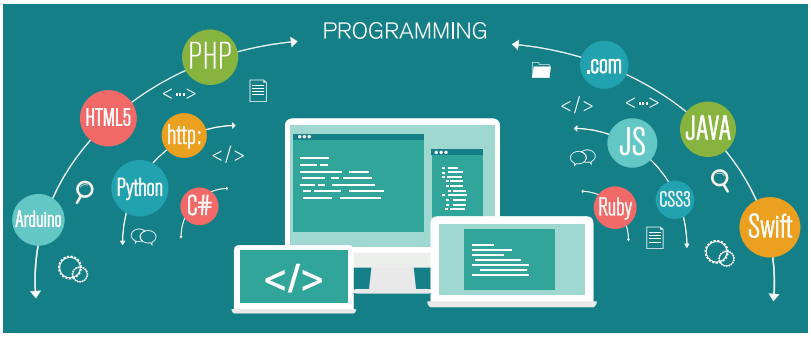
Programming Languages

THERE ARE MANY computer programming languages in use today. It is important to understand the strengths and weakness of each language to make an educated choice when selecting a computer programming language for an application.



Objective:

Summarize common computer programming languages, differentiate between programming languages and create content using programming languages.

Key Terms:

- Assembler
- Assembly Language
- Compiler
- Cross-platform Development
- Fifth generation language
- First generation language
- Fourth generation language
- Integrated Design Environment (IDE)
- Interpreter
- Mnemonic
- Object code file
- OOP (Object Oriented Programming)
- Procedural Programming
- Procedure or function
- Second generation language
- Source code file
- Third generation language

Commonly Used Programming Languages

Computer programs can be written in one of many different programming languages. Each programming language has certain key features, along with advantages and disadvantages. These need to be understood before developing an application in a programming language.

SUMMARIZE COMMON COMPUTER PROGRAMMING LANGUAGES

In this section you will look at popular programming languages in use today and understand their advantages and disadvantages.

What Is a Computer Programming Language?

A computer program contains instructions that can be executed by a computer. These instructions can be written in several different formats. The end goal of the two sets of instructions is the same. However, the first format provides instructions that are more detailed than those listed for the second set of instructions. The first set of instructions is more specific, and individual steps can be modified if needed. For example, the pot may be filled up to the three-quarters mark, instead of the half way mark as shown in the instructions. The second set of instructions seem simpler. However, this set is harder to modify because it has made assumptions along the way. For example, it is hard to ascertain the level of water in the pot, because it is not explicitly stated in the instructions. Selecting a computer programming language to solve a problem is a similar process. If the application requires very specific sets of instructions, a computer language that allows for detailed specifications is chosen, such as C or C++. If, on the other hand, the goal is to complete a task, without providing detail, a computer language that allows for high-level instructions is chosen, such as Python.

Format #1	• Go to counter. • Pick up pot. • Take pot to faucet. • Fill with water up to halfway mark in pot. • Move pot to stove. • Start stove. • Wait until water reaches 100°C.
Format #2	• Place pot with water on stove. • Heat until water starts boiling.

FIGURE 1. Here are different formats for providing instructions to boil water.

Computer Languages Classified by Generation

Computer languages have been in use since the 1940s. Since the first program was developed and executed, many computer programming languages have been developed. Computer languages are classified by generation.

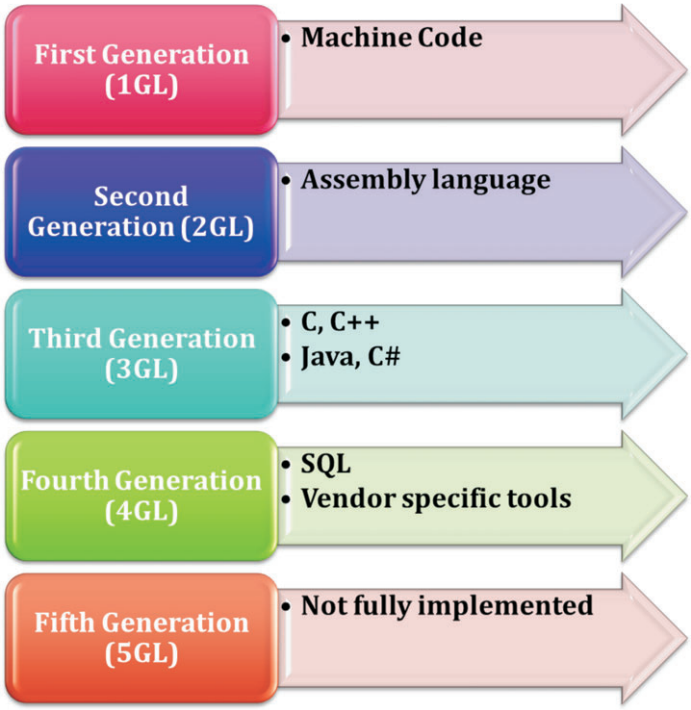


FIGURE 2. The 5 generations of computer programming languages.

First Generation Language (1GL)

When computers were first developed, programs were created in machine code. Machine code is referred as a **First Generation Language**. Machine code executed directly on a computer and is composed of instructions in binary notation, using 0's and 1's. 1GL languages are not used much anymore because they are difficult to develop and modify.

Second Generation Language (2GL)

The main difficulty with creating programs in machine code is that binary codes for instructions are difficult to remember. This lead to be creation of **Assembly language**, which is a **Second Generation Language**. In an Assembly language program, each machine code instruction is coded as a **mnemonic**. A mnemonic is written in alphabets, making it easier to read a program in Assembly than the corresponding program in machine code. See FIGURE 3. It shows mnemonics for a few machine code instructions for the x86 chip. Mnemonics are easier to remember than binary sequences. However, assembly language programs cannot execute in a computer. The statements in an assembly language program need to be converted into machine code. The program that converts an assembly language program into machine code is called an **Assembler**. Typically, 1 line of assembly language code translates to 1 machine code instruction. Moving away from 1GL languages introduces a level of complexity to the programming process. When working with machine code, we create modify and maintain a single file that contains machine code instructions. With Assembler programs, we now need to maintain 3 programs. These are the assembly language program that we coded, the Assembler program to convert our program into machine code, and the machine code program created by the Assembler program.

Machine Code Instruction	Machine Code in Hex	Assembly Language Mnemonic
1100 0110	C6	MOV
0000 0010	02	ADD
0001 0110	16	PUSH

FIGURE 3. Mnemonics for a few machine code instructions for the x86 chip.

Third Generation Language (3GL)

Developing programs in Assembly requires knowledge of the instruction set of the underlying chip, since Assembly language provides nothing more than text "short-cuts" or mnemonics for a binary machine code instruction. Assembly language mnemonics differ from chip to chip. For example, the x86 chip implements a mnemonic called MOV, while IBM Assembler uses the mnemonic MVC. Therefore, Assembly programs need to be changed as they are moved from platform to platform. In the late 1950s, **Third Generation Languages** were devel-

oped. These languages are more programmer friendly and easier to code. See FIGURE 4. It shows lines of code in Assembly language and the corresponding code in C++, a popular 3GL language. As can be seen, the C++ code is shorter and easier to read. Nearly all popular languages in use today, such as C, C++, C#, Java, and JavaScript are all considered 3GL languages. 3GL programs cannot be executed at the machine level. They need to be converted into machine code to be executed. The program that performs the conversion is called a compiler or an interpreter.

C++ code	Assembly language code

`int main()`
`{`
`  int num_classes;`
`  num_classes = 5;`
 	
`  return 0;`
`}`
	`main: # @main`
	`  push rbp`
	`  mov rbp, rsp`
	`  xor eax, eax`
	`  mov dword ptr [rbp - 4], 0`
	`  mov dword ptr [rbp - 8], 5`
	`  pop rbp`
	`  ret`

FIGURE 4. Lines of code in Assembly language and the corresponding code in C++, a popular 3GL language. As can be seen, the C++ code is shorter and easier to read.

Fourth Generation Language (4GL)

A **Fourth Generation Language** is a language that allows for creation of programs that are shorter in length than programs in a 3GL language. There is no strict definition of features that need to be implemented to label a programming language as a 4GL. Database languages, such as **SQL(Structured Query Language)** are considered 4GL languages. SQL is a vendor neutral language that is used to retrieve information from databases. It is a high-level language. SQL statements need to be converted into machine code before they can be executed, but a programmer who issues a SQL command does not need to do so; the SQL engine performs that task. Other 4GL languages in the market are vendor specific. For example, the SAS and

DIGGING DEEPER...

UNCOVERING ADDITIONAL FACTS: What Programming Languages Have Fallen Out of Favor?

New computer languages seem to be popping up every day. Some of these newer languages have replaced languages that were popular in the past. One such language is Perl. Perl was a widely used scripting language in the 80s. Its use has declined, and Python has gained popularity.

Perl is a highly flexible language and code can be written using different styles. The Perl language is based on the philosophy "There is more than one way to do it". This means that Perl programs are difficult to modify, especially if the program was written in a style that the modifier was unfamiliar with. Python, on the other hand, has rigid rules on how code should be written. This made Python easier to learn, since code could only be written in one way and made it easier to read code written by someone else.

To learn more about why Perl lost popularity visit https://www.youtube.com/watch?v=yFGdRC8XhuQ.

SPSS applications provide high-level languages of their own to perform statistical and data analysis. But one vendor's 4GL code cannot be used on another vendor's platform.

Fifth Generation Language (5GL)

Programmers develop code to solve a problem using a 3GL or 4GL. Before writing code, the programmer must break down the problem into a series of steps and write code to implement each step in the chosen programming language. A **Fifth Generation Language** can be provided high level, general input, and it generates the steps to solve the problem and the code to implement the logic. While there are a couple of 5GL languages used in the field of Artificial Intelligence, there are no fully functional, general purpose 5GL languages today. Studies have shown that a human is required to develop logic and algorithms; this step cannot be performed by a computer autonomously.

Popular Languages in Use Today

The most commonly used languages are predominantly 3GL languages, with some 4GL languages used for specialized purposes. FIGURE 5 shows the most commonly used languages of today, along with their generation and usage in the industry. Languages may be classified using different features. For example, languages may be compiled or interpreted. A language may support functional decomposition, while another language may support the Object-Oriented paradigm. These classification modes are discussed later in this E-unit.

Language	Generation	Usage
C	3GL	Operating Systems, embedded systems
C++	3GL	Games, Languages
Java	3GL	General purpose language
JavaScript	3GL	Client-side programs on the web, Server-side programs on webservers
PHP	3GL	Server-side programs on web servers
Python	3GL	General purpose language used in scientific and data analytics apps
Ruby	3GL	Server-side programs on web servers
SAS	4GL	Used for statistical analysis
SQL	4GL	Used to work with databases

FIGURE 5. The most commonly used languages, along with their generation and usage in the industry.

What Is an IDE?

When code is developed in a 3GL, it is saved in a file. This file is referred to as the **source code file**. The source code file is written using english-like words and can be modified in any text editor. The source code file is converted into machine code using a compiler or an inter-

preter. An **Integrated Design Environment** makes the process of developing programs in a 3GL and converting them into machine code easier. An IDE is a graphical tool that enhances the software development process. A C++ program written using the Visual Studio IDE can be edited using the Eclipse IDE without any issues. The underlying source code file is not dependent upon the IDE used.

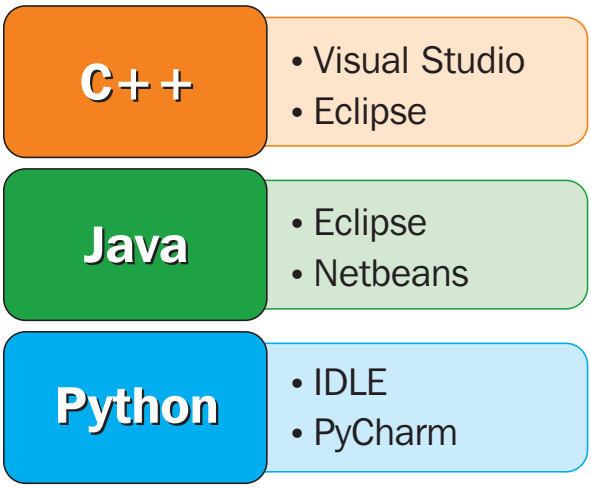


FIGURE 6. Popular IDEs in use today and the languages that they support.

DIFFERENTIATE BETWEEN PROGRAMMING LANGUAGES

In this section, you will learn to classify languages using various criteria.

Compiled vs Interpreted Languages

3GL languages are written in english-like text and stored in source code files. They cannot be executed directly in a computer and must be converted into machine code which can be executed. The conversion from source code to machine code is performed by one of 2 programs. These 2 programs are the language compiler and language interpreter.

Compiled Languages

A **compiler** is a special program that converts source code into machine code. If the compiler detects even a single error in the source code, the object code file is not created. Once machine code has been created, it is saved in a file which is referred to as the **object code file**. When a change needs to be made to the program, it is made to the source code file and the compile process is performed again to generate a new object code file. See FIGURE 7. It depicts the steps to create an object file and to modify it. The object code file gener-

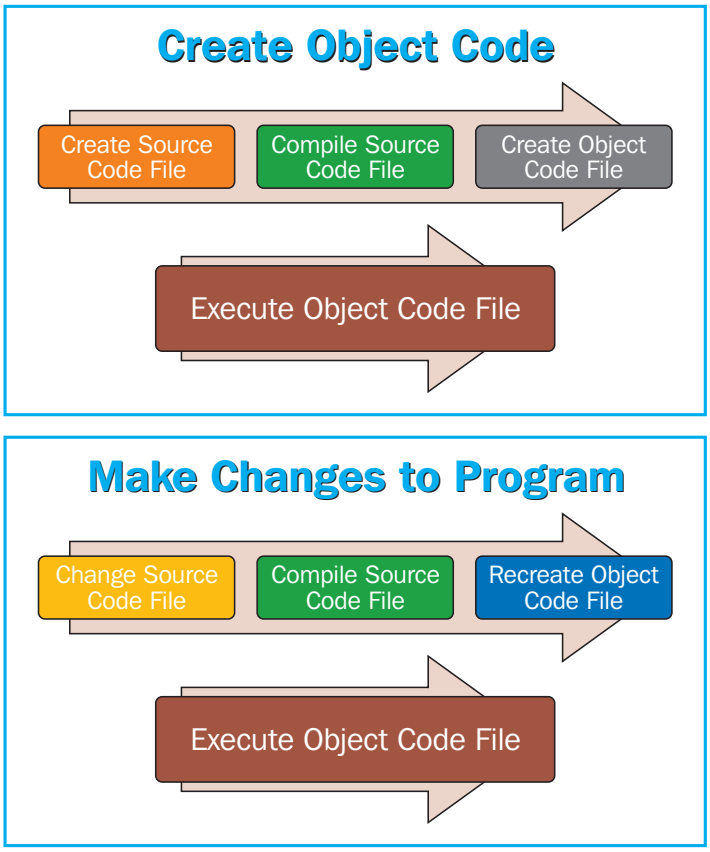


FIGURE 7. The steps to create an object file and to modify it.

ated by the compile process can be executed as many times as needed. The source code file is not needed during its execution. Most commercial applications are distributed in object code format. Since object code cannot be read—it is in machine code format—it may be distributed without fear that a user can retrieve code from it and modify it. However, if the source code file is lost or deleted, the program may no longer by modified. Therefore, source code control mechanisms need to be in place to ensure that source code is maintained safely. Compilers are platform specific. For example, a C++ compiler written for the Windows platform produces machine code that can only run on a Windows machine. To create object code for a Linux machine, the C++ source code must be recompiled on the Linux machine to create object code for the Linux platform.

Interpreted Languages

Interpreted languages use a different approach. An interpreter reads only one line at a time from the source code file. If the line is error free, it converts the line into machine code and executes it, before proceeding to the next line in the source code file. When the interpreter finds an error in the source code, the translation to machine code stops. The generated object code is not stored. This is shown in FIGURE 8. When the program needs to be re-executed, the interpreter, reads the source code once again and translates a source code into machine code and then executes it line by line. The translation process is performed each time the pro-

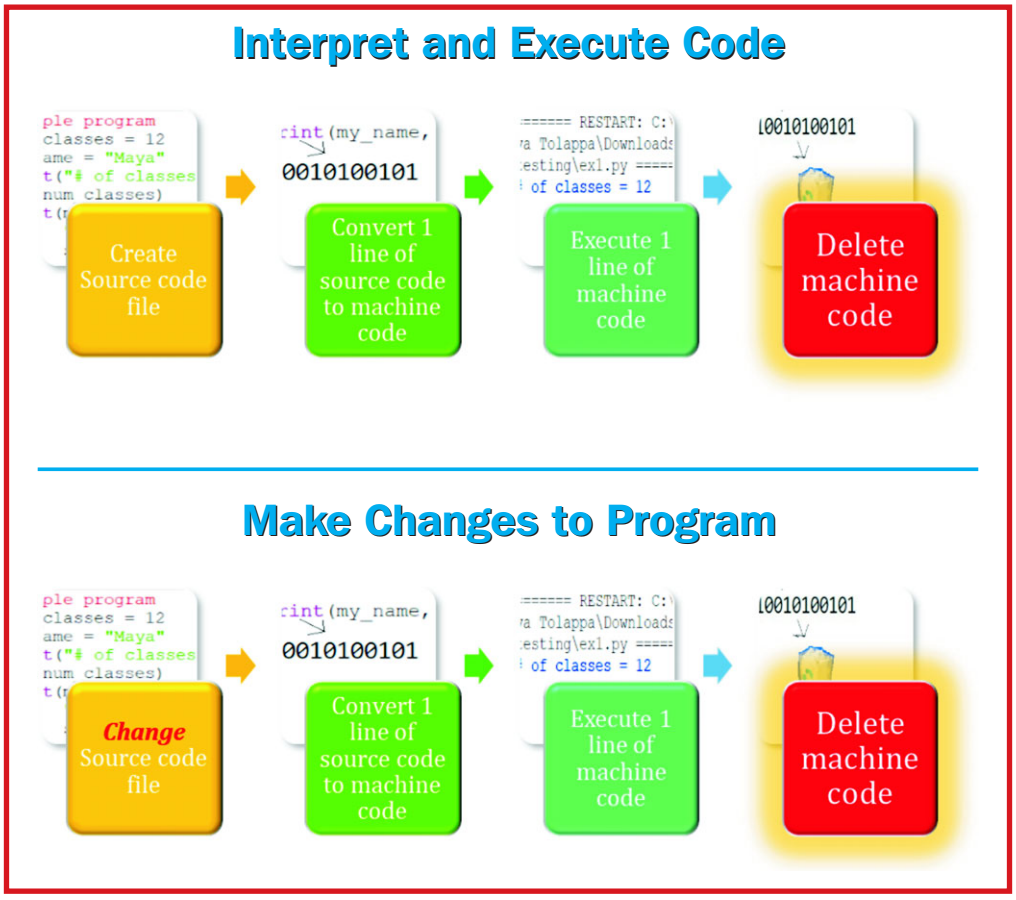


FIGURE 8. How does an interpreter work?

gram executes. When a program needs to be modified, changes are made to the source code file and the interpreter converts it into machine code, line by line, and executes each individual line of machine code. FIGURE 9 compares compilers and interpreters. The main advantage of compiled programs is that once the object code file has been created, executing it is very fast. Maintaining and testing interpreted programs is easier, since there is only 1 file to maintain. The Java programming language uses a combination of both a compiler and an interpreter to create applications. A Java source code file is compiled into a bytecode file, which is half-way between source code and object code. The bytecode file is platform independent and can be

	Compiler	Interpreter
Speed of conversion to machine code	Creating machine code is time consuming.	Creating machine code one line at a time is time consuming.
Speed of execution	Once object code has been created, it executes at high speed.	*Since translation to machine code is performed each time, execution is slower than a compiler.*
Ease of use	*Compilation fails even if there is a single mistake in source code file.*	Process proceeds until an error is encountered, making testing easier.
File maintenance issues	*Need to maintain two files—source code file and object code file.*	Only need to maintain one file—the source code file.

FIGURE 9. Comparison of compilers and interpreters.

used on both Windows and Linux platforms. Bytecode is not 100% machine code. It is converted into machine code at run time using an interpreter called the Java Virtual Machine. There are Java virtual machines for nearly all platforms today. Most of the languages used in web development are interpreted. See FIGURE 10 for classification of web development languages. Java is considered a compiled and an interpreted language since it is compiled to bytecode which is interpreted. C#, which is developed by Microsoft, is compiled while the remaining languages are interpreted.

Language	Interpreted	Compiled
C#		☑
Java	☑	☑
JavaScript	☑	
PHP	☑	
Python	☑	
Ruby	☑	

FIGURE 10. Classification of web development languages. Java is considered a compiled and an interpreted language since it is compiled to bytecode which is interpreted. C#, which is developed by Microsoft, is compiled while the remaining languages are interpreted.

Procedural vs. Object Oriented Languages

Computer programs are written to solve problems. A problem is broken down into the input information provided to the programmer, and the process required to transform the input values into the required output values. The process used to perform transformation may follow one of 2 approaches. The 2 approach are referred to as Procedural programming and Object Oriented Programming.

Procedural Programming

In the early days of programming, developing solutions to problems followed a top-down approach. A problem was broken into smaller pieces in a repetitive process, until a piece was a problem that could be solved using program code. The piece, which was a portion of the problem to be solved, was referred to as a **procedure or function**. The problem was eventually solved by assembling all the procedures that were written and executing them in the correct order. Consider the logic to create a password based upon the first character of the first name, last character of the last name, and a random number. FIGURE 11 shows this problem broken into 4 procedures. Obtain the first name and return its first character. Obtain last name and return the last character. Generate a random number. Concatenate the values returned by the 3 procedures and return it. The order in which these functions are invoked or used is documented in a hierarchy chart. The process to determine the result, user id, emphasizes the process that needs to be followed. Therefore, this approach is referred to as **procedural programming**. Most 3GL languages, such as C, C++, and JavaScript support functional programming. It is possible to create complex functions that take in multiple inputs and gener-

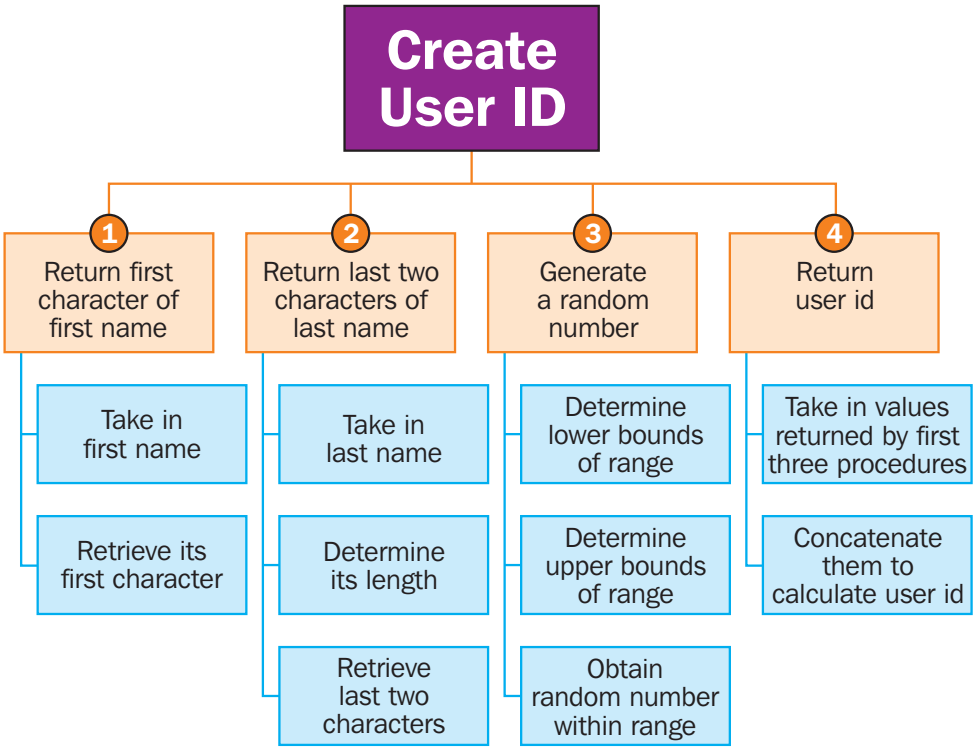


FIGURE 11. Logic to create a password.

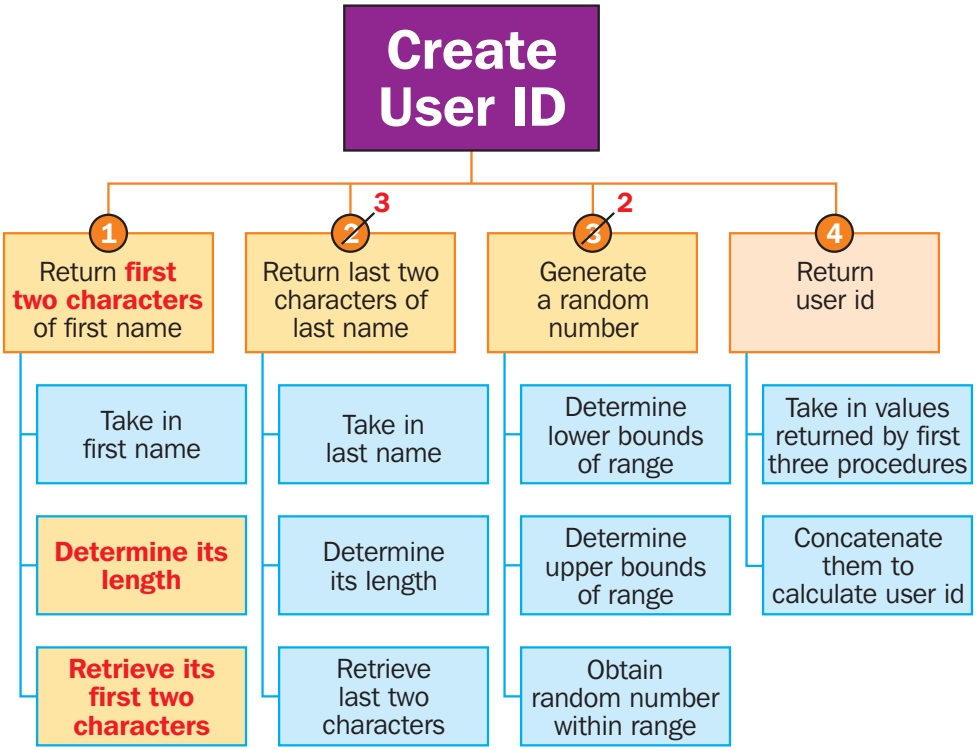


FIGURE 12. How are changes made to procedures?

ate outputs. However, procedural programming has drawbacks. They are difficult to modify and changes higher up in the hierarchy chart of functions may requires major changes in functions that are downstream in the hierarchy chart. For example, consider our password creation algorithm. Suppose that we wish to change the logic to create the password so that the first 2 characters of the first name are concatenated to a random number, followed by last 2 characters of the last name. This requires changes in some of the functions, along with the order in which they are used as shown in FIGURE 12. Another drawback with procedural programming is that heavy emphasis is placed on process while less attention is paid to data. This is contrary to how organizations behave. The most important asset in an IT shop is the data. Procedural programming values processes over data.

Object Oriented Programming Languages

As disadvantages of procedural programming became more apparent, there was a push to develop new paradigms for developing programs. This led to the development of **OOP (Object Oriented Programming)** philosophy, culminating in the development of the Simula and Smalltalk programming languages. In OOP, primary emphasis is placed on objects. Objects represent entities in the computer application. For example, in the password example, there can be an object named password. In a college environment, there will be objects to represent students, teacher, classrooms, etc. Any "thing" that needs to be incorporated into a computer application is an object. FIGURE 13 shows objects that are likely to be modeled in an application that runs in a school. Each object is represented as an oval shape. Objects interact with one another and this is depicted using arrows in the diagram. Objects also serve as reposi-

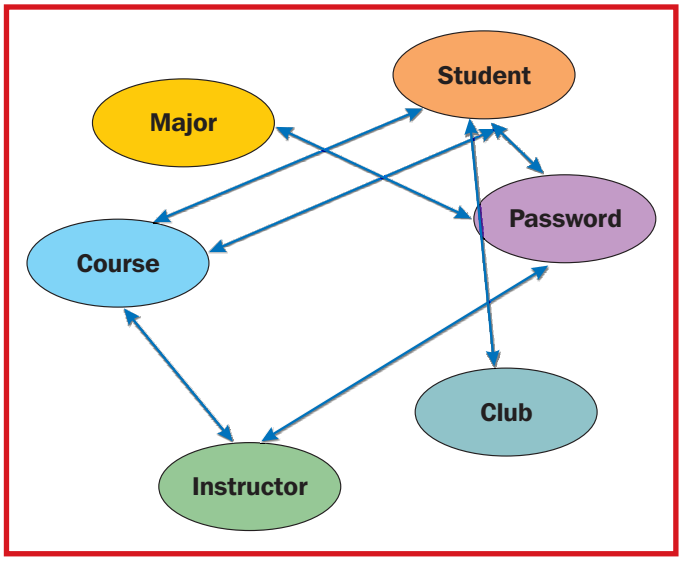


FIGURE 13. Objects that are likely to be modeled in an application that runs in a school. Each object is represented as an oval shape. Objects interact with one another and this is depicted using arrows in the diagram.

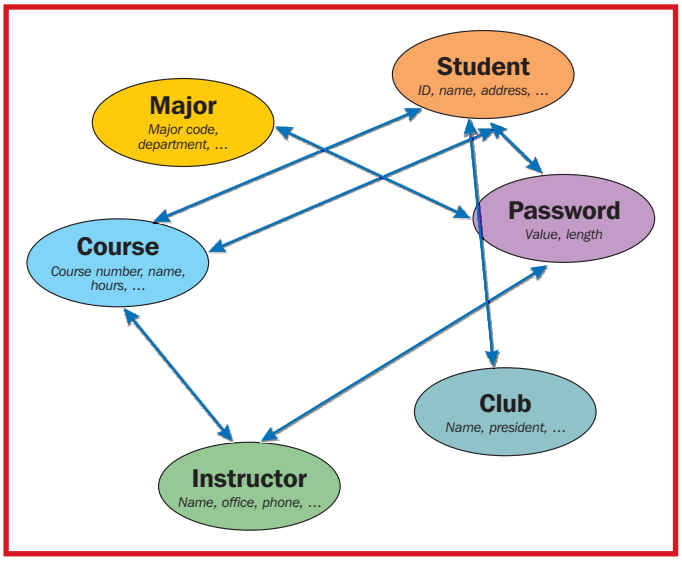


FIGURE 14. Objects in a college application, along with their data points. In OOP, an object owns and controls its data.

tories of data. FIGURE 14 shows objects, along with their data points. In OOP, an object owns and controls its data. FIGURE 15 lists languages used in web development and their OOP behavior. Java is a truly OOP language and is completely object oriented. The other languages support procedural programming and OOP.

Language	Supports Procedural Programming	Supports OOP Programming
Java		☑
JavaScript	☑	☑
PHP	☑	☑
Python	☑	☑

FIGURE 15. OOPs support in web development languages.

Languages Used in Web App and Mobile App Environments

Web apps are platform independent since they run in a browser window. A web app performs the same in a Windows machine, a Mac machine, an Android phone and an iPhone. This is because web apps are developed in HTML, CSS (Cascading Style Sheets) and JavaScript, and all 3 of these languages are interpreted by the browser software. Mobile apps, on the other hand, are not universal. A mobile app on an iPhone may look the same as the app on an Android phone, but the code run-

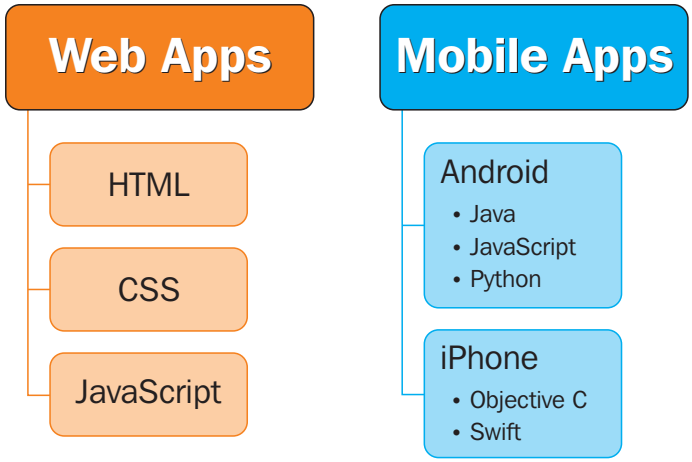


FIGURE 16. Programming languages used in web apps and mobile apps.

FURTHER EXPLORATION...

ONLINE CONNECTION:
Creating Android Apps Using App Inventor for Android

Creating mobile apps for an Android can be fun using the App Inventor for Android software app. This app was developed by Massachusetts Institute of Technology (MIT) as a method to introduce young students to computer programming. It provides a block-based environment to develop code. You can create a fully functional app by dragging and dropping blocks to implement logic. App Inventor for Android allows non-programmers to learn about programming, app development, and logic implementation. MIT plans to release the App Inventor for iOS app soon.

To learn more about App Inventor for Android, visit https://appinventor.mit.edu/explore/index-2.html.

ning on the two phones are completely different. This is because the languages used to develop mobile apps are platform dependent. FIGURE 16 shows programming languages used in web apps and mobile apps. There are numerous programming languages to choose from when developing Mobile apps. The most popular language for developing Android apps is Java. The most popular languages for developing apps for the iPhone are and Objective C and Swift.

CREATE CONTENT USING PROGRAMMING LANGUAGES

Content creation for web apps and mobile apps are significantly different because different languages are used for each of the two. When learning a new programming language, it is customary to create a program that displays the text "Hello World", and the program is referred to as the "Hello World" program. Accordingly, this section develops the HelloWorld web app and mobile app.

Web Apps

Web apps are rendered on the display device with browser software. The page displayed to the user is a result of 3 technologies. FIGURE 17 shows the tasks performed by each of the 3 technologies. All web apps are required to have HTML content. Additionally, CSS code may be included to specify how the HTML should be displayed. Any content on the page that is dependent upon logic, such as displaying date, or changing the background image based upon time of

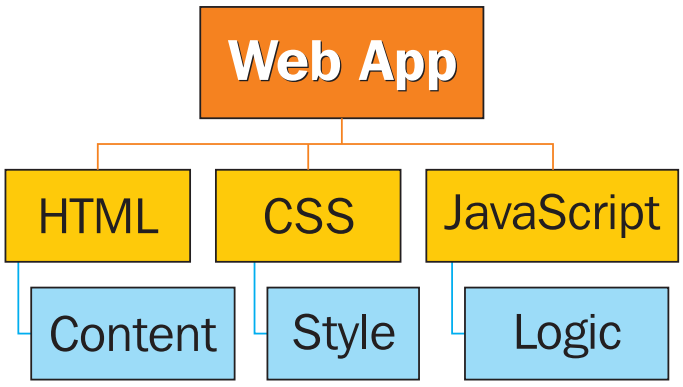


FIGURE 17. Tasks performed by each of the 3 technologies.

day, must be implemented using code in the JavaScript programming language. A web app can be coded in a single page or can be broken down into multiple files. Good programming practice indicates that HTML, CSS and JavaScript be maintained in separate files. This allows for reuse of files in other web apps. See FIGURE 18. It shows code for a web app that displays the text "Hello World" in the browser and the color of the text changes when the mouse moves over the text . This web app's output is not exciting, but demonstrates the code for implementing the app. To keep the example simple, the html file for the web app includes the CSS code and the JavaScript code. The web app is implemented in a single file called web_app.html. Lines 8-20 contain CSS code and lines 21-26 contain JavaScript code. The logic in the JavaScript code populates the text to be placed inside a div element called h1Area. Web app do not need to be developed using specialized software. Code for web apps can be written using a free editor such as Notepad which ships with all Windows computers. As can be seen in the code, web apps require knowledge of HTML, CSS and JavaScript.

web_app.html Code

```
1  <!doctype html>
2
3  <html lang="en">
4  <head>
5      <meta charset="utf-8">
6
7      <title>Hello World Web App!</title>
8      <style>
9          body {
10             background-color: khaki;
11         }
12
13         h1 {
14             color: green;
15             text-shadow: 3px 3px 3px silver;
16         }
17         h1:hover {
18             color:red;
19         }
20     </style>
21     <script >
22         function displayMessage(){
23             document.getElementById('h1Area').innerHTML =
24                 "<h1>Hello World!</h1>"
25         }
26     </script>
27 </head>
28
29 <body onload ="displayMessage();">
30     <div id = "h1Area">
31     </div>
32 </body>
33 </html>
```

CSS Code

JavaScript Code

Output in browser

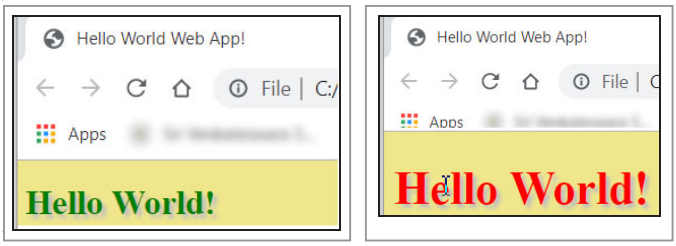


FIGURE 18. Code for a web app that displays the text "Hello World" in the browser and the color of the text changes when the mouse moves over the text . This web app's output is not exciting, but demonstrates the code for implementing the app.

Mobile Apps

Mobile apps for an Android and an iPhone are developed using different languages, IDE and methodologies.

Android Platform

Android devices run an operating system called Android Operating System. The most popular language for developing apps on Android is Java. The Java compiler creates bytecode and there is a Java Virtual Machine for the Android platform that will interpret bytecode into machine language. When the Java language was developed in 1996, mobile apps had not been invented. Therefore, the language does not natively support Android features. However, there are libraries that can be used to make use of functionality needed to create a mobile app. Android apps are developed in an IDE called Android studio that can be downloaded from https://developer.android.com/studio for free. This installs the IDE (Integrated Development Environment) for development of Java Android Apps. Even for a simple mobile app, many different files are created in Java and XML. As can be seen, creating a mobile app for the Android platform required strong skills in Java and a knowledge of the XML (Extensible Markup Language) language.

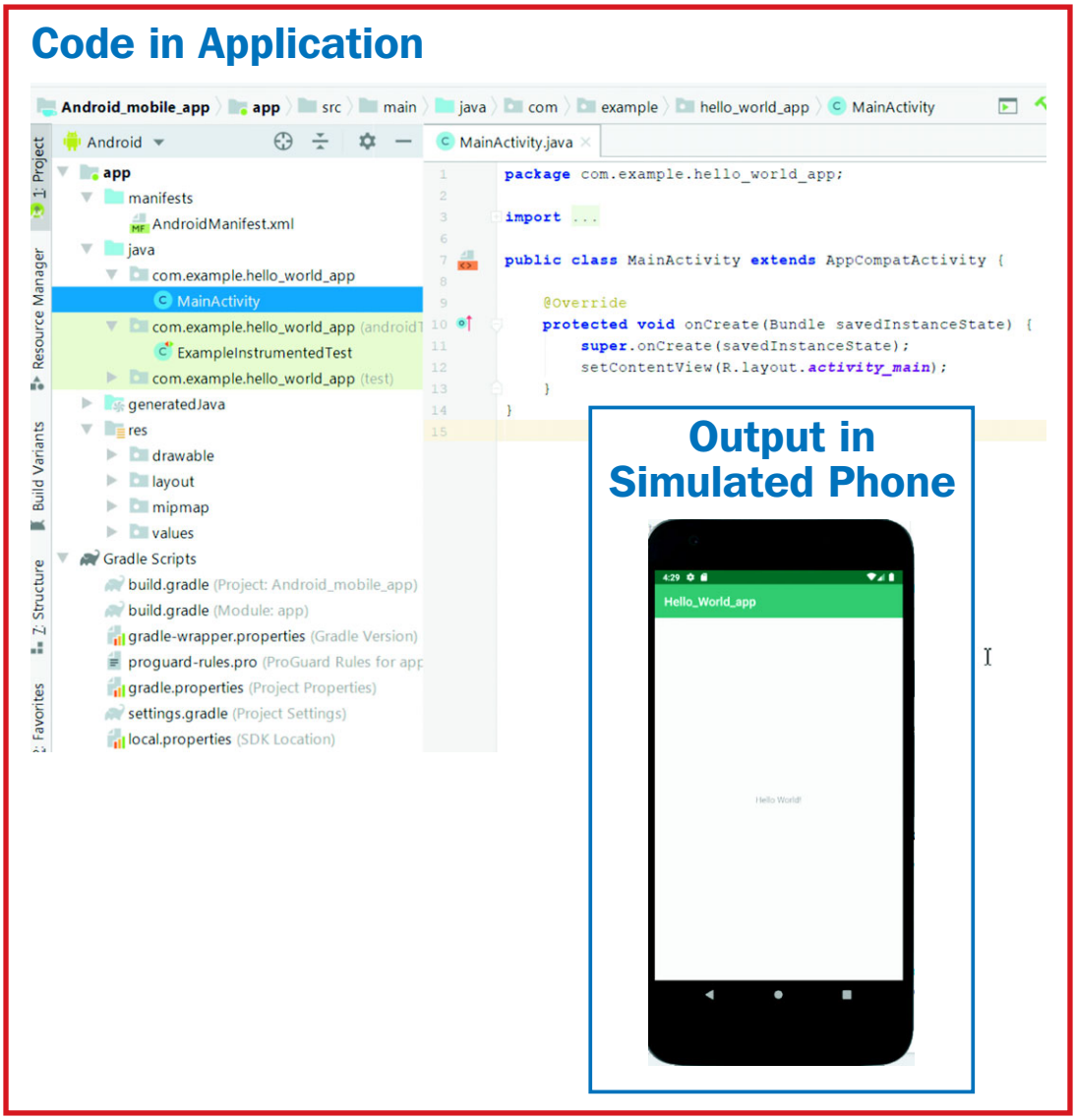


FIGURE 19. The HelloWorld project in the Android Studio App.

iPhone Platform

iPhone mobile apps can only be developed on a Mac using the Xcode IDE. iPhone apps can be developed using the Objective-C or Swift language. Developing an iPhone HelloWorld app is like the steps followed for the Android app. The IDE performs the background work to bring in libraries that are needed to create a mobile app. Creating apps for the iPhone requires knowledge of Objective-C, which is a C variant that supports objects. However, it is different from industry standard C and C++. Knowledge of C and C++ is needed to learn Objective-C.

Code in the Application

(taken from https://www.appcoda.com/hello-world-build-your-first-iphone-app/)

Output in Simulated Phone

FIGURE 20. Screens with Objective-C, along with output screens on a phone simulator in Xcode, the IDE used to develop iPhone apps. (Screenshots were copied from https://www.appcoda.com/hello-world-build-your-first-iphone-app/.)

Cross-platform Development for Mobile Apps

Creating apps for the Mobile arena can be time consuming since native Android code and native iOS code are incompatible with one another. There has been a movement in the indus-

try towards a develop-once-deploy-everywhere model. This is referred to as **Cross-platform Development**. There are several products that allow for creating a mobile app once and deploying it on both the Android and iOS platform.

Tool Name	Language	IDE
React	JavaScript	
Xamarin	C#	Visual Studio
PhoneGap	JavaScript	
JQuery Mobile	JavaScript	

FIGURE 21. Popular cross-platform development frameworks and the language used in each of the platforms.

Summary:

There are many programming languages in use today. Understanding the pros and cons of each language is important because application development and performance are dependent upon the language that the application is coded in. Additionally, not all languages run on all platforms. Once an application has been deployed, changing the language that the program has been developed with requires rewriting the entire application. Therefore, acquiring a deep knowledge about computer languages is an important skill.

Checking Your Knowledge:

1. Why are 3GL languages more popular than 2GL languages?
2. Why are compiled programs more efficient that interpreted programs?
3. What led to the popularity of interpreted languages?
4. Why are web apps easier to deploy?
5. Why are cross-platform mobile development tools used?

Expanding Your Knowledge:

Use the knowledge in this E-unit to create a presentation explaining the language you will choose to pursue further. Explain your reasoning and indicate the types of applications you will develop (web apps, server-side apps or mobile apps).

Web Links:

Hello World! Build Your First IPhone App
https://www.appcoda.com/hello-world-build-your-first-iphone-app/

What Is a Generation Language
https://www.computerhope.com/jargon/num/1gl.htm

X86 Opcode and Instruction Reference Home
http://ref.x86asm.net/coder32.html#xC6