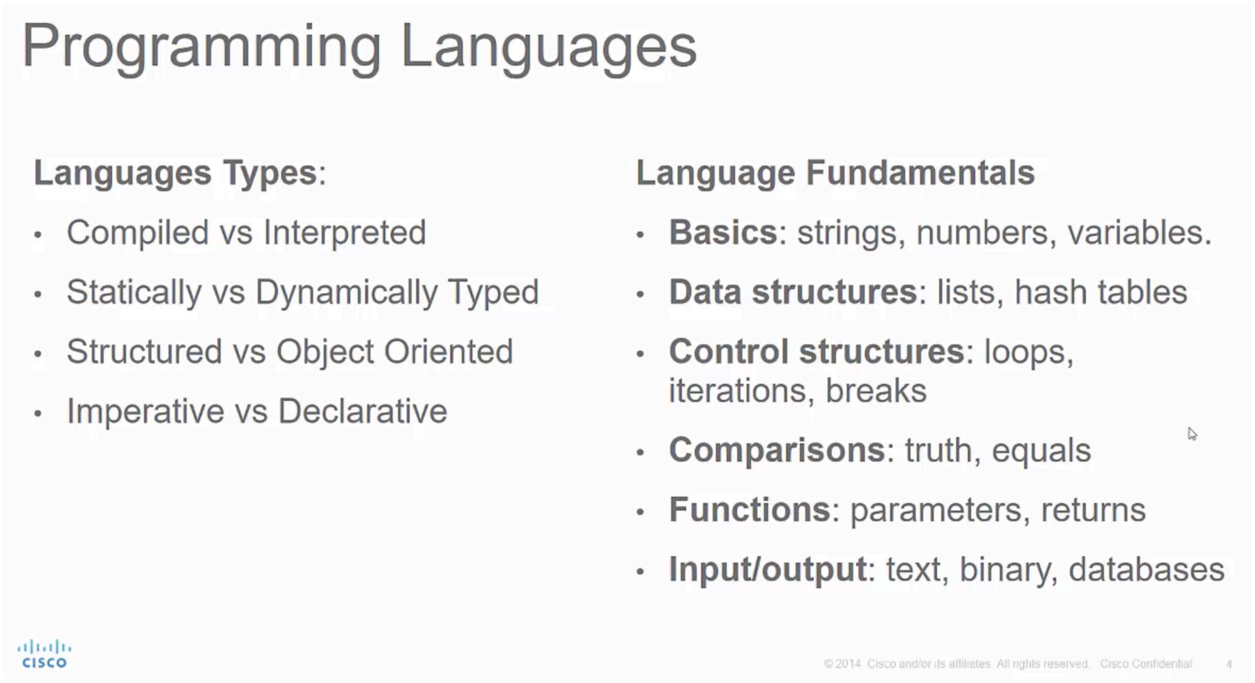

Programming Languages

Languages Types:

- Compiled vs Interpreted
- Statically vs Dynamically Typed
- Structured vs Object Oriented
- Imperative vs Declarative

Language Fundamentals

- **Basics**: strings, numbers, variables.
- **Data structures**: lists, hash tables
- **Control structures**: loops, iterations, breaks
- **Comparisons**: truth, equals
- **Functions**: parameters, returns
- **Input/output**: text, binary, databases

There are dozens of programming languages at use in the world of computing today, being used for everything imaginable. Below are some of the important characteristics of programming languages in general.

Compiled versus Interpreted Languages - Some languages are compiled into machine code appropriate for the specific machine on which it is to run. Interpreted languages do not have a compilation step, but rather are 'interpreted' by the language implementation itself, and executed line by line in this manner. And some languages can be compiled, but into 'byte code' rather than to a machine language – which is then run at execution time. Python can be run as an interpreted application. One line is executed at a time by the Python implementation, using the command *python* from the command line. Python can also be compiled into byte code – even into Java byte code, if so desired. Also, Python can be compiled into a more efficient format when performance of the Python application is critical.

Statically versus Dynamically Typed - Some languages require the programmer to explicitly declare the 'type' of every variable that they intend to use – is it integer, float, string, a complex structure? Other languages are able to determine the 'type' dynamically, when it is used. Python is a dynamically typed language.

Note
Do not confuse the discussion of static versus dynamic typing with strong versus weak typing. Python is strongly typed in that variables are determined to be a specific type once used, and must adhere to appropriate restrictions and behavior based on that type.

Structured/Procedural versus Object Oriented - Some languages are purely structural, using functions, also known as procedures, in order to organize and utilize code. Other languages are purely object-oriented (OO), meaning everything is declared as an 'object' and must be used as such. Python is definitely structured, but can also be used as an OO language, making use of classes, class methods, class data, and so on.

Imperative versus Declarative – In an imperative language, the programmer explicitly tells the system how to accomplish something. A declarative language attempts to allow the programmer to specify what is to be done, and the language figures out how to do it. In general, programs that are written using a declarative language are more concise than programs that are written in an imperative language because the programmer does not have to code every action to be taken. Python is imperative in nature, but it is possible to create a declarative system using Python.

Programming Language Basics

The following are some of the basic components of any programming language:

- **Basics**: A language needs the basic primitive objects that constitute all languages, like strings, numbers, variables, and even constant values.

- **Data structures**: A language will need to allow the programmer to store data as the application executes, and such storage can take many forms, including basic variables, but also lists, structures, and hash tables (called dictionaries in Python).

- **Control structures**: An imperative language must allow the application developer to implement the "how" part of an application, which typically involves control structures such as loops.

- **Comparisons**: A language must allow the developer to compare two or more items in order to make intelligent decisions.

- **Functions**: A structural language provides the ability to organize and isolate certain bits of intelligence into functions. These functions can be repeatedly invoked from various locations in a program.

- **Input/output**: A language must allow the programmer to interact with longer-term storage entities such as files, user input, and databases.