# Crash Course on Python

1. Hello Python
2. Basic Python Syntax
3. Loops
4. Strings, Lists and Dictionaries
5. Object Oriented Programming (Optional)
6. Final Project

## **Week 1: Hello Python!**

### Welcome to the Course!

**Welcome to the course!**

This course is designed to teach you the foundations of programming in Python. We’re excited to join you on this journey as you learn one of the most-in-demand job skills in IT today. In the U.S. alone, according to Burning Glass data from May 2019, there were ~530K job openings in 2018 asking for Python skills.

**Course prerequisites**

This course requires no previous knowledge of programming. Familiarity with basic IT concepts, like operating systems, files and processes, networking and data management will be required in further courses. For learners with no IT background at all, we recommend taking the [IT Support Professional Certificate 101](https://www.coursera.org/specializations/google-it-support).

**How to pass the class**

You can review videos, readings, discussion forums, in-video questions, and practice quizzes in the program for free. However, to access graded assignments and be eligible to receive your official Google IT Support certificate, you must:

* Pay the [course certificate fee](https://www.coursera.support/s/article/209818963-Payments-on-Coursera?language=en_US), or apply and be approved for [Coursera Financial Aid](https://www.coursera.support/s/article/209819033-Apply-for-Financial-Aid-or-a-Scholarship?language=en_US).

AND

* Pass all graded assignments in the six courses at the minimum passing level, or above. Each graded assignment in a course is part of a cumulative grade for that course. The passing score for each course is 80%.

**How deadlines work**

When you enroll in the course, the system automatically sets a deadline for when you need to complete each section. Heads up: These deadlines are there to help you organize your time, but you can take the course at your own pace. If you "miss" a deadline, you can just reset it to a new date. There’s no time limit in which you have to finish the course, and you can earn the certificate whenever you finish.

**Getting and giving help**

Here are a few ways you can give and get help:

1. Discussion forums: You can share information and ideas with your fellow learners in the discussion forums. These are also great places to find answers to questions you may have. If you're stuck on a concept, are struggling to solve a practice exercise, or you just want more information on a subject, the discussion forums are there to help you move forward.
2. Coursera learner support: Use the [Learner Help Center](https://learner.coursera.help/hc/en-us) to find information on specific technical issues. These include error messages, difficulty submitting assignments, or problems with video playback. If you can’t find an answer in the documentation, you can also report your problem to the Coursera support team by clicking on the Contact Us! link available at the bottom of help center articles.
3. Course content issues: You can also flag problems in course materials by rating them. When you rate course materials, the instructor will see your ratings and feedback; other learners won’t. To rate course materials:

* Open the course material you want to rate. You can only rate videos, readings, and quizzes.
* If the item was interesting or helped you learn, click the thumbs-up icon.
* If the item was unhelpful or confusing, click the thumbs-down icon.

**Finding out more information**

Throughout this course, we'll be teaching you the basics of programming and automation. We'll provide a lot of information through videos and supplemental readings. But sometimes, you may also need to look things up on your own, now and throughout your career. Things change fast in IT, so it’s critical to do your own research so you stay up-to-date on what’s new. We recommend you use your favorite search engine to find additional information— it’s great practice for the real world!

On top of search results, here are some good programming resources available online:

* The official [Python tutorial](https://docs.python.org/3/tutorial/index.html). This tutorial is designed to help people teach themselves Python. While it goes in a different order than the one we're taking here, it covers a lot of the same subjects that we explore in this course. You can refer to this resource for extra information on these subjects.
* The [Think Python](https://greenteapress.com/wp/think-python/) book. This book aims to teach people how to program in Python. It's available online in PDF and browsable forms. Again, you can use this resource to learn more about some of the subjects we cover.

The [official language reference](https://docs.python.org/3/reference/index.html). This is a technical reference of all Python language components. At first, this resource might be a little too complex, but as you learn how Python works and how it’s built, this can be a useful reference to understand the details of these interactions.

### Program Surveys

During this certificate program, you will be asked to complete a few short surveys. These are part of an important research study to understand how effective the certificate is for participants like you. Make sure to check out below for a rundown of what each survey will cover.

Heads up: Your survey participation is optional but strongly encouraged. There are no correct answers, and your responses and personal data:

* will not affect your course experience, scores, or ability to receive a certificate or job in any way
* will be kept confidential, with your name separated from your data
* will not be shared outside of our research team except where you give permission to share contact information with hiring partners.

Thanks for your consideration and time! It is only by hearing from learners like you that our programs can be successful.

**Entry survey**

In the following course item, you'll have the opportunity to fill out a brief survey to help us understand why you’ve enrolled in this class. If you do not fill it out, you'll receive an invitation to fill out the survey after completing one lecture or assignment.

We will ask about your experiences leading up to this course, and what you hope to accomplish. This is critical information for making sure we can meet the needs of learners like you, and can offer this program in the future.

**Individual Course Feedback**

When you **complete the last graded assignment** **within an individual course**, you may be asked to answer a survey that revisits earlier questions and asks what you have learned up to that point in the program.

**Certificate Completion Survey**

After you **complete the last graded assignment in the final course** of the certificate, you will be asked to answer a survey that revisits earlier questions.

It also asks about what you have learned, and if you want to share your contact information with prospective employers.

### 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, and more. In contrast, automated processes complete instructions exactly as coded, in a consistent manner. They can run 24 hours a day, everyday, without tiring. For many tasks that are appropriate for automation, it can be more cost effective to use automation than human labor.

**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.

### Study Guide: Introduction to Programming

Your first practice quiz is coming up soon. This handy study guide should help you prepare for that quiz. The practice quizzes do not count towards your grade in this course. Practice quizzes are opportunities for you to check your understanding of the materials before you take the graded assessments at the end of each module.

Key Terms

* **Programming code** - Programming code is a set of written computer instructions, guided by rules, using a computer programming language. It might help to think of the computer instructions as a detailed, step-by-step recipe for performing tasks. 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. Programming syntax includes rules for the order of elements in programming instructions, as well as the use of special characters and their placements in statements. This concept is similar to the syntax rules for grammar and punctuation in human language.
* **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. Computer programs can be long, complex, and accomplish a variety of tasks. They are often developed by computer programmers and software engineers, but anyone can learn to create them. Computer programs may involve a structured development cycle. They can be written in a wide variety of programming languages, such as Python, Java, C++,  R, and more. The completed format of a program is often a single executable file.
* **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.

### 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.

### 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)