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.

How to pass the class

The Course Certificate gives you a way to prove your new programming skills to employers. To qualify for the certificate, you have to enroll in the program, pay the fee, and pass the graded assessments. If you don't want to pay, you can still audit the course for free. This lets you view all videos and submit practice quizzes as you learn. One thing to remember though, this option doesn't let you submit assessments, earn a grade, or receive the certificate.

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 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 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. 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 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. 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.

When writing code, using correct syntax is super important. Even a small typo, like a missing parentheses or an extra comma, can cause a syntax error and the code won't execute at all. Yikes. If your code results in an error or an exception, pay close attention to syntax and watch out for minor mistakes.

If your syntax is correct, but the script has unexpected behavior or output, this may be due to a semantic problem. Remember that syntax is the rules of how code is constructed, while semantics are the overall effect the code has. It is possible to have syntactically correct code that runs successfully, but doesn't do what we want it to do.

When working with the code blocks in exercises for this course, be mindful of syntax errors, along with the overall result of your code. Just because you fixed a syntax 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 submit it to have your work checked.

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.

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.
- Search for answers or ask a question on Stack Overflow.
- Subscribe to the Python tutor mailing list, where you can ask questions and collaborate with other Python learners.
- Subscribe to the Python-announce 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 Wikipedia page or in the section on the history of the software 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 (Stack Overflow)
- Why is Python Growing So Quickly - Future Trends (Netguru)
- By the numbers: Python community trends in 2017/2018 (Opensource.com)
- Developer Survey Results 2018 (Stack Overflow)

First Programming Concepts Cheat Sheet

Functions and Keywords

Functions and keywords are the building blocks of a language's syntax.

Functions are pieces of code that perform a unit of work. In the examples we've seen so far, we've only encountered the print() function, which prints a message to the screen. We'll learn about a lot of other functions in later lessons but, if you're too curious to wait until then, you can discover all the functions available here.

Keywords are reserved words that are used to construct instructions. We briefly encountered for and in in our first Python example, and we'll use a bunch of other keywords as we go through the course. For reference, these are all the reserved keywords:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

You don't need to learn this list; we'll dive into each keyword as we encounter them. In the meantime, you can see examples of keyword usage here.

Arithmetic operators

Python can operate with numbers using the usual mathematical operators, and some special operators, too. These are all of them (we'll explore the last two in later videos).

- **a + b** = Adds a and b
- **a - b** = Subtracts b from a
- **a * b** = Multiplies a and b
- **a / b** = Divides a by b

- **a ** b** = Elevates a to the power of b. For non integer values of b, this becomes a root (i.e. a**(1/2) is the square root of a)
- **a // b** = The integer part of the integer division of a by b
- **a % b** = The remainder part of the integer division of a by b