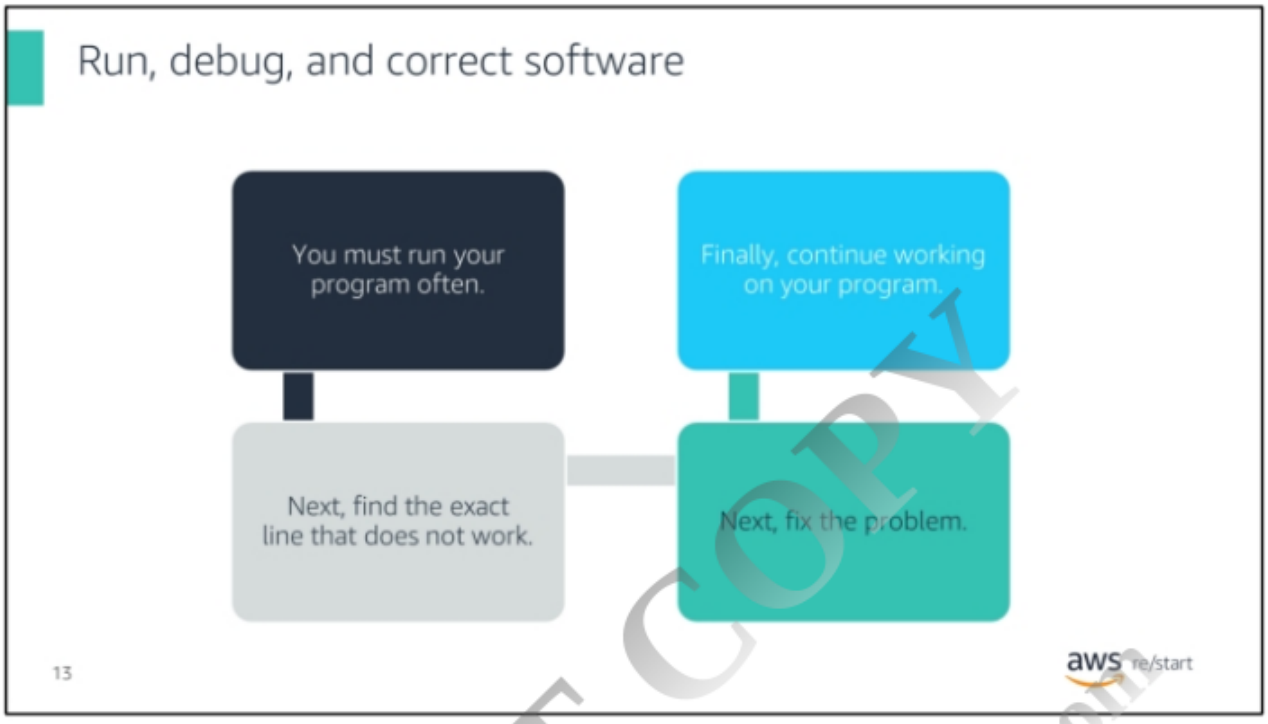
aws re/start

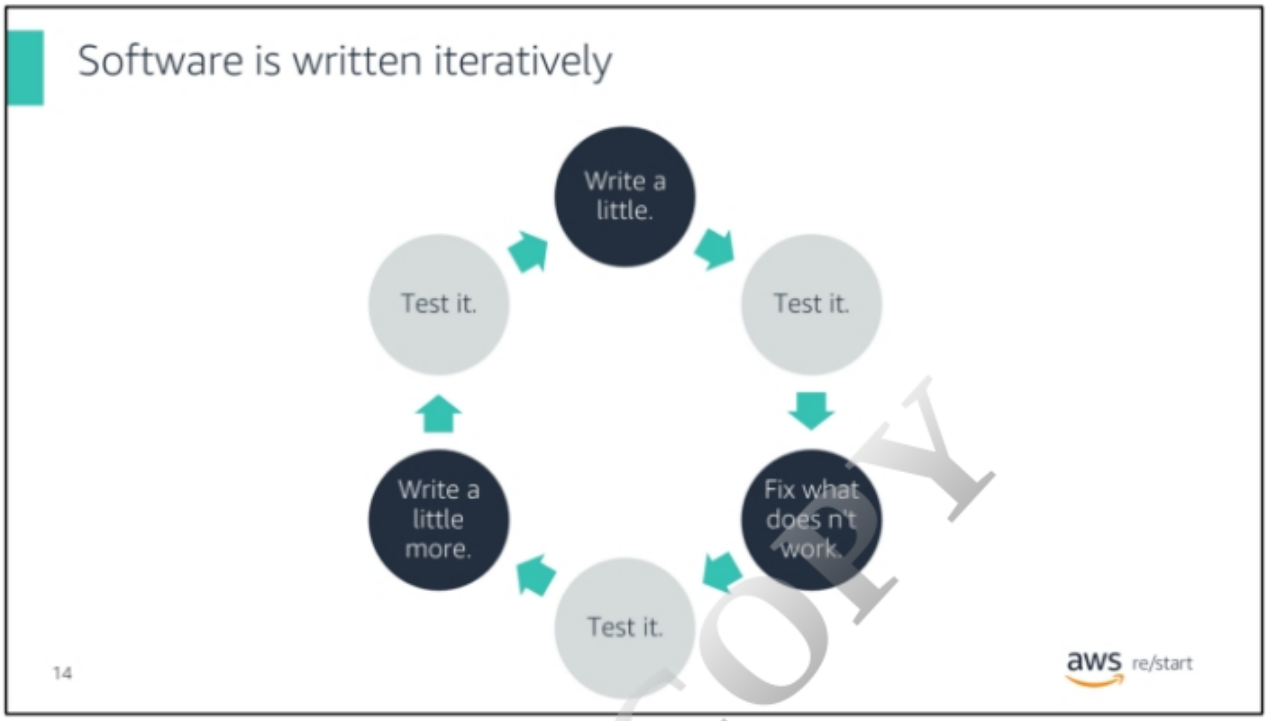


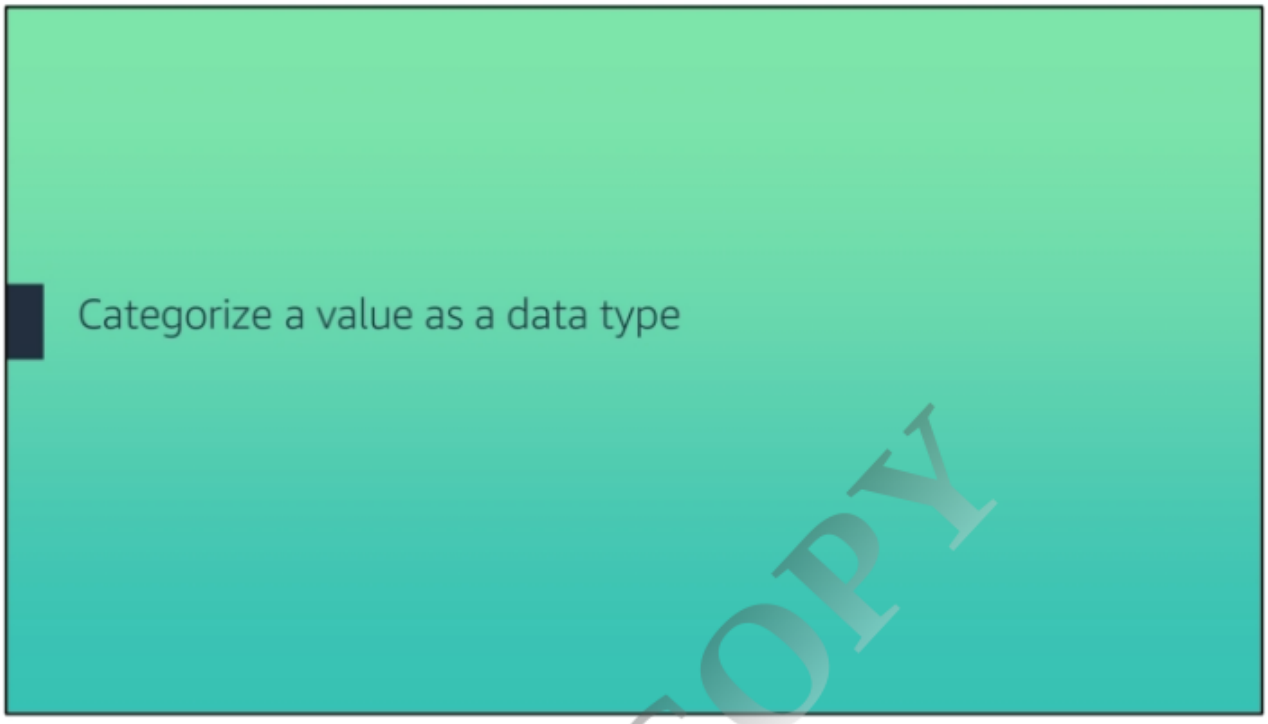




JavaScript is *interpreted* when it is loaded into a webpage.
Java is *compiled* before the program runs on a computer.







What you will learn

At the core of the lesson

You will learn how to:

- Define the term *data type*
- Correctly match data types to data
- Assign values to variables and variables to data types

16

aws re/start

In this lesson, you will learn how to:

- Define the term *data type*
- Correctly match data types to data
- Assign values to variables and variables to data types

What is a data type?

A data type is the classification of a value that tells the computer how the programmer intends the data to be interpreted.

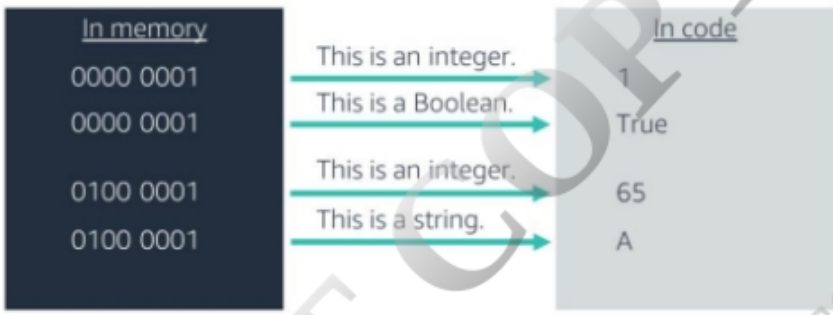
Examples:

Data Value	Data Type
45	Integer
290578L	Long
1.02	Float
True	Boolean
"My dog is on the bed."	String
"45"	String

aws re/start

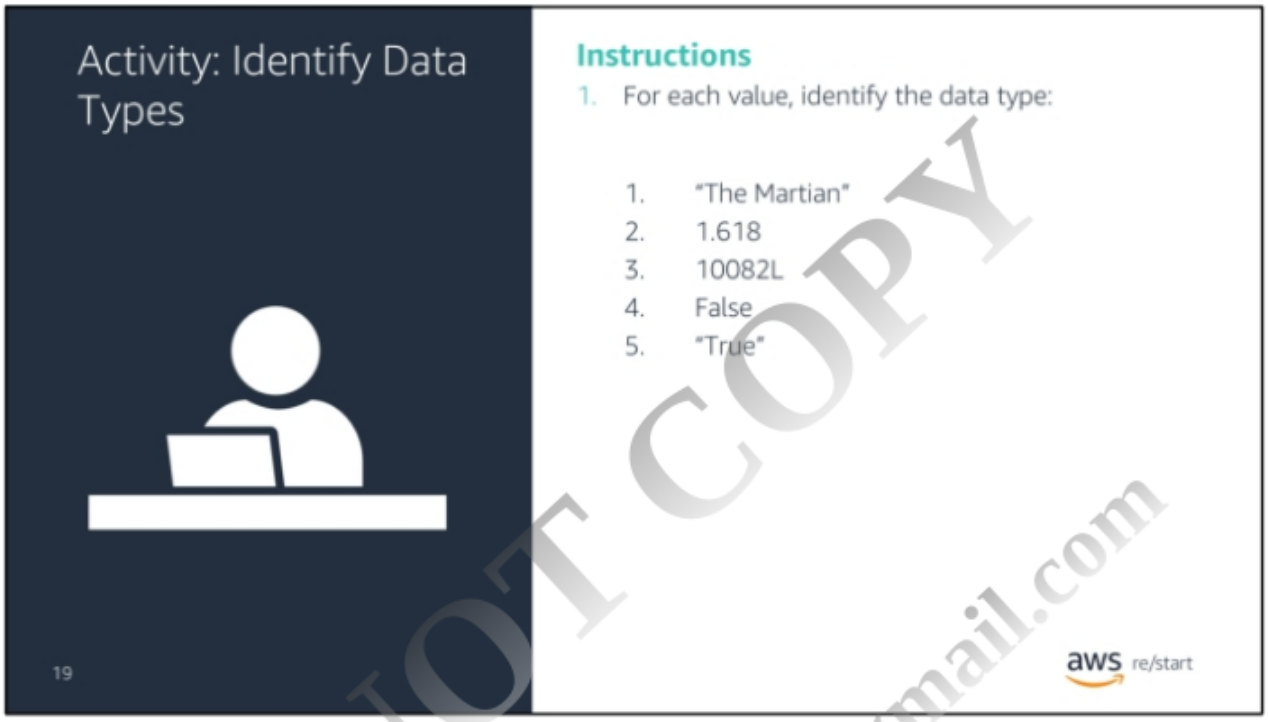
Why must the type of data be tagged?

- In memory, everything consists of 0s and 1s.
- Data typing tells the computer how to:
  - Encode a data value into memory
  - Decode a data value out of memory



18

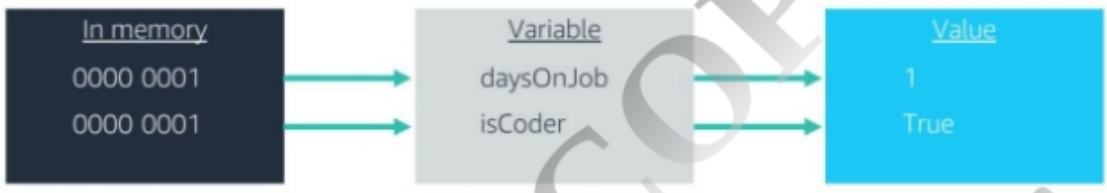
aws re/start



"The Martian" - String
1.618 - Float
10082L - Long
False - Boolean
"True" - String

What is a variable?

- A variable is an identifier in your code that represents a value in memory.
- The variable name helps humans to remember what the value means.



aws re/start

Assigning a variable to a value

- Nearly all languages have an *assignment operator*.
- Most languages use the equal sign (=).

Python **isCoder = True**	VB.NET **daysOnJob = 1**
F# **daysOnJob <- 1**	Pascal **isCoder := true**

aws re/start

Assigning a data type to a variable

- Some languages **expect** the data type to be included when first using the variable.

Objective C
int daysOnJob = 1

Java
boolean isCoder = true

Assigning a data type to a variable, continued

- Some languages *infer* the data type based on the value that is assigned.

Python
count = 10

Swift
var daysOnJob = 1

TypeScript
let isCoder = true

Activity: Represent letters as numbers

American Standards Association for Information Interchange (ASCII) is a system that associates encoding characters into computers.

Use the ASCII table at **ASCII Table** to decipher the following message:

72 – 97 – 118 – 101 – 32 – 121 – 111 – 117 – 32 – 115 – 101 – 101 – 110 – 32

109 – 121 – 32 – 99 – 97 – 114 – 32 – 107 – 101 – 121 – 115 – 63

aws re/start

Activity: Represent letters as numbers

American Standards Association for Information Interchange (ASCII) is a system that associates encoding characters into computers.

Use the ASCII table at **ASCII Table** to decipher the following message:

72 – 97 – 118 – 101 – 32 – 121 – 111 – 117 – 32 – 115 – 101 – 101 – 110 – 32

H a v e _ y o u _ s e e n _

Space

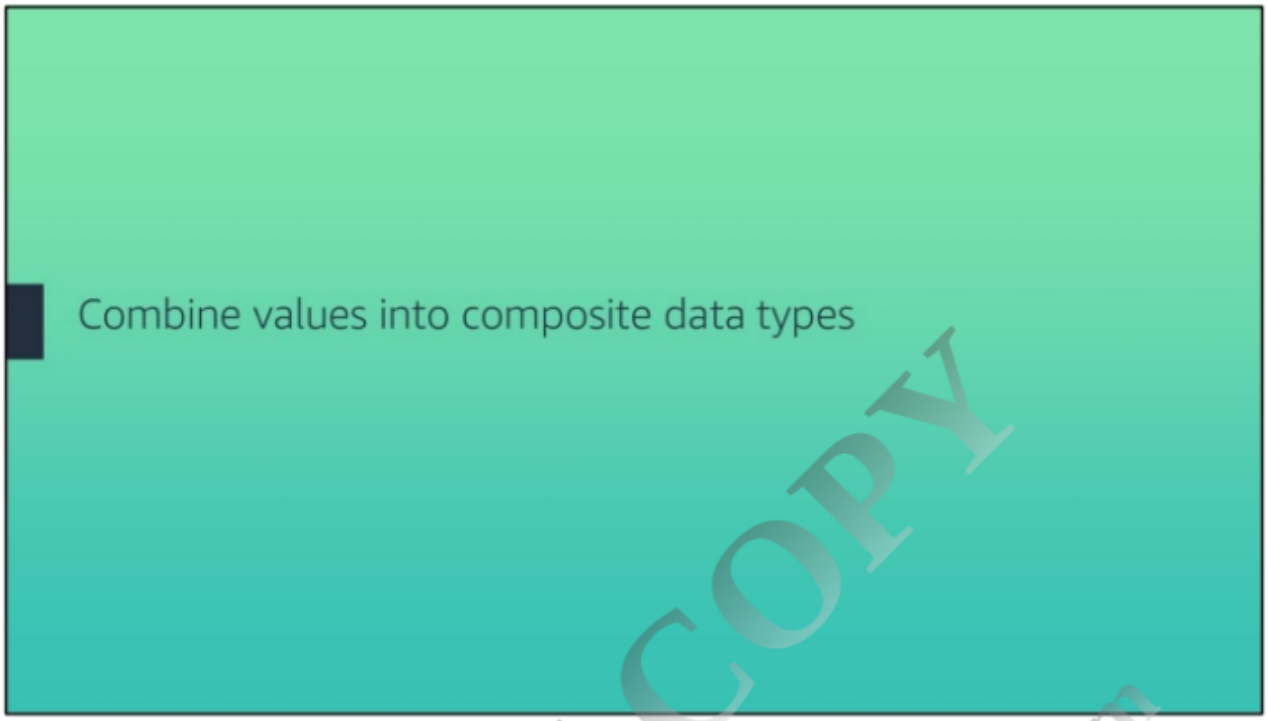
109 – 121 – 32 – 99 – 97 – 114 – 32 – 107 – 101 – 121 – 115 – 63

m y _ c a r _ k e y s ?

25

aws re/start

Answer: Have you seen my car keys?



What you will learn

At the core of the lesson

You will learn how to:

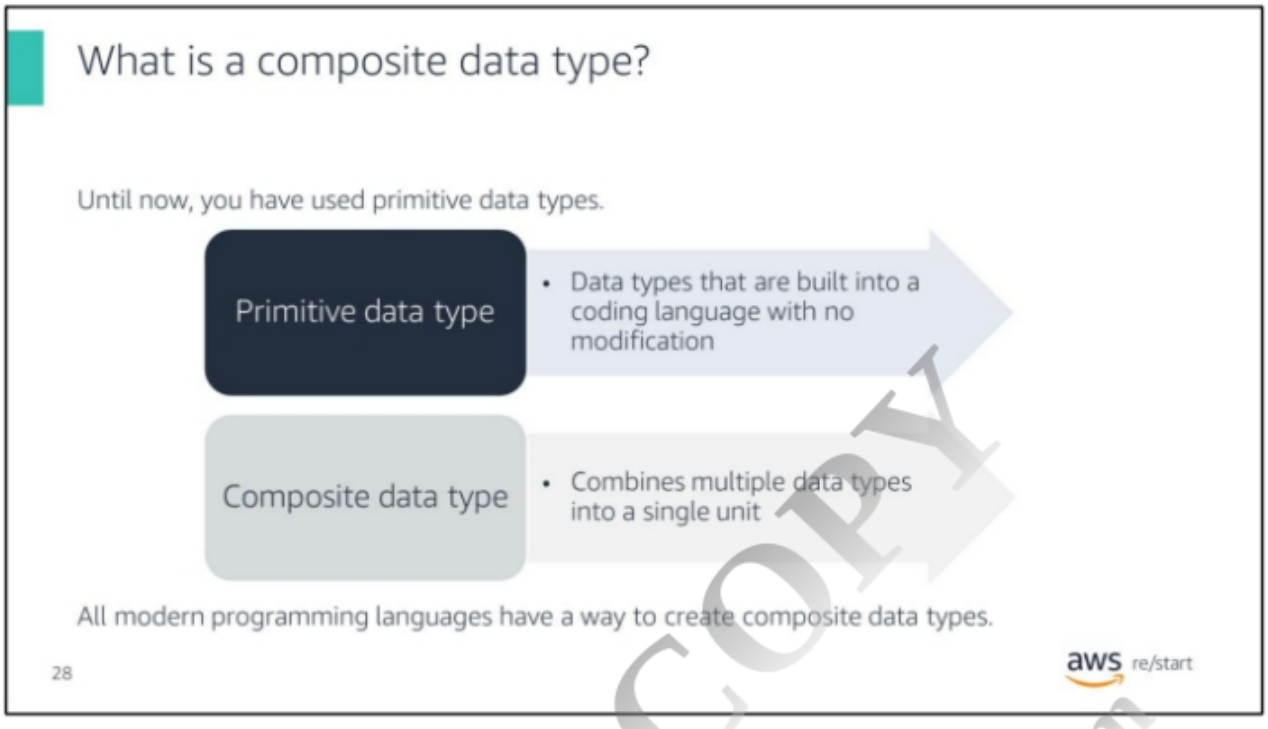
- Define composite data types
- Identify composite data types from a list of properties
- List the attributes of a function
- Define different types of collections
- Combine values into a composite data type

27

aws re/start

In this lesson, you will learn how to:

- Define composite data types
- Identify composite data types from a list of properties
- List the attributes of a function
- Define different types of collections
- Combine values into a composite data type



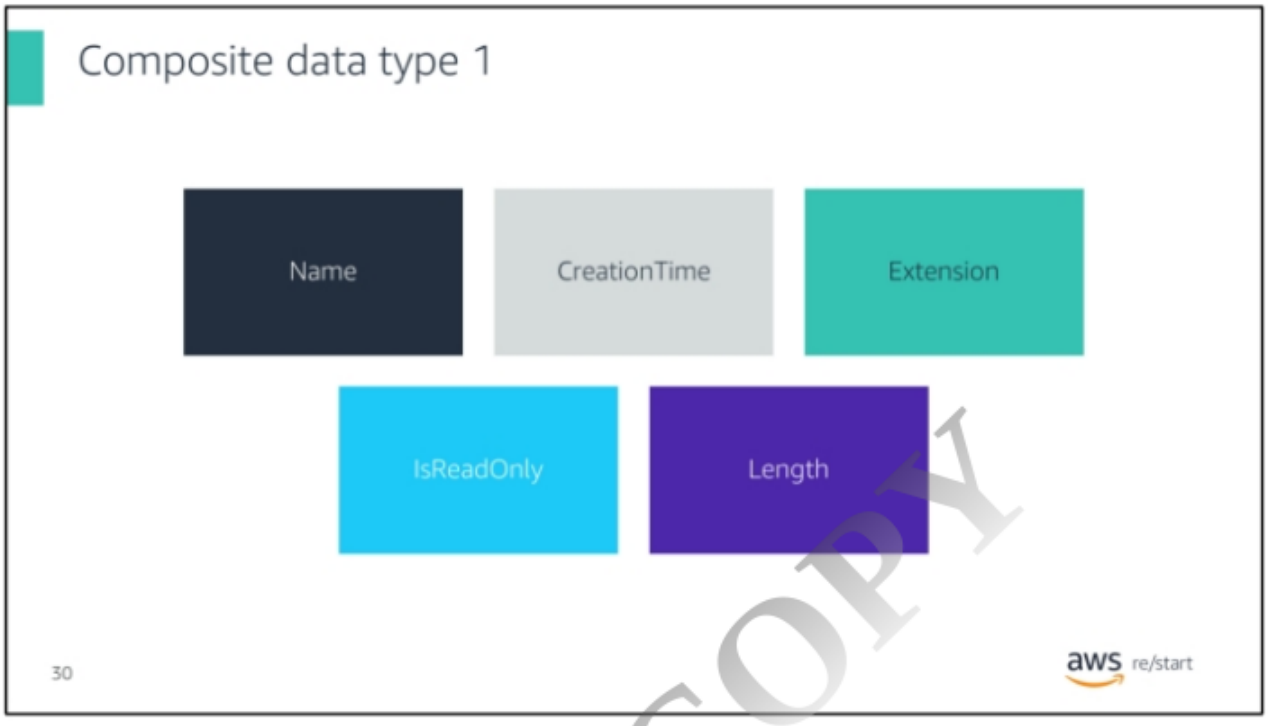
Composite data type example: Movie

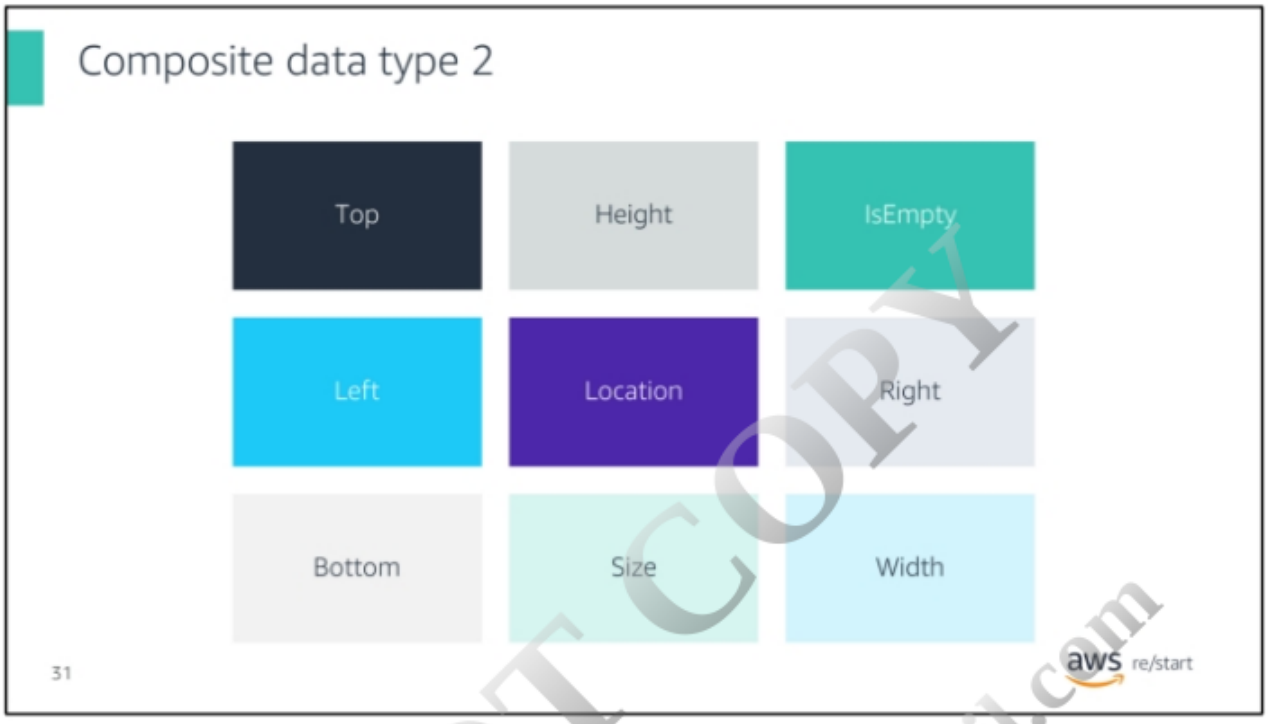
Each movie could be described with:

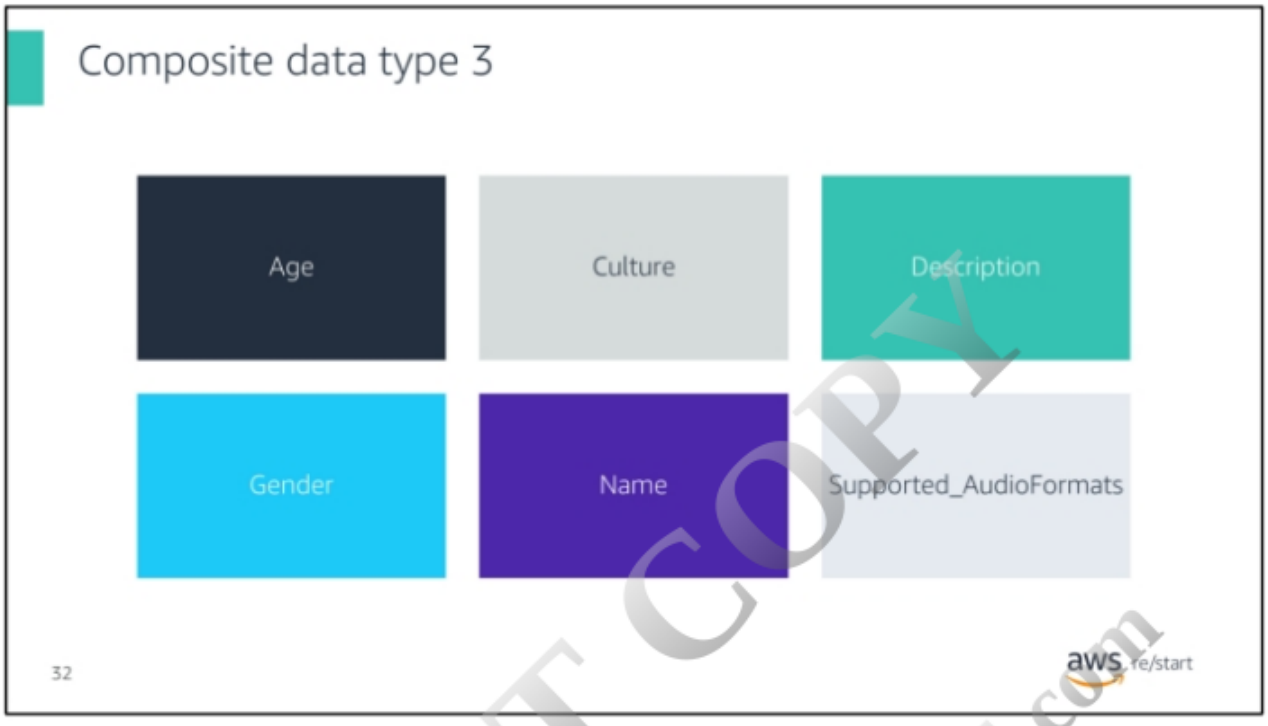
- A *name* (data type would be a *string*)
- A *year* that it was released (data type would be a *number*, or *integer*)
- A *flag* to indicate whether you have seen it or not (data type would be a *Boolean*)

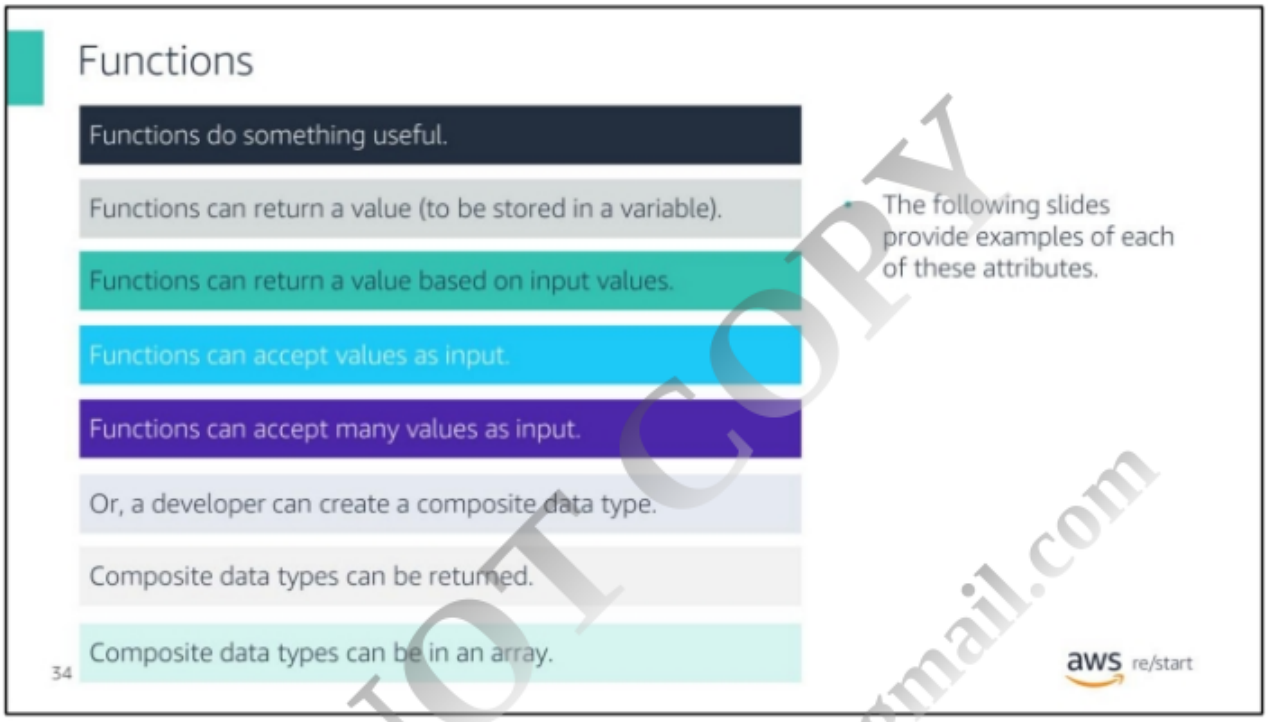
29

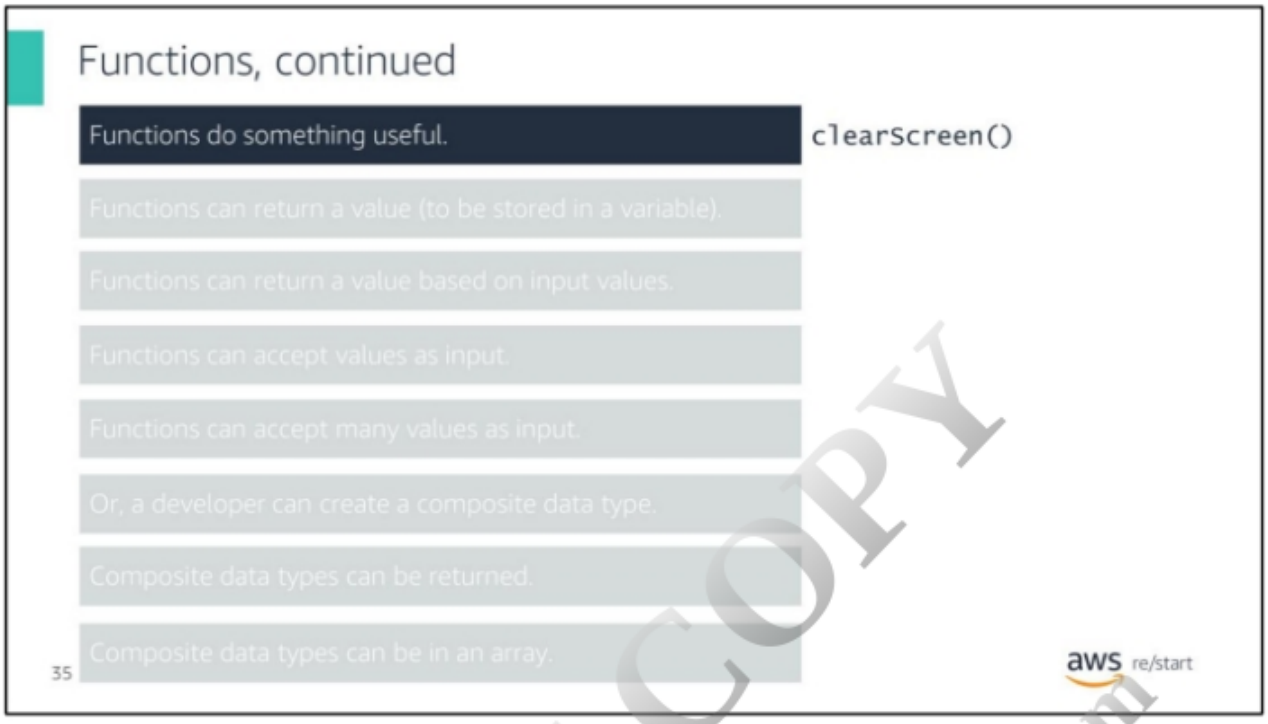
aws re/start

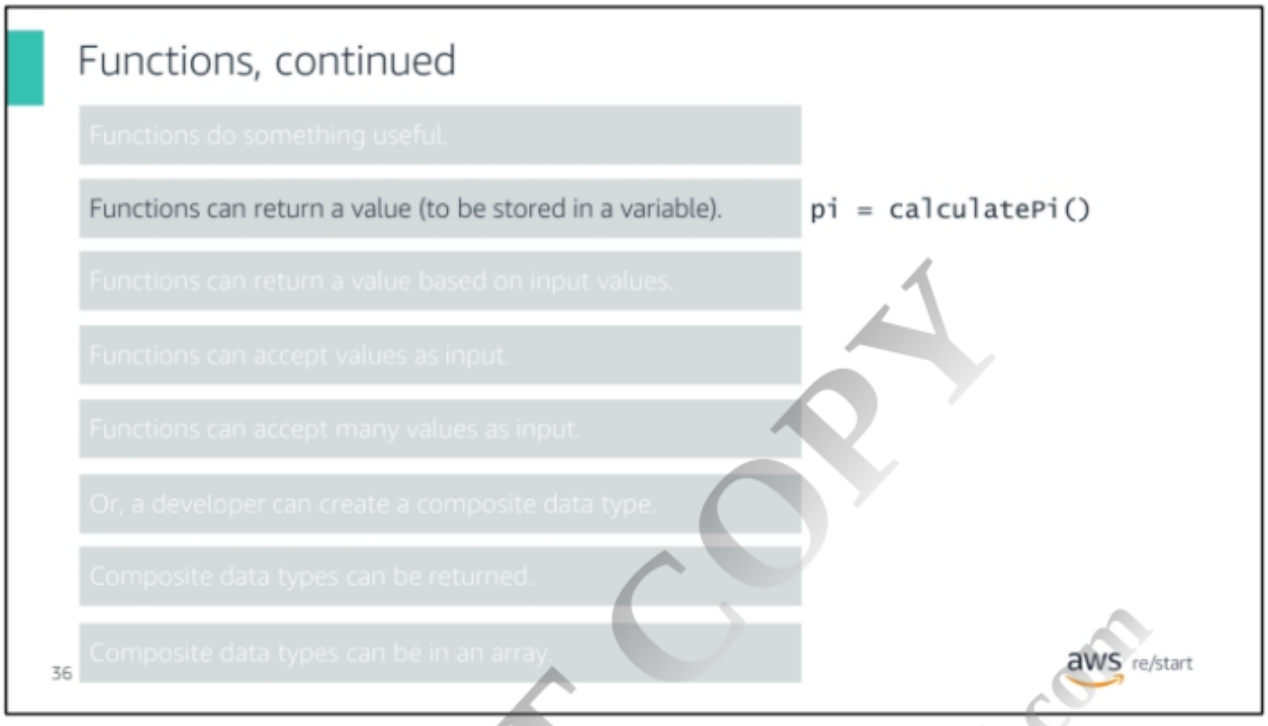


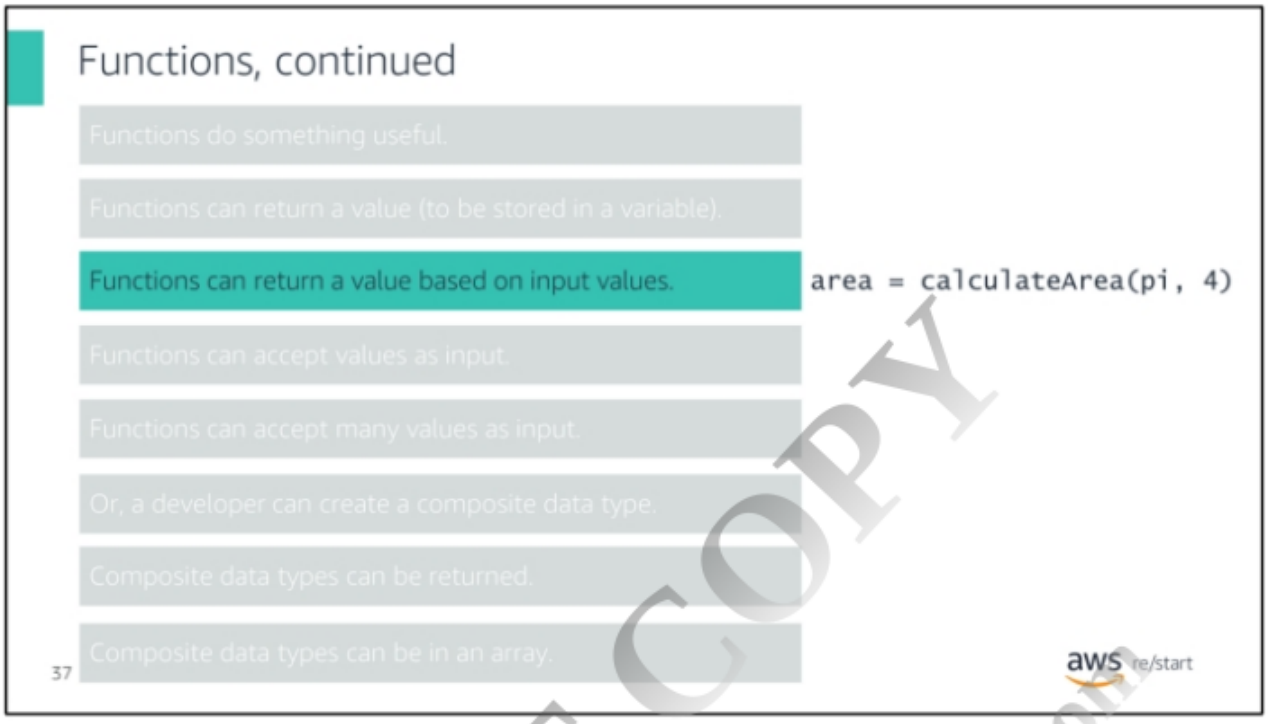


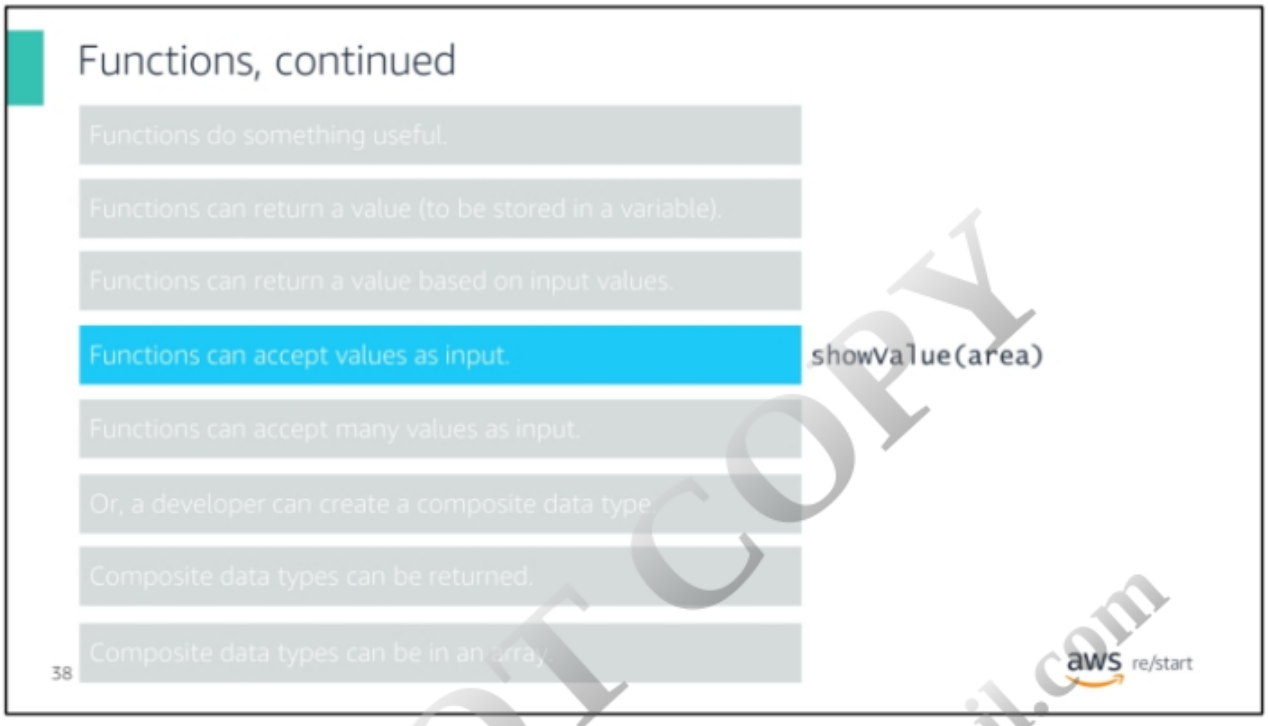


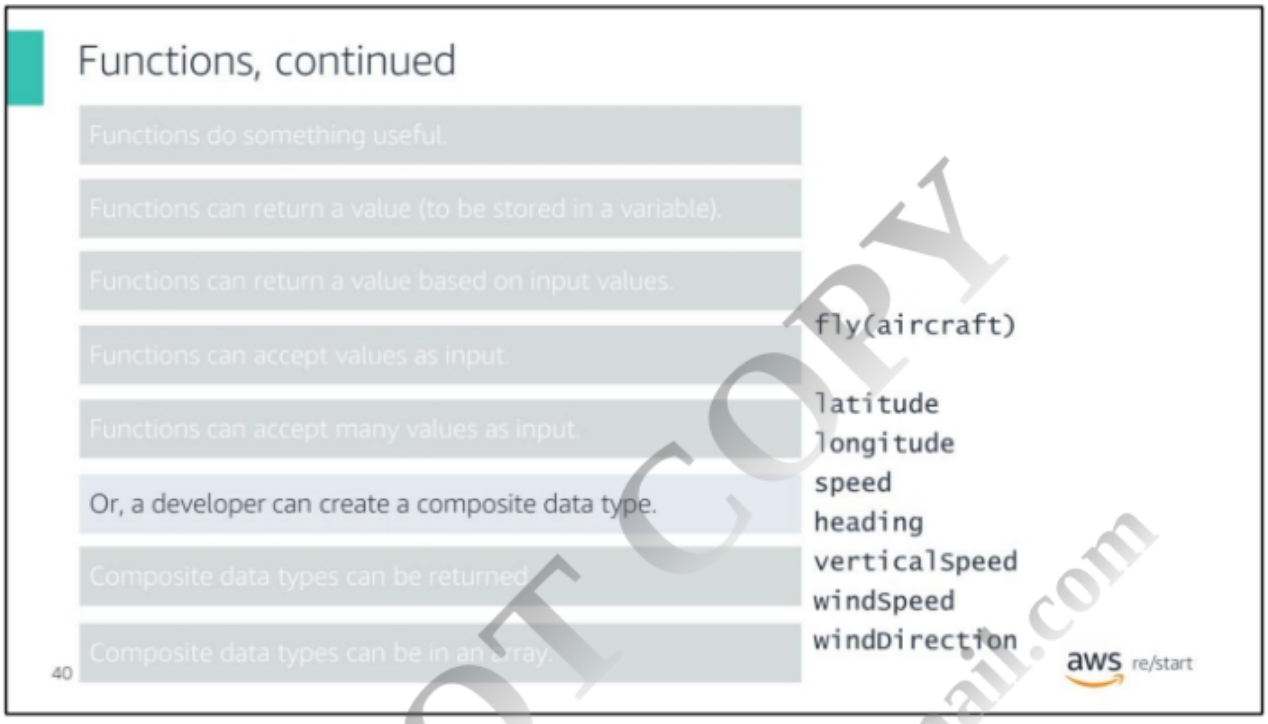


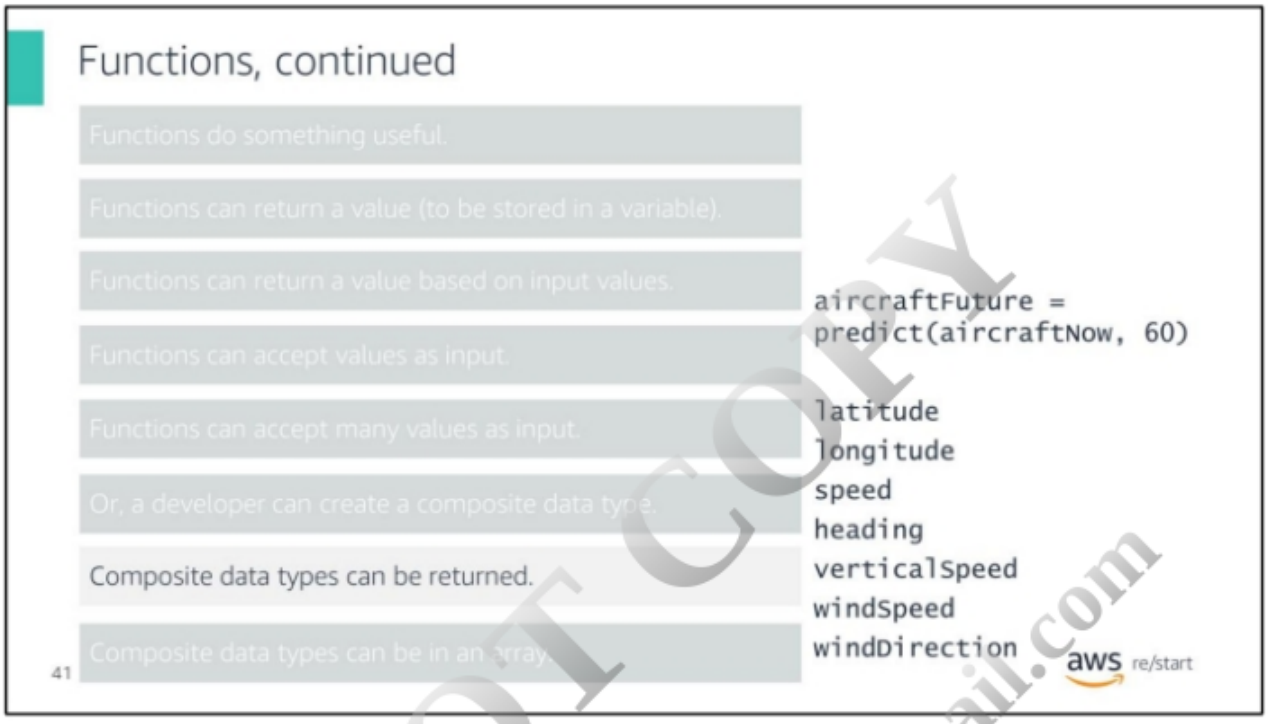


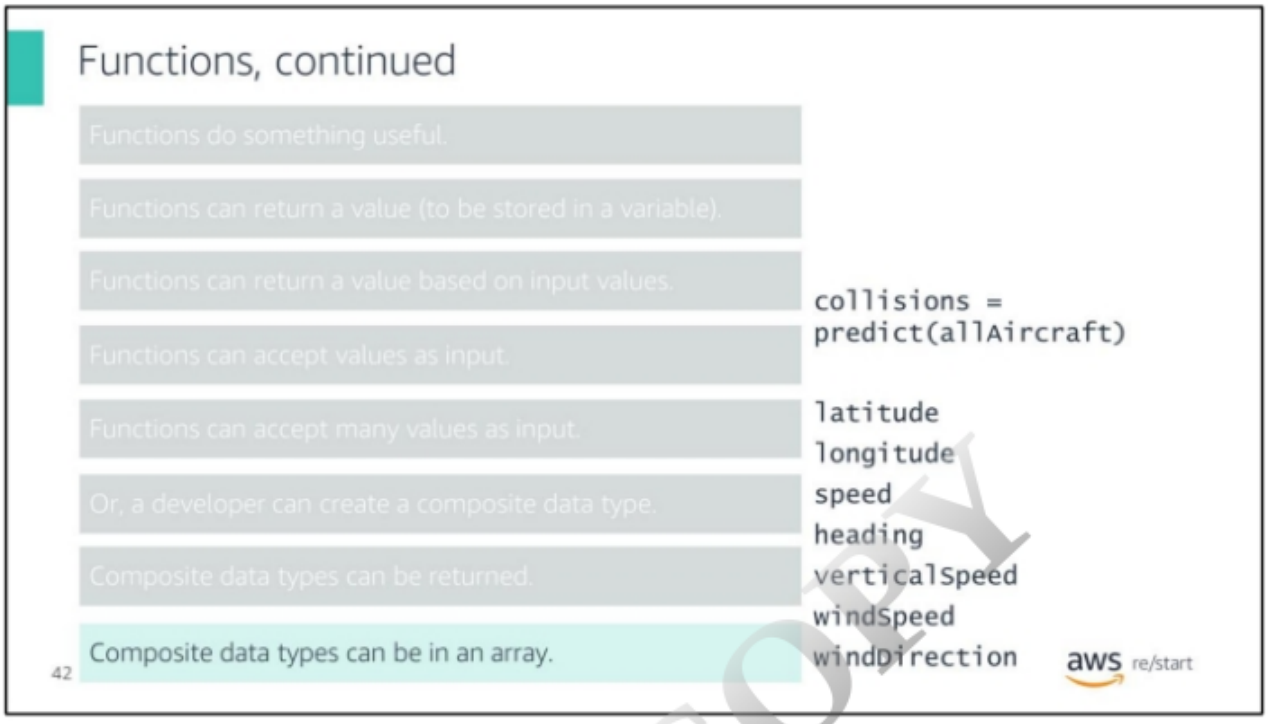


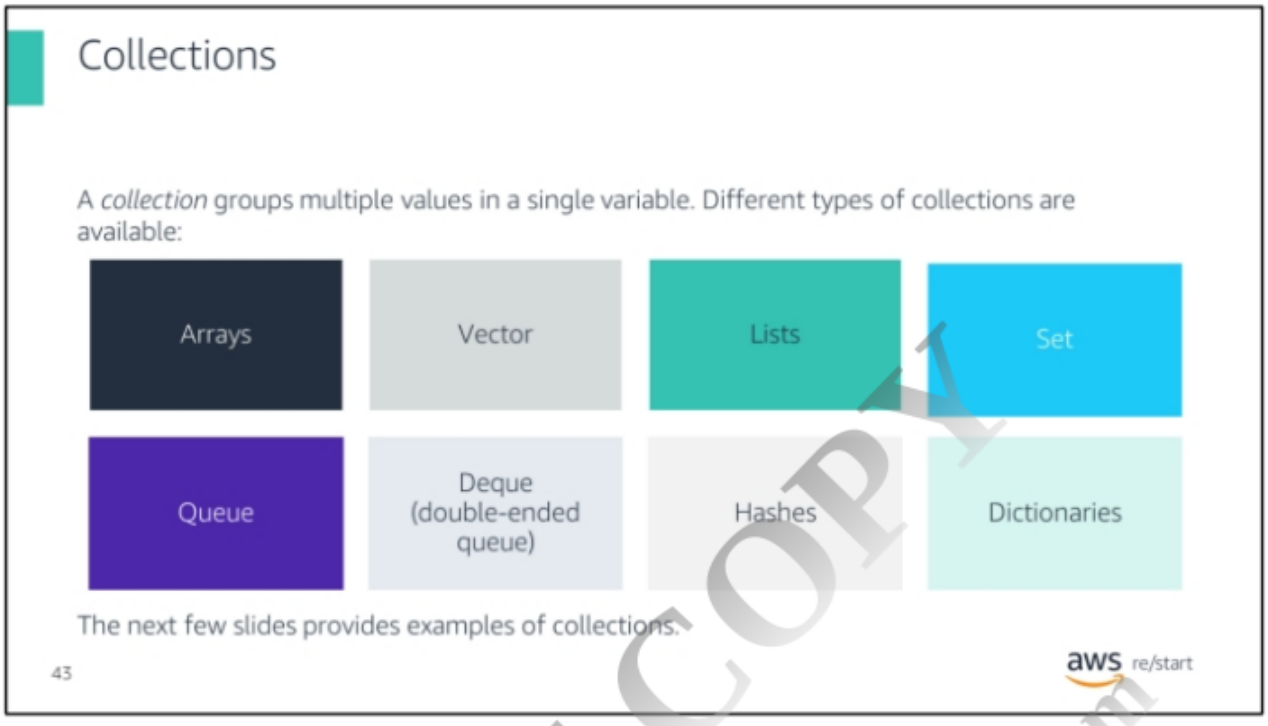












Example collection: Array

The table shows a basic *array* of ages. Each age is assigned a slot, is of the same data type, and is adjacent to the previous age and the next age in memory.

- The first slot is almost always 0 in every language.
- Notice that the values do not need to be in any order.

Slot	Data Type	Value
0	Integer	87
1	Integer	10
2	Integer	2
3	Integer	46
4	Integer	22
5	Integer	19
6	Integer	66

aws re/start

Python collection example: List

A Python *List* is an **ordered** collection of items.

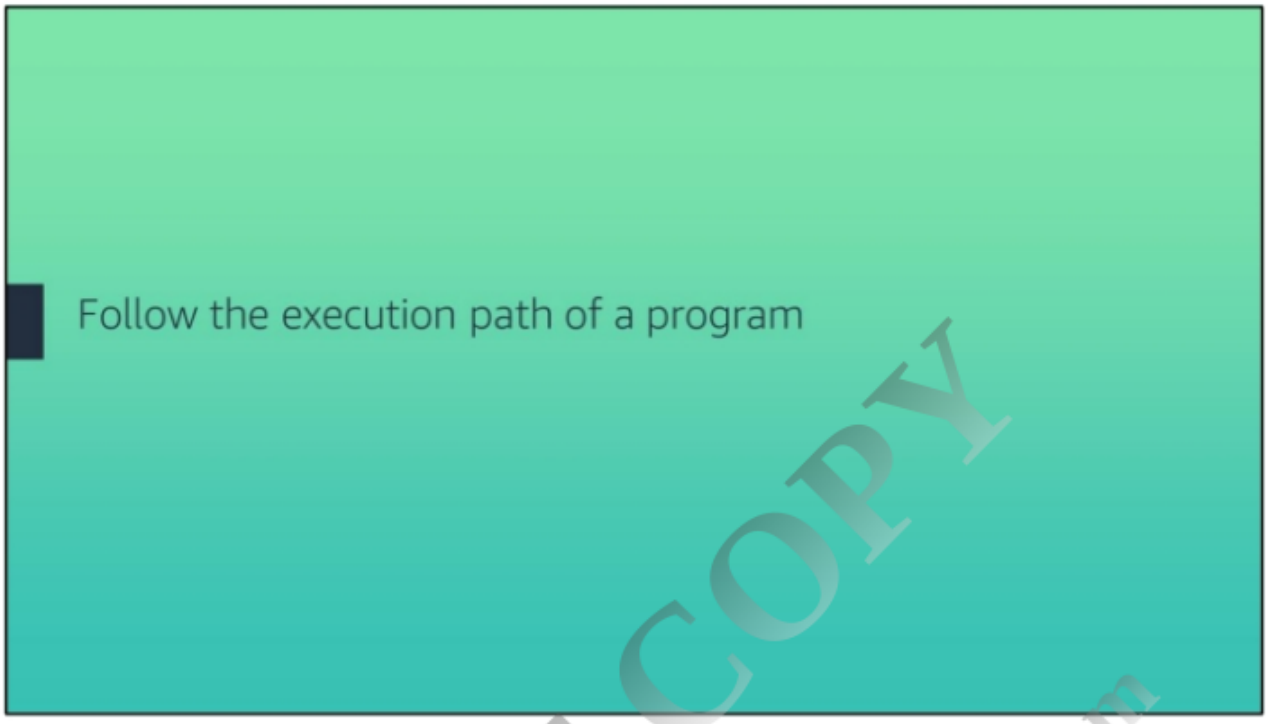
- Each item can be of a different data type.
- Items can be accessed using an *index*.
- Items can have duplicate values.

Index	Data Type	Value
0	String	"John"
1	Integer	55
2	Boolean	True
3	String	"male"
4	String	"Carlos"
5	Integer	22
6	String	"male"

45

aws re/start

A Python list is ordered because each new item is added to the end of the list.



What you will learn

At the core of the lesson

You will learn how to:

- Explain what an execution path is
- Describe the use of different types of flow control mechanisms, including loops, conditionals, and switches

47

aws re/start

In this lesson, you will learn how to:

- Explain what an execution path is
- Describe the use of different types of flow control mechanisms, including loops, conditionals, and switches

What does *execution path* mean?

- The sequence of steps that the program performs when it runs
- The program might …
  - Come to an either-or choice
  - Come to multiple choices
  - Perform work on each item in a loop
- The programmer must be able to predict what those steps will be…
  - When they write the code initially
  - When they debug problems that they encounter

Conditionals

- All programming languages have a way to choose an either-or path in the code.

```
if (guess > number):
    print("Too high!")
elif (guess < number):
    print("Too low!")
else
    print("Exactly right!")
```

Example from Python

```
if(name == 'Juan'):
    bonus = 300
else:
    bonus = salary * 0.1
```


aws re/start

Switches

- Most programming languages have a way to conveniently handle multiple possible cases of a value.

```
switch(sign):
    case "Stop":    pressBreak()
    case "Merge":   accelerate()
    case "Exit":    decelerate()
    default:            ignore()
```

Loops

- All programming languages have a way to do something on each item in a collection.

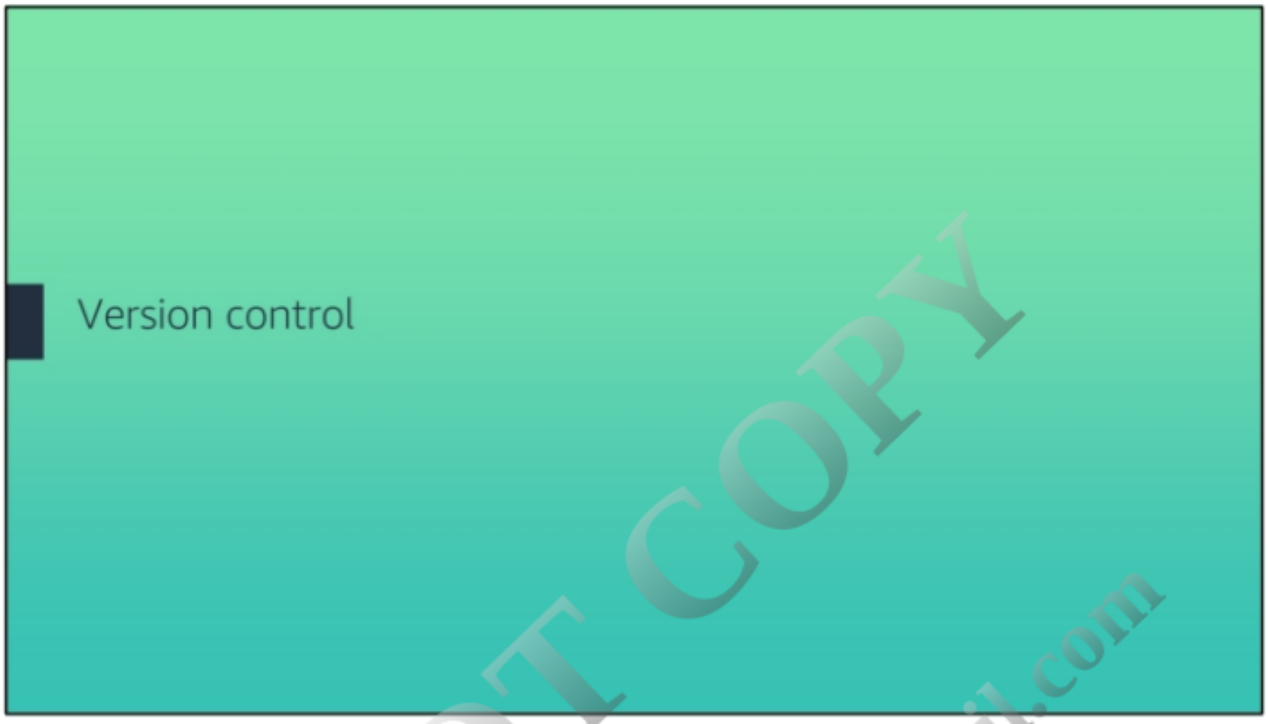
```
names = {“wei", “nikki", “akua"}

for name in names:
    newName = capitalize(name)
    print newName
```

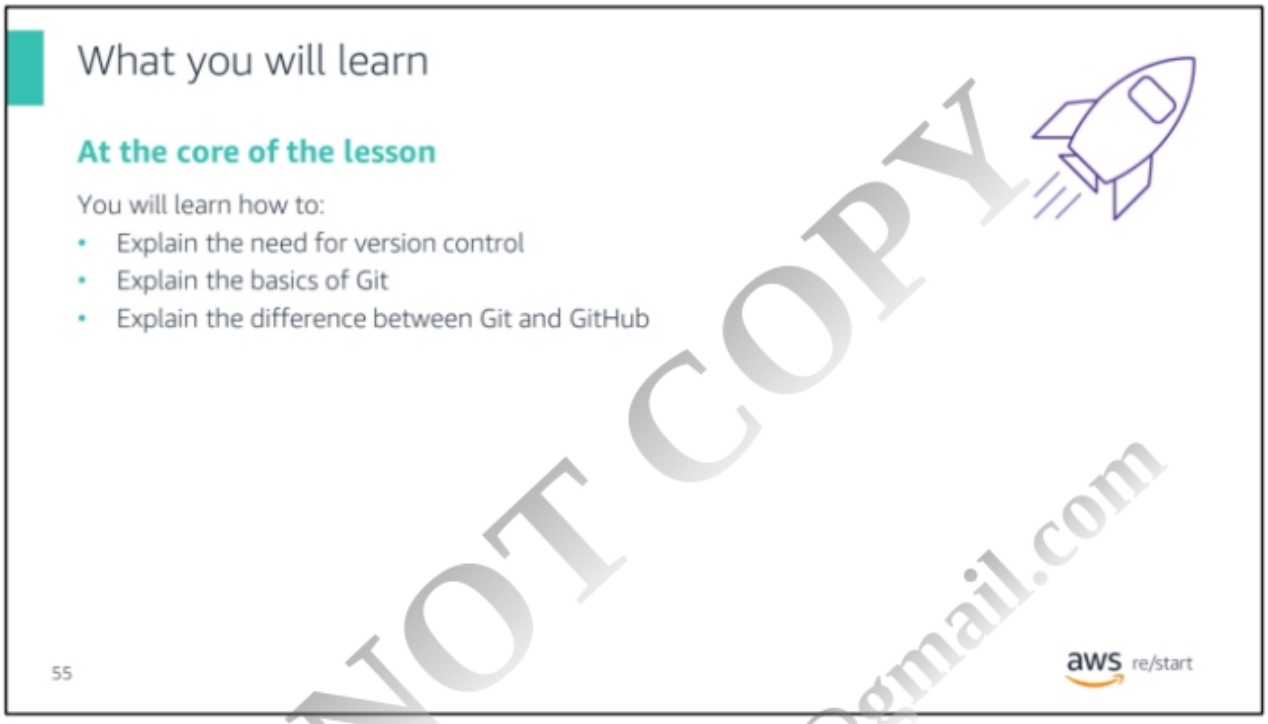

aws re/start

Example from Python

```
for(employee in employees):
    employee.bonus = employee.salary * 0.1;
```

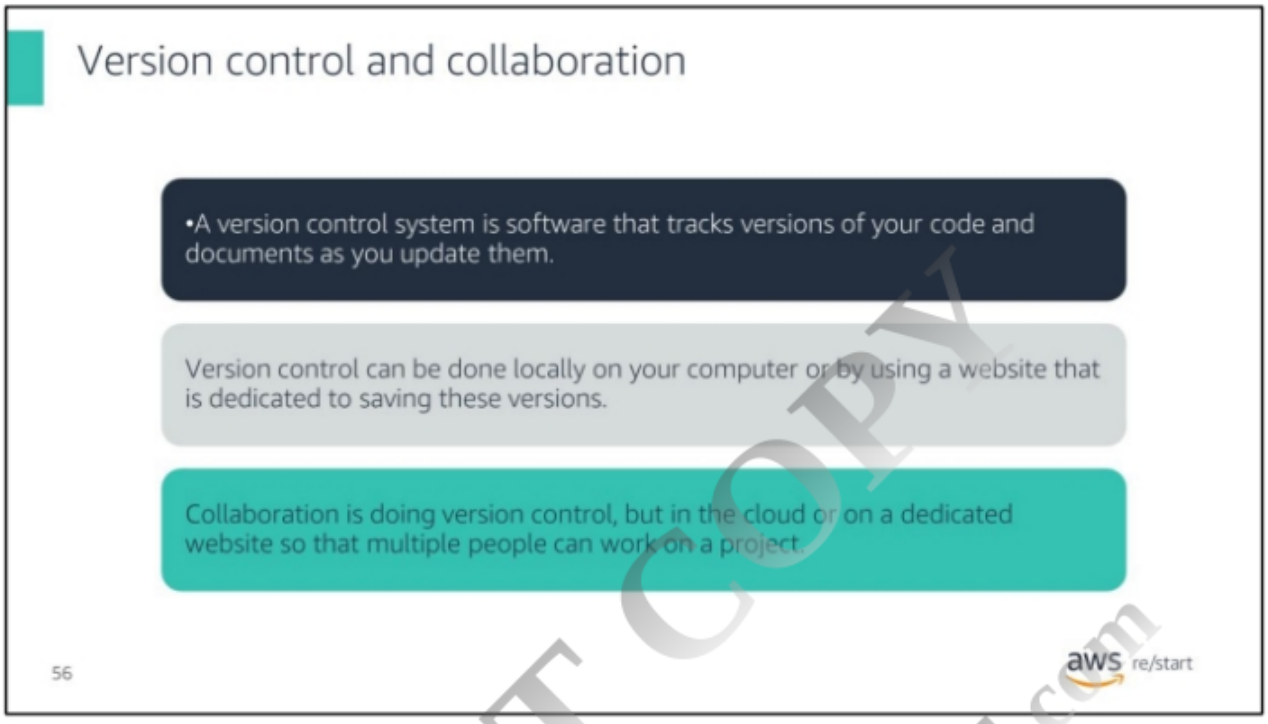

aws re/start

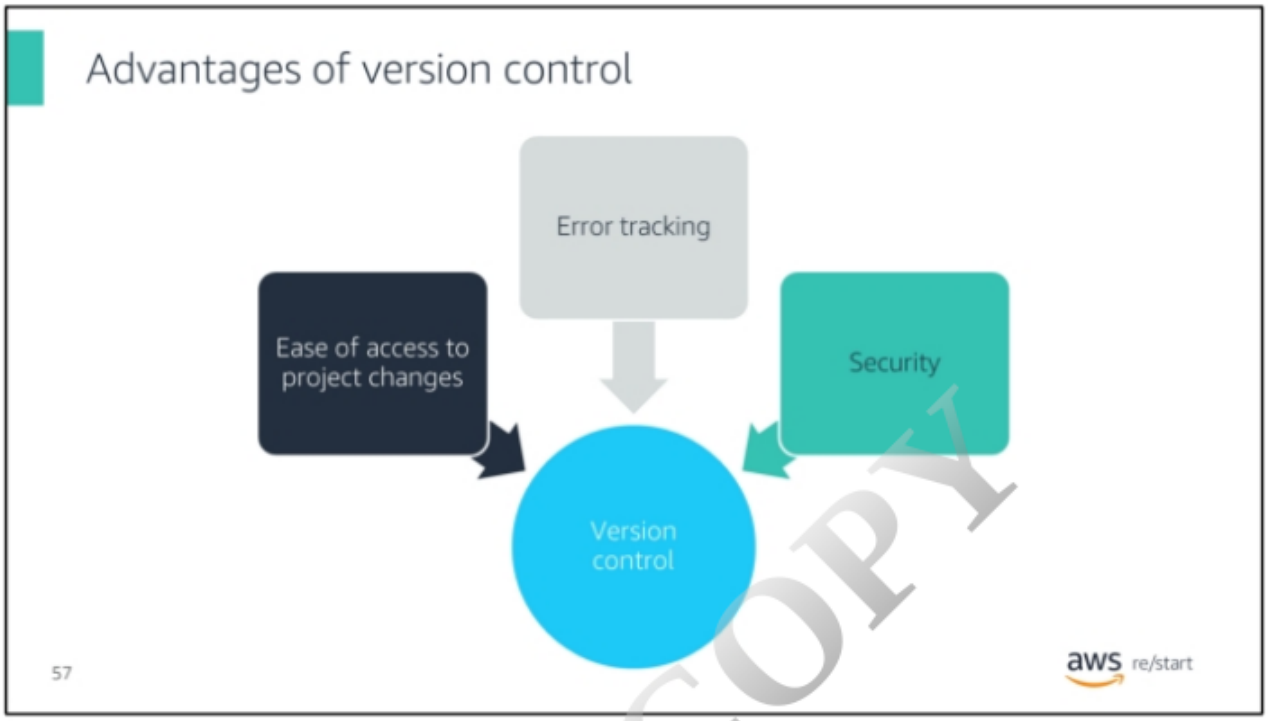




In this lesson, you will learn how to:

- Explain the need for version control
- Explain the basics of Git
- Explain the difference between Git and GitHub





Utilizing cloud infrastructure

The cloud, or a dedicated website, is useful for storing changes in code.

Version control that is only on a local computer can be easily lost, even if it is more secure than saving multiple versions of a file.

Data that is stored in the cloud has a reduced risk of being lost.

Version control tools

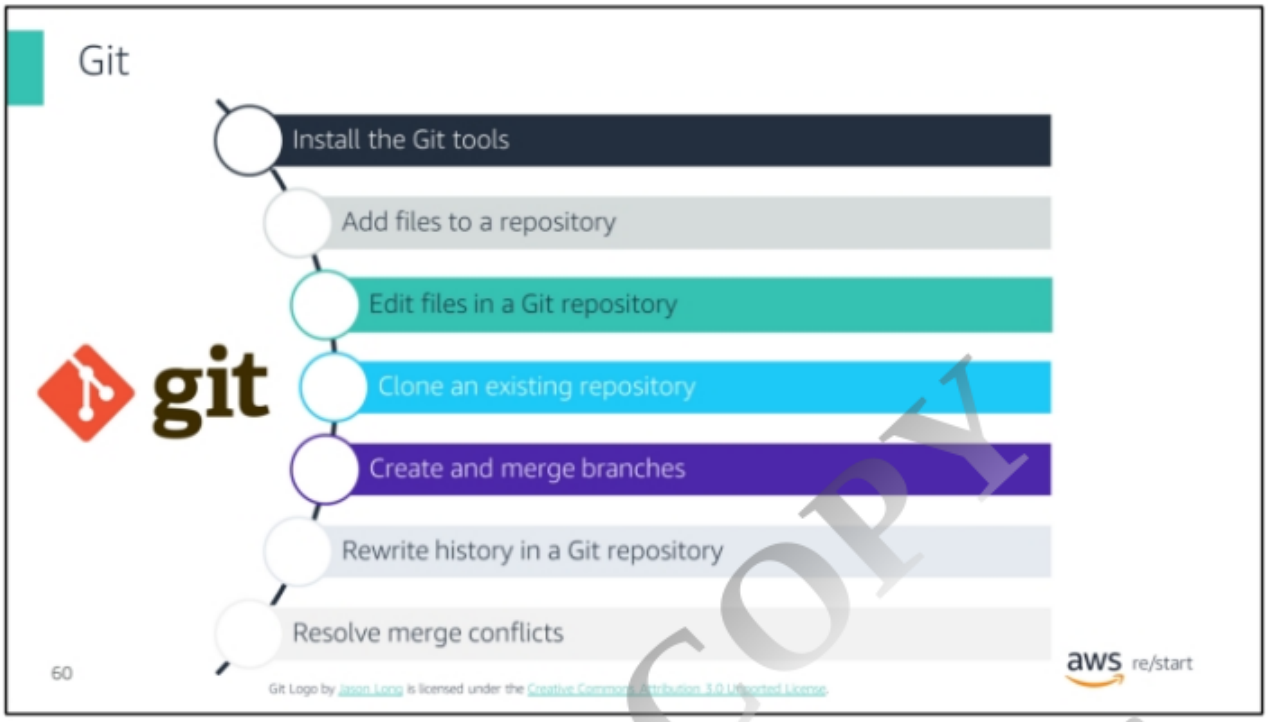
Several version control tools are available:

- Git and GitHub
- GNU arch
- Mercurial

More version control tools exist, but the one that this course focuses on is Git and GitHub.

Mercurial  GitHub

aws re/start



Git, continued

The best way to learn Git is to experience it.

1. Install the Git Tools from the website: About Git.
2. After you install the Git Tools, open the Git Bash program.
3. In the Bash terminal, create a folder or directory that is named *GitTest* by entering: *mkdir GitTest*
4. Change directories by entering: *cd GitTest*
5. Create a local Git repository by entering: *git init*
6. Confirm that a new repository is there, by entering *ls -ldF .** and looking for the .git repository.
7. Next, create a new file by entering: *touch newFile.txt*
8. Open the new file by entering: *nano newFile.txt*

aws re/start

Git, continued

9. Enter any text that you want.
10. To exit from nano, press CTRL+X.
11. In the terminal, enter: *git status*
    You should see newFile.txt in red characters.
12. Now, add newFile.txt to the local Git repository by entering *git add newFile.txt* and then *git status*
    You should see that newFile.txt is in now in green characters, which means that it is ready to be committed.
13. To finish the process, enter *git commit -m "Initial commit"* followed by *git status* to confirm that Git is not tracking any new changes.

Git, continued

If the commit message did not work, you most likely saw a message that said *Please tell me who you are* in the terminal.

14. To clear this message, in the terminal, enter the following two commands :
    - *git config –global user.email "<you@example.com>"*
        - Replace <you@example.com> with an email address that you actually own and want to use for this course.
        - The rest of the command is entered exactly, including the quotation marks (" ").
    - *git config –global user.name "<Your name>"*
        - Replace <*Your name*> with a user name that you want to use for this course. Make it unique.
        - The rest of the command is entered exactly, including the quotation marks.

aws re/start

Git, continued

15. After you are finished, enter *git commit -m "Initial commit"* again, followed by *git status* to confirm that Git is not tracking any new changes.

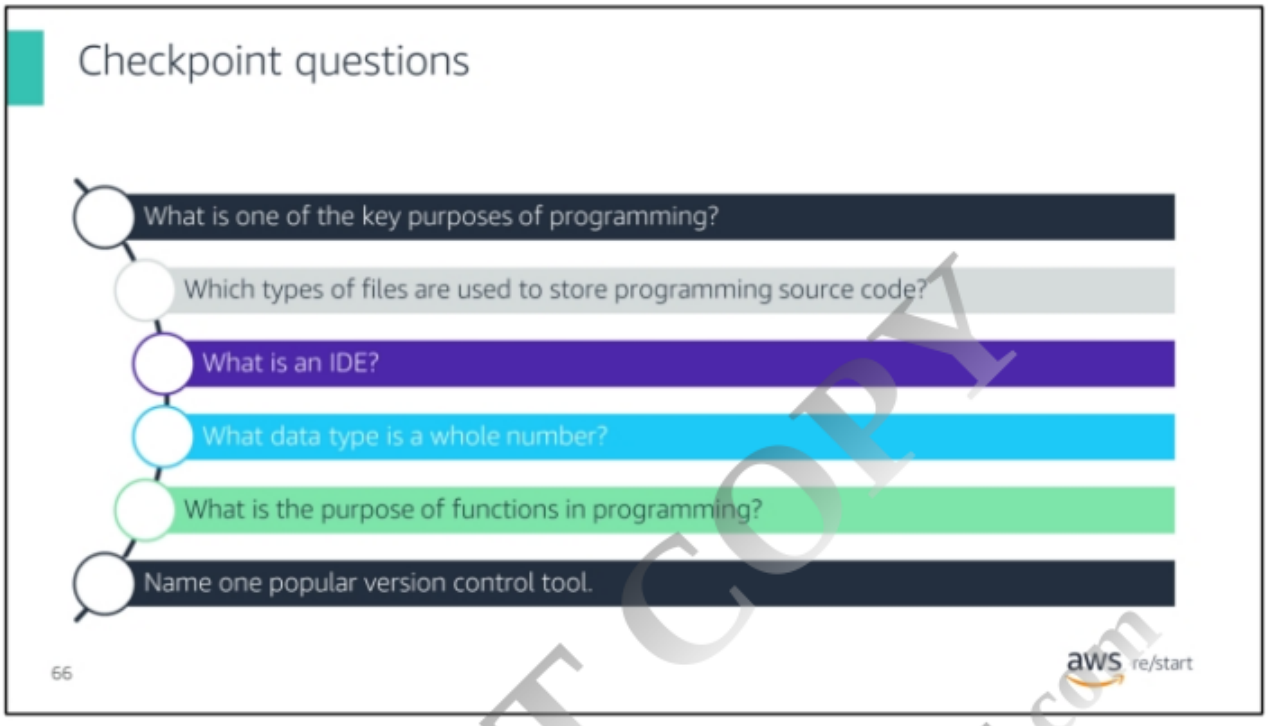
aws re/start

GitHub

GitHub is a repository hosting service. It uses repositories and the same commands as the terminal.

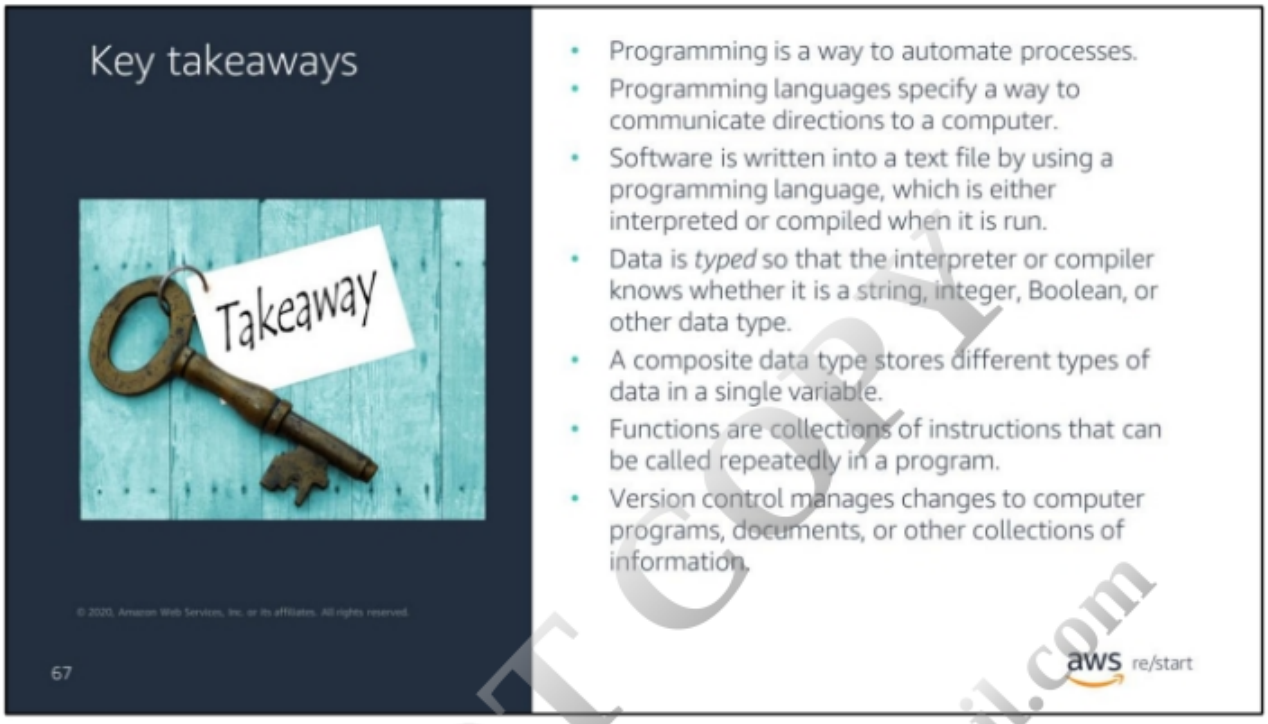
To prepare for the lab, create a GitHub account on GitHub, if you don't already have one.

Sites like GitHub are also a good way to exhibit your code to developers and potential employers.



Answers:

1. From the lesson, it was shown that automation is one of the uses of programming.
2. Programs are written in text files.
3. An IDE is an integrated development environment that helps with writing code.
4. Whole numbers are typically stored as integers.
5. Functions enable developers to create discrete sections of code that can be called by using the function's name. The resulting function is then available to the rest of the source code.
6. Git.



Some key takeaways from this lesson include:

- Programming is a way to automate processes.
- Programming languages specify a way to communicate directions to a computer.
- Software is written into a text file by using a programming language, which is either interpreted or compiled when it is run.
- Data is *typed* so that the interpreter or compiler knows whether it is a string, integer, Boolean, or other data type.
- A composite data type stores different types of data in a single variable.
- Functions are collections of instructions that can be called repeatedly in a program.
- Version control manages changes to computer programs, documents, or other collections of information.