**What we are going to learn In Week 1**

Congrats. You made it to the end of the first module. Great job. You've taken the first steps to learning a new programming language, and growing your IT skillset. Getting there shows real determination and a will to learn. We've covered a lot of topics, and many might be new to you if you've never learned about programming before. You've discovered what scripting is, what the syntax and semantics of a programming language are all about, and how they relate to automation. We've got to grip some small blocks of Python code, talked about why Python is relevant to IT, and explored what other programming languages are available. We’ve had our first approach to how to input data, and write a script that puts this data to use, and we've seen how you can use Python to perform typical math calculations. Not bad for your first Python steps, right

**What is Programming**

In general, you can think of scripts as programs with a short development cycle that can be created and deployed rapidly. In other words, a script is a program that is short, simple, and can be written very quickly.

**SYNTAX** is the rules for how each instruction is written and the

**SEMANTICS** is the effects the instructions have. Much like spoken languages

**What is Automation**

Automation is the process of replacing a manual step with one that happens automatically. Take a traffic light for example, which continuously regulates the flow of vehicles at an intersection. A traffic light requires human intervention only when it needs repairs or maintenance

**Uses for Automation**

Scripts can be used for automating specific tasks. Automation is used to replace a repetitive manual step with one that happens automatically. Humans are fallible. They can become tired, make mistakes, fail to follow instructions, be inconsistent in their job performance,.

**Appropriate uses for automation include:**

* The automatic timing and regulation of traffic lights
* A repetitive task that is at high risk for human error
* Sending commands to a computer
* Detecting and removing duplicates of data
* Sending automated emails that are personalized by pulling individual names from a database and plugging them into the email
* Updating a large number of file permissions
* Reporting on system data, like disk or memory usage
* Installing software
* Generating reports
* Deploying a file or a computer program to all computers on a company network
* Using a configuration management system to deploy software patches, after a human has designed the system
* Populating an e-commerce site with products
* Setting the home directory and access permissions for users

**Automation is not always an appropriate or complete solution**

Automation cannot perform all human work. Tasks that call for human creativity, social connection, psychology, flexibility, ingenuity, evaluation, and/or complex analytic work are not good candidates for full automation. Sometimes automation can be used to perform one or more subtasks of a larger set of tasks – but – human intervention is required to complete the tasks. The following are some examples of tasks that cannot or should not be fully automated:

* Items that require human evaluation and analytic skills:
* Designing a configuration management system
* Investigating and troubleshooting all end user problems
* Writing a computer program
* Building a new startup business
* Items that require human creativity and/or an eye for aesthetic qualities:
* Designing an attractive webpage (AI can do this, but simple automation cannot)
* Wedding photography
* Haircuts and styling
* Items that cannot be automated due to basic physics:
* Troubleshooting or repairing machines that cannot power on or boot up
* Items that need human interaction, psychology, and/or evaluation skills:
* Interviewing and hiring new employees
* Customer service (chat bots cannot address every customer service need)
* Items that should not be fully automated due to costs and safety:
* Grocery store checkout process, including bagging groceries
* Tasks that are less expensive to perform manually

**Artificial Intelligence**

It is important to understand that basic automation is not the same as artificial intelligence. Automation is used to explicitly instruct a machine on how to perform a task. Artificial intelligence (AI) involves training a computing machine to perform more complex tasks through a process called machine learning. This process prepares the AI software to perform new tasks without a human needing to program explicit instructions for each task. Although AI is often used for automating human tasks, AI automation is much more complex than basic automation.

# Getting Computers to Work for You

because when you automate a task you avoid the possibility of human errors, and reduce the time it takes to do it. Imagine this scenario: your company had a booth at a recent conference and has gathered a huge list of emails from people interested in learning more about your products.

Automating tasks allows you to focus on projects that are a better use of your time, letting computers do the boring stuff for you. Learning how to program is the first step to being able to do this

.

What things would you like to automate using programming?

# Study Guide: Introduction to Programming

# Key Terms

* **Programming code** - Programming code is a set of written computer instructions, guided by rules, using a computer programming language. The instructions tell computers and machines how to perform an action. Programming code may also be referred to as source code or scripts.
* **Programming languages** - Programming languages are similar to human spoken languages in that they both use syntax and semantics. Programming languages are used to write computer programs.  Some common programming languages include Python, Java, C, C++, C#, and R.
* **Syntax** - Syntax is a set of rules for how statements are constructed in both human and computer languages.
* **Semantics** - Semantics refers to the intended meaning or effect of statements, or collections of words, in both human and computer languages. Semantic errors are also referred to as logical errors.
* **Computer program** - A computer program is a step-by-step list of instructions that a computer follows to reach an intended goal. It is important to be clear and precise about the actions a computer program is supposed to perform because computers will do exactly what they are instructed to do
* **Script** - Scripts are usually shorter and less complex than computer programs. Scripts are often used to automate specific tasks. However, they can be used for complex tasks if needed. Scripts are often written by IT professionals, but anyone can learn to write scripts. Scripts have a shorter, less structured development cycle as compared to the development of complex computer programs and software. Scripts can be written in a variety of programming languages, like Python, Javascript, Ruby, Bash, and more. Some scripting languages are interpreted languages and are only compatible with certain platforms.
* **Automation** - Automation is used to replace a repetitive manual step with one that happens automatically.
* **Output** - Output is the end result of a task performed by a function or computer program. Output can include a single value, a report, entries into a database, and more.
* **Input** - Input is information that is provided to a program by the end user. Input can be text, voice, images, biometrics, and more.
* **Functions** - A function is a reusable block of code that performs a specific task.
* **Variables** - Variables are used to temporarily store changeable values in programming code.

**What is python**

the Python interpreter is the program that reads what is in the recipe and translates it into instructions for your computer to follow

# Python Resources

## **More About Python**

### Using Python on your own

The best way to learn any programming language is to practice it on your own as much as you can. If you have Python installed on your computer, you can execute the interpreter by running the python3 command (or just python on Windows), and you can close it by typing exit() or Ctrl-D.

If you don’t already have Python installed on your machine, that’s alright. We’ll explain how to install it in an upcoming course.

### Python practice resources

In the meantime, you can still practice by using one of the many online Python interpreters or codepads available online. There’s not much difference between an interpreter and a codepad. An interpreter is more interactive than a codepad, but they both let you execute code and see the results.

Below, you’ll find links to some of the most popular online interpreters and codepads. Give them a go to find your favorite.

* <https://www.python.org/shell/>
* <https://www.onlinegdb.com/online_python_interpreter>
* <https://repl.it/languages/python3>
* <https://www.tutorialspoint.com/execute_python3_online.php>
* <https://rextester.com/l/python3_online_compiler>
* <https://trinket.io/python3>

### Additional Python resources

While this course will give you information about how Python works and how to write scripts in Python, you’ll likely want to find out more about specific parts of the language. Here are some great ways to help you find additional info:

* Read the [official Python documentation](https://docs.python.org/3/).
* Search for answers or ask a question on [Stack Overflow](https://stackoverflow.com/).
* Subscribe to the Python [tutor](https://mail.python.org/mailman/listinfo/tutor) mailing list, where you can ask questions and collaborate with other Python learners.
* Subscribe to the [Python-announce](https://mail.python.org/mailman/listinfo/python-announce-list) mailing list to read about the latest updates in the language.

### Python history and current status

Python was released almost 30 years ago and has a rich history. You can read more about it on the [History of Python](https://en.wikipedia.org/wiki/History_of_Python) Wikipedia page or in the section on the [history of the software](https://docs.python.org/3.0/license.html) from the official Python documentation.

Python has recently been called the fastest growing programming language. If you're interested in why this is and how it’s measured, you can find out more in these articles:

* [The Incredible Growth of Python](https://stackoverflow.blog/2017/09/06/incredible-growth-python/) (Stack Overflow)
* [Why is Python Growing So Quickly - Future Trends](https://www.netguru.com/blog/why-python-is-growing-so-quickly-future-trends) (Netguru)
* [By the numbers: Python community trends in 2017/2018](https://opensource.com/article/18/5/numbers-python-community-trends) (Opensource.com)
* [Developer Survey Results 2018](https://insights.stackoverflow.com/survey/2018#technology) (Stack Overflow)

# A Note on Syntax and Code Blocks

When writing code, using correct syntax is critical. Even a small typo, like a missing parenthesis bracket or an extra comma, can cause a syntax error and the code won't execute at all. If your code results in an error or an exception, pay close attention to syntax and watch out for minor mistakes. A single wrong character could take hours to identify in long code so it is important to be mindful of syntax when writing code.

## **Common syntax errors:**

* Misspellings
* Incorrect indentations
* Missing or incorrect key characters:
  + Bracket types - ( curved ), [ square ], { curly }
  + Quote types - "straight-double" or 'straight-single', “curly-double” or ‘curly-single’
  + Block introduction characters, like colons - :
* Data type mismatches
* Missing, incorrectly used, or misplaced Python reserved words
* Using the wrong case (uppercase/lowercase) - Python is a case-sensitive language

If your syntax is correct, but the script has unexpected behavior or output, this may be due to a semantic problem. Syntax is like the vocabulary, grammar, spelling, and punctuation of code. Semantics are the meaning and logic of coded statements. It is possible to have syntactically correct code that runs successfully, but doesn't do what we want it to do.

## **Common semantic errors:**

* Creating functional code, but getting unintentional output
* Poor logic structures in the design of the code

When working with the code blocks in exercises for this course, be mindful of syntax and semantic (logic) errors, along with the overall result of your code. Just because you fixed an error doesn't mean that the code will have the desired effect when it runs! Once you’ve fixed an error in your code, don't forget to click Run to check your work.

# Why is Python relevant to IT?

Python makes it easy to express the fundamental concepts of programming like data structures and algorithms with easy to read syntax.

.

Its first version was released by Guido van Rossum back in 1991

It's also because there's more tools available in Python for a growing range of applications.You can use Python to calculate statistics, run your e-commerce site, process images, interact with web services, and do a whole host of other tasks. Python is perfect for automationmThat's why Python is the language of choice for lots of people working in IT support, system administration, and web development

# Other Languages

Although we picked Python for this course, There are platform-specific scripting languages like PowerShell

which is used on Windows, and Bash which is used on Linux. Both are widely used by system administrators on those platforms. There are also general-purpose scripting languages similar to Python, like Perl or Ruby, which are also widely used for scripting and automation. JavaScript, which was originally developed as a client-side scripting language for the web, is increasingly used server-side for a broader set of tasks.

# Study Guide: Introduction to Python

This study guide provides a quick-reference summary of what you learned in this lesson and serves as a guide for the upcoming practice quiz.

In this segment, you learned that Python is a general purpose programming language that is commonly used for scripting and automation, as well as to develop a wide variety of applications. Python is compatible with most operating systems, including Windows, Linux, and Mac OS, and is updated every few years. Python can also run on a variety of machines, such as servers, workstations, PCs, mobile devices, IoT, and more.

Python is widely used in the IT field, including IT support, system administration, web development, machine learning, data analytics, and more. Python can be used to calculate statistics, run your e-commerce site, process images, interact with web services, and do a whole host of other tasks. Python instructions resemble the English language, which is what makes it easier to learn and understand when compared to other programming languages.

**Python is:**

* a general purpose scripting language;
* a popular language used to code a variety of applications;
* a frequently used tool for automation;
* a cross-platform compatible language;
* a beginner-friendly language.

**Python is not:**

* a platform-specific / OS-specific scripting language;
* a client-side scripting language;
* a purely object-oriented programming language.

## **Code comparison with Python**

You will be learning about both Python and Bash scripting in this program. The following code illustrates a syntax difference between the two languages:

| **Print to screen in Python** | **Print to screen in Bash** |
| --- | --- |
| **>> print("Hello, how are you?")**  **Hello, how are you?** | **>> echo Hello, how are you?**  **Hello, how are you?** |

# Key Terms

* **Platform-specific / OS specific scripting language** - Platform-specific scripting languages, like PowerShell (for Windows) and Bash (for Linux), are used by system administrators on those platforms.
* **Client-side scripting language** - Client-side scripting languages, like JavaScript, are used mostly for web programming. The scripts are transferred from a web server to the end-user’s internet browser, then executed in the browser.
* **Machine language** - Machine language is the lowest-level computer language. It communicates directly with computing machines in binary code (ones and zeros). In binary code, one equals a pulse of electricity and zero equals no electrical pulse. Machine language instructions are made from translating languages like Python into complex patterns of ones and zeros.
* **Cross-platform** **language** - Programming language that is compatible with one or more platforms / operating systems (e.g., Windows, Linux, Mac, iOS, Android).
* **Object-oriented programming language** - In object-oriented programming languages, most coding elements are considered to be objects with configurable properties. For example, a form field is an object that can be configured to accept only dates as input in the mm/dd/yy format, and can be configured to read from and write to a specific database.
* **Python interpreter -** An interpreter is the program that reads and executes Python code by translating Python code into computer instructions.

# Resources

For additional Python practice, the following links will take you to several popular online interpreters and codepads:

* [Welcome to Python](https://www.python.org/shell/)
* [Online Python Interpreter](https://www.onlinegdb.com/online_python_interpreter)
* [Create a new Repl](https://repl.it/languages/python3)
* [Online Python-3 Compiler (Interpreter)](https://www.tutorialspoint.com/execute_python3_online.php)
* [Compile Python 3 Online](https://rextester.com/l/python3_online_compiler)
* [Your Python Trinket](https://trinket.io/python3)

It's because Print is a Python function that writes what we tell it to on the screen. Functions are pieces of code that perform a unit of work. We'll talk a lot more about functions later on, and you'll even learn how to write your own. Keywords are reserved words that are used to construct instructions. These words are the core part of the language and can only be used in specific ways.

Python : As we called out, the keywords and functions used in Python are what makes up the syntax of the language

# Study Guide: First Programming Concepts

This study guide provides a quick-reference summary of what you learned in this lesson and serves as a guide for the upcoming practice quiz.

# Functions

A function is a piece of code that performs a unit of work. In the examples you've seen so far, you have only encountered the **print()** function,

1

2

3

# Syntax for printing a string of text. Click Run to check the result.

print("Hello world!")





RunReset

# Keywords

A keyword is a reserved word in a programming language that performs a specific purpose. In your first Python example, you briefly encountered the keywords **for** and **in**. Note that keywords will often appear in **bold** in this course.

In the next few weeks, you will also learn the following keywords:

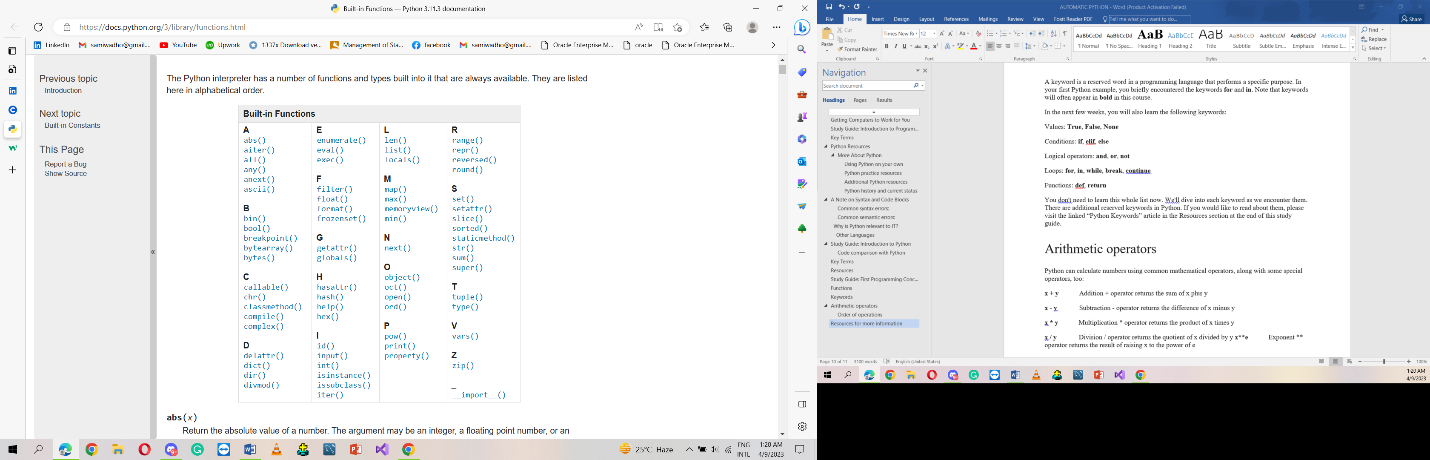
Values: **True**, **False**, **None**

Conditions: **if**, **elif**, **else**

Logical operators: **and**, **or**, **not**

Loops: **for**, **in**, **while**, **break**, **continue**

Functions: **def**, **return**



You don't need to learn this whole list now. We'll dive into each keyword as we encounter them. There are additional reserved keywords in Python. If you would like to read about them, please visit the linked “Python Keywords” article in the Resources section at the end of this study guide.

# Arithmetic operators

Python can calculate numbers using common mathematical operators, along with some special operators, too:

**x + y**            Addition + operator returns the sum of x plus y

**x - y**             Subtraction - operator returns the difference of x minus y

**x \* y**            Multiplication \* operator returns the product of x times y

**x / y**             Division / operator returns the quotient of x divided by y

**x\*\*e**            Exponent \*\* operator returns the result of raising x to the power of e

**x\*\*2**            Square expression returns x squared

**x\*\*3**            Cube expression returns x cubed

**x\*\*(1/2)**   Square root (½) or (0.5) fractional exponent operator returns the square root of x

**x // y**           Floor division operator returns the integer part of the integer division of x by y

**x % y**          Modulo operator returns the remainder part of the integer division of x by y

## **Order of operations**

The order of operations are to be calculated from left to right in the following order:

1. **P**arentheses ( ), { }, [ ]
2. **E**xponents xe   (x\*\*e)
3. **M**ultiplication \* and **D**ivision /
4. **A**ddition + and **S**ubtraction -

You might find the **PEMDAS** mnemonic device to be helpful in remembering the order.

# Resources for more information

For more information about the concepts covered in this reading, please visit:

* [Built-in Functions](https://docs.python.org/3/library/functions.html) - Lists and summarizes Python’s built-in functions.
* [Python Keywords](https://www.w3schools.com/python/python_ref_keywords.asp) - Lists Python’s reserved keywords and a brief description of what each keyword does.
* [Different Arithmetic operators in Python](https://flexiple.com/python/arithmetic-operators-in-python/) - Provides more examples of the proper syntax for using arithmetic operators in Python.

For additional Python practice, the following links will take you to several popular online interpreters and codepads:

* [Welcome to Python](https://www.python.org/shell/)
* [Online Python Interpreter](https://www.onlinegdb.com/online_python_interpreter)
* [Create a new Repl](https://repl.it/languages/python3)
* [Online Python-3 Compiler (Interpreter)](https://www.tutorialspoint.com/execute_python3_online.php)
* [Compile Python 3 Online](https://rextester.com/l/python3_online_compiler)
* [Your Python Trinket](https://trinket.io/python3)

