PYTHON

PROGRAMMING FOR BEGINNERS

A Must Read Introduction To Python Programming

ROBERT RICHARDS

Python Programming For Beginners

A Must Read Introduction to Python Programming

Robert Richards

Copyright and Disclaimer

We support copyright of all intellectual property. Copyright protection continues to spark the seed of creativity in content producers, ensures that everyone has their voice heard through the power of words and the captivity of a story. Uniqueness of culture and content has been passed down through generations of writing and is the DNA of every intelligent species on our planet.

This publication is intended to provide helpful and informative material. It is not intended to diagnose, treat, cure, or prevent any health problem or condition, nor is intended to replace the advice of a physician. No action should be taken solely on the contents of this book. Always consult your physician or qualified health-care professional on any matters regarding your health and before adopting any suggestions in this book or drawing inferences from it.

The author and publisher specifically disclaim all responsibility for any liability, loss or risk, personal or otherwise, which is incurred as a consequence, directly or indirectly, from the use or application of any contents of this book.

Any and all product names referenced within this book are the trademark of their respective owners. None of these owners have sponsored, authorized, endorsed, or approved this book.

Always read all information provided by the manufacturers' product labels before using their products. The author and publisher are not responsible for claims made by manufacturers.

DIGITAL EDITION

Any and all product names referenced within this book are the trademark of their respective owners. None of these owners have sponsored, authorized, endorsed, or approved this book.

What You Will Learn In This Book

How This Book Will Help You and Why

In the current technology driven world, finding professional success has essentially become tantamount to staying one step ahead in the field of technology. No matter what field or profession you may find yourself in, there is simply no way to find large scale success without a certain command of technological aspects, whether in marketing, promotion, or distribution. The best way to get ahead, and stay ahead, when it comes to technology in business is to build the technological aspects of your business from the ground up. Doing this, of course, requires a certain knowledge of how to manage the building blocks of technology: namely a working knowledge of code. This includes Python Programming. This book covers Python Programming essentials for beginners that will help the person understand more about this type of coding.

Dive Right into the Book! Or Learn a Bit More About the Author

About The Author

Robert is a teacher and Python instructor, he started learning about Python since 1999, and started writing Python code and teaching students in live classes a year later. With over a decade of teaching students Python Robert has become a source of knowledge on the subject and recognized that where 90% of the students struggled was at the beginning, just wrapping their head around the power of Python.

He created this book to help those students and to make their tasks easier in understanding the basic concepts in Python. This is a must read book for all beginners to comprehend the terminology and standards associated with Python programming.

Contents

ACKNOWLEDGMENTS

Words To The Wise

"One machine can do the work of fifty ordinary men. No machine can do the work of one extraordinary man."

- Elbert Hubbard

What Is Python Programming

A Code to Crack All

In the current technology driven world, finding professional success has essentially become tantamount to staying one step ahead in the field of technology. No matter what field or profession you may find yourself in, there is simply no way to find large scale success without a certain command of technological aspects, whether in marketing, promotion, or distribution. The best way to get ahead, and stay ahead, when it comes to technology in business is to build the technological aspects of your business from the ground up.

Doing this, of course, requires a certain knowledge of how to manage the building blocks of technology: namely a working knowledge of code. To the beginner just becoming introduced to it, code can look like an overwhelming challenge. Without intimate knowledge of the workings of it for the novice finding the right system to suit your purposes can seem an almost insurmountable challenge.

Enter Python, a code build upon the simplicity of use and small learning curve for first time users. As far as novices to the code world go, Python would seem to be the perfect option for managing technological business aspects regardless of the type of field you're in. Python is a code that runs on multiple software systems. It brings already programmed in variety of applications useful in managing business operations.

Python is billed as one of the easiest and simplest codes to learn and operate on the market, and it backs up this claim with a community of programmers willing to engage in dialogue, workshops and conferences all in order to help each other become more proficient. A relative newcomer to the programming world, Python 3.4 was released in

February, and as such will only continue to improve upon a reputation already based on how easy it is to use.

There are already numerous well documented success stories of Python helping various businesses, from all different types of industries. From biology to film, Python runs applications and services perfectly tailored to fit the needs of professionals of all types and levels. In addition to the community of programmers, on its website Python offers a plethora of resources for troubleshooting and figuring out how to use python to its maximum effectiveness. This easy access to help, more than anything, is what makes Python the perfect code for those who need to get ahead technologically without being computer engineers, or hiring one.

History of Python Programming

How It All Started

Python was first created in the 1800's and officially went into effect in December of 1989. Python was launched by Guido Van Rossum at CWI. The name Python was inspired by, and derived from, Brit-com Monty Python's Flying Circus. Today, Python continues to be developed by a large organization of volunteers.

In January 1994, Python's version 1.0 featured new programming tools such as reduce, filter, map and lambda. It was announced by Van Rossum that these new programming tools were gained thanks to Lisp Hacker.

Before Van Rossum left CWI, the last version made was Python version 1.2. Rossum then released more versions after continuing his work at CNRI. Python had acquired new features such as data hiding, complex numbers support and keyword arguments.

After the first version of Python, Python 1.6 and 2.0 were released closely together. Both the 1.6 and 2.0 Version of Python were released in 2000. Python 1.6 was released in September of 2000 and Python 2.0 was released in October. Version 2.0 came with many big new updates. Some of these major updates included Unicode support, list comprehensions and a garbage collector. In 2002, the development group also moved to BeOpen.com and formed the PythonLabs organization.

The version Python 2.0 was the only version released by BeOpen.com. Then, Van Rossum

and the rest of the developing crew moved on to work with Digital Creations.

Python version 2.1, introduced in 2001, was relatively similar to both 1.6 and 2.0 versions. This version offered more advance language specifications. All specifications and codes to this version is owned by PSF. This version's license was then named Python Software Foundation License. Python version 2.2 was released in December of 2001 and featured new generators. Version 2.7 was the last version released of the Python version 2 series.

Python gradually advanced through more updated versions and reached version 3.0 in December of 2008, bringing many more new advanced features to the table since version 2.0. Python 3.0 was simplified by taking duplicate features of prior versions and creating only one convenient feature for each that would accomplish the same task. This version is also sometimes called PY3K or Python 3000 and Backward Compatibility was removed in this version. Python continued to improve features and language development throughout all of the version 3 series, which included versions 3.1-3.4. The latest Python version is 3.4, which was recently released in March of 2014

Main Python Features

What You Get With a Python

One of the most notable things about Python is its elegant syntax which makes all programs easy to read. The language is extremely easy to use and is perfect for ad-hoc programming uses and prototype development. It is easily comparable to Java, Schema and Perl, being a powerful and clear OOPL. It also has an interactive mode which makes short code testing easier. This free software can be downloaded without any cost and it can also be redistributed or modified for free.

Variety of Data Types, Strings, Dictionaries and Lists – Python supports 4 basic number types (int, long, float and complex). Both Unicode and ASCII strings are available. Python's basic data structure is 'sequence' and 6 sequences are available. Tuples and lists are the most common sequences. Lists are versatile and can be written as values separated with commas and enclosed in brackets. The Python dictionary container can store any number of objects and this includes container types. They are also called hash tables.

Object Oriented Programming – Python offers both OOP and POP. In languages that are procedure oriented, the entire program is built around reusable programs that are functions or procedures. In languages that are object oriented, the objects are what the program is built around, combining functionality and data. As compared to Java and C++, Python is

extremely simple when it comes to OOP. Also, being a high level language, Python does allow the programmer to waste time with low level issues like memory management.

Interpreted – Basically, programs that are written in C++, C or compiled languages are converted into the computer's language (binary code) from the source language (C++,C) with the help of a compiler. The loader/linker software copies programs to the memory from the hard disk and run them. In Python, however, compilation is not necessary and program can be run directly. The source code is converted to bytecodes internally by Python and then run it after translating it to binary. This ensures portability in Python programs.

Libraries – The standard library of Python is pretty huge, letting programmers use features like expressions, unit testing, document generation, databases, threading, ftp, CGI, WAV, HTML, GUI, web browsers, XML-RPC, and XML.

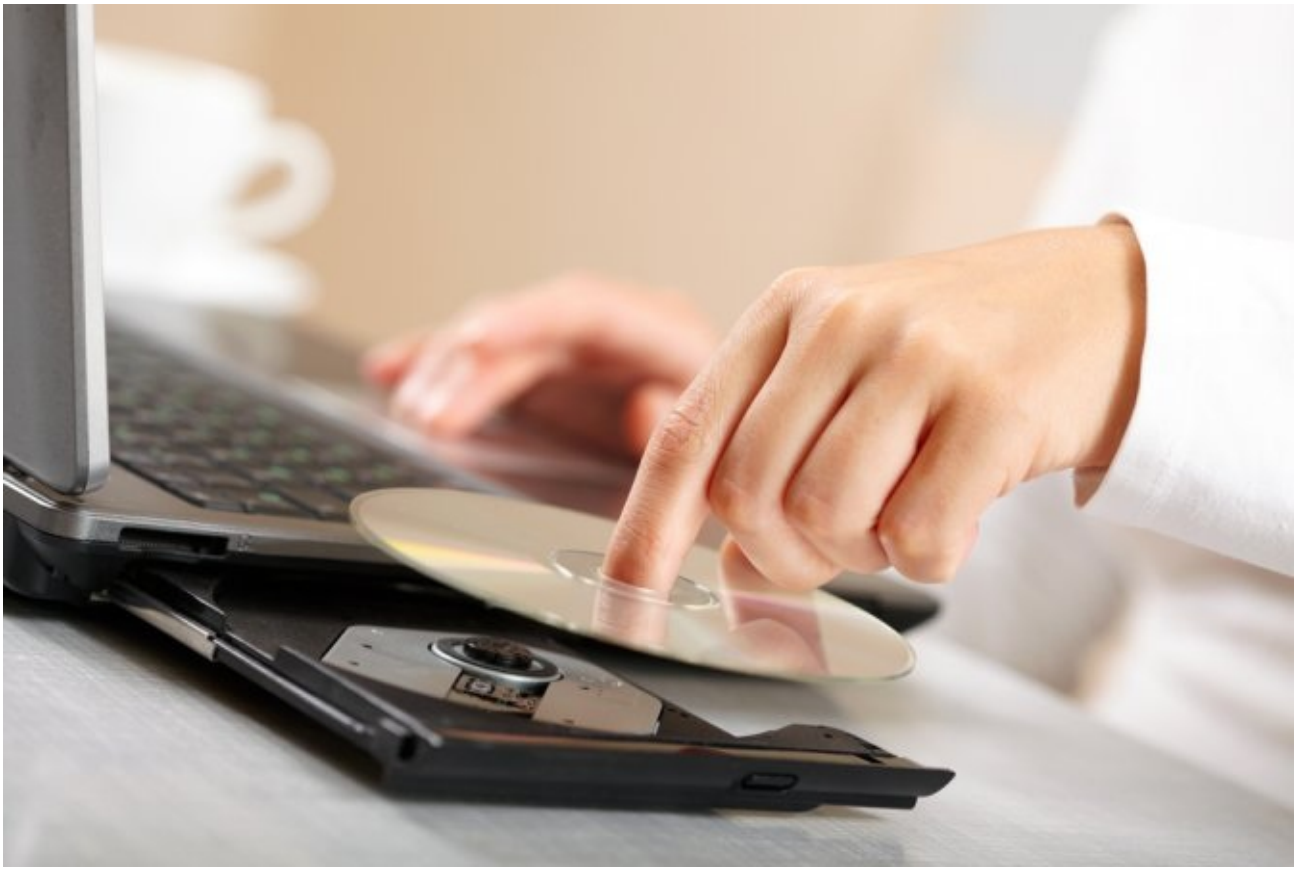
By using C++ or C, new modules can be easily added and Python can be extended. It is also easily embeddable in an application, running on various OS and computers. Python is simple, easy to learn and portable.

How to install Python

On Mac, Linux & Windows

Installing python on windows

Go to the main Python download site, www.python.org. On the left side of the window, you will see a title “Quick Links”. Click on Windows installer to start downloading the program. Python offers two major versions 2.x and 3.x Sphinx 1.3 can run under different python versions including 2.6, 2.7, 3.2 and 3.3. The recommended Python version is 2.7. Follow the step by step provided by the installer to complete the installation.



After installing python, it is important to add python executable directories to the environment variable path. This will make it possible to run python and package programs from the command prompt.

To add the directories, you only need to right click on “My Computer icon” and then choose properties. Look for the advanced tab and on Environment Variables. If you find that the PATH is already part of the “System Variables”, edit it. If PATH is not part of the “variables”, you just need to save it as PATH variables. Your Command Prompt is now ready to use. To check whether the installation was successfully, type the work Python and then Enter. If “type ctrl+z and Enter to quit” appears, then your installation was successful.

Install python in Linux

If you are using Linux, there are several packages that you need for you to download Python. These packages include Fedora, Gentoo, Ubuntu and Debian. With these packages, you can simply follow the steps there and install your program. Users of Gentoo may get messages telling them to change USE or keyword to be able to download Python. If you get this message, all you need to do is to run this command together with —autounmask-write, then run sudo dispatch-conf to configure the changes. This should solve the problem.

Install Python in Mac

Just like in Linux, if you are using Mac you need to install some packages before downloading and installing the Python. The only unfortunate thing here is that Mac doesn’t have several packages like Linux. But you can install Macports. After installing this package, you only need to visit the official Python website and start downloading the program. Follow the steps provided there to fully install and start running the program on your computer.

Python Terminology And Guide

What You Need To Know

These are all the most important terms and things you need to know in Python.

Basic Python Syntax

Python is a prototypical offshoot of Basic programming that is useful even for the most advance PC and Mac systems in the modern era. In many ways it sidesteps some of the requirements of traditional Basic language, but retains familiar protocols.

The crux of Python language is command and line structures which are based around specific program objects. Paired syntax characters equate to known values and are expressed on single or multiple lines. The objects in focus are directed by Python syntax to variable lists and consequent related prior functions. Python language streamlines the commands needed to loop, compare, print, list, comparatively slice, and create sets of objects.

Most sets and strings of objects in Python Basic language ordering are organized using specific bracketing principles that are easily learned from a foundational Basic language knowledge.

Python syntax derives its name from the notion of “stringing together” commands in Basic language that would normally require multiple, and somewhat redundant, lines. With a Python style of language, multiple object directions can be housed together using various bracketing techniques.

Python syntax is akin to a Morse code sender broadcasting a coded message without breaks for responses. Instead of single words and key points, the sender can code detailed paragraphs about a certain object or situation with the full confidence that the receiver possesses the ability to decode messages with a modicum of grammar and referential

knowledge. Python syntax assumes the capability of a computational core to acknowledge bundled commands to execute on a certain object. Python syntax saves time, keystrokes, and utilizes system efficiency.

Anyone having experience with practical application of Basic computer coding language will likely assimilate Python syntax with ease. Python is a coding technique that simplifies obtuse coding lines into object-specific strings of related commands that a system can understand using base programming parameters. Python is the skill of communicating related commands with single lines using specific symbols and brackets to bypass time-consuming code-writing norms. Python syntax produces the same results as standard Basic language, but it is accomplished in much less time and fewer code lines.

Python Variable Types

The first, and probably the most common variable type, is how you assign values. Let's say for example you want your name to be directly associated with your age. To do so, we would program the variable type like this.

John = 32

This means that every time we use the term 'John' we get an output of 32. So, in theory, if we have a piece of code that is written like this

print John + 2

means we get an output of 34. This is because the computer is taking John (32) and adding the number 2 to that and giving us 34. You might notice we use 'print' in the code above. Print is telling the computer to perform this piece of code. If you're doing a math equation, getting names, etc. we want to use print as the means to tell the computer what to do.

Another piece of code in variable types that are helpful are # and strings. The # acts as a note to whoever reads this code. The computer will ignore anything after the # and programmers often use this to make notes.

The string function assigns a word value to a variable instead of a number. We implement this as follows.

name = "Sarah"

anything in the "" quotations gets saved as a string.

Python - Basic Operators

Pythjon is really like an algebra-like programming language that allows you to talk to your computer through command prompts. The information you want the computer to display should be enclosed in quotation marks. In python, “/n” is used for “new line”. The quotation marks are not needed for mathematical operations. Two or more items or commands can be printed on the same line if a comma separates them. The command “print” is also used to tell the computer to display the answer of the mathematical operation. When you finish using Python just type exit.

When writing programs or script, you must save them in a file. You can have a folder for all your Python files. If you want to run a saved program, you can find your program either through a file window or a command window. Once you have saved programs, “file.py” in you Python folder, you can edit them by opening them with Notepad. Do not save your file.py files as file.text or they won’t execute. You can also make a shortcut on your desktop for quick use.

The Python language does not have semicolons or braces. Blocks have the indention. The “#” identifies comments for users to read. Variables are used to store a value that can be changed. You assign a value by typing “v=1” (or whatever number you want to input). Variables that contain text are called strings.

In Python, identifiers are names used in association with variable, functions, and other objects such as class and modules. Identifiers are expressed with an upper or lower case letter or an underscore first, than a zero or more letters, digits 0 through 9, and underscores. Class names are capitalized and other identifiers are lower case. Using an underscore to start an identifier means the identifier is private. Two underscores starting an identifier mean strongly private. Ending in two underscores means the identifier is a special name.

In Python a group of statements that make up a code block are called suites. The following reserve words are compound or complex statements. A header line begins a statement and a colon ends the statement. A suite is the line or lines that follow the statement.

Reserve Words (cannot be used as constants or variables).

and assert break

class continue def

elif else except

exec finally

for from global

if import in

is lambda not

or pass print

raise return try

while with yield

Python - Decision Making

Decision making is required when a programmer wants to execute a code if a certain condition is satisfied. The programmer is required to specify the different conditions that will be tested by the program. They will also determine which a set of statements that will be worked on if the condition specified turns out to be true or false. In most programming languages, you will find a structure that assumes zero or null values as false and non-zero or non-null as true values. This python programming language three main types of decision making statements; if statements, if…else/elif statements and nested if statements.

If statements

This python statement is similar to other grammatical statements. The statement contains a logical expression which is used to compare data. A decision is normally made based on the results of that comparison. If the expression is determined to be true, then the statement or statements inside the If statement will be worked on. However, if the boolean expression is determined to be false, only the first code after the end of the If statement will be executed. An if statement can also occur as a single statement suite. These are statement(s) with only one line. The single suite statement can have the same format as the header statement. This means that after the code is executed, the programmer will get the same result.

If…else statements

The if…else statement is an optional statement. The statement contains the set of code that is used to execute the conditional expression if the if statement turns out to be 0 or false value. There can only be one if…else statement following if. On the other hand, the elif statement allows the programmer to check multiple expressions. This means that you can execute a set of code when one of the conditions turns out to be true. The elif statement is optional just like the else statement. The main difference between these two statements is that while the if…else statement allows the programmer to have only one statement after if, there can be several elif statements following if.

Nested if statements

Sometimes a programmer may want to check for other conditions after one of the conditions resolves to be true. Such situations call for nested if statements. The programmer can have an if…else…elif statement inside another if..else…elif statement. Indentation is the only way a programmer can figure out the level of nesting. This statement can be confusing and it is advisable to try and avoid it if possible.

Basics of Python Loops

Each loop in Python is a command that needs to follow a basic format if it is to be done correctly. The loops have requirements that will keep them separated from other parts of the code. For example, each loop will feature a description and a conditional statement. If conditions of the statement are not meant, then the command is not processed until it finds the proper requirements set forth by the command. This basic architecture is what makes Python loops unique and a great way to code thanks to their individuality. Plus, Python requires that each statement is tested to determine its functionality. Even if the conditions are not met, one can be sure that the line was thoroughly researched by the coding language.

Python language also makes multiple lines of the same statement or task simplified by abbreviating the code. There is a set of control that can be met with the statements as well. The control code will keep execution in check so that it never deviates from its intended purposed. If a line was to differentiate from its coding, then it would destroy the rest of the sequence. Thus, it is essential that a control loop is established to maintain the quality and property values of the code itself.

PYTHON-NUMBERS

Understanding Python Numbers is a part of basic Python programming. Numerical values are stored in number data type.

Assigning Value and Del Statement - Being immutable data, when any number's value is changed, a new allocated object is the result. These objects are created when a value is assigned to them.

For instance –

var1 = 1

var2 = 10

And so on.

The 'del' statement can be used for deleting a number reference. Its syntax is –

del var1[,var2[,var3[,var4[,var5[….,varN]]]]]

Number Types - Four types of numbers are supported by Python –

Signed Integers or 'int' – These are negative or positive whole numbers, often referred to as 'ints' or 'integers', without decimal points.

Long Integers or 'long' – These are unlimited size integers and are written just like an integer except 'L' in lower or upper case follow them. This was dropped in Python 3.0. Instead, 'int' is used.

Floating Point Real Value or 'float' – Floats are representatives of real numbers. They can be e or E, i.e. scientific notation, and are written with the fractional and integer parts being divided with a decimal point.

Complex Numbers or 'complex' – These are in the 'a + bJ' form with a, b being floats and J being the √-1; 'b' forms the imaginary part and 'a' is the number's real part. However, Python doesn't use complex numbers a lot.

Up Cast – The up cast of the number types in automatic and usually follow the order –

Int

Long

Float

Complex

The farther towards the right we go, the more the precedence is.

Conversion To Number Type – Numbers are converted internally in Python but there are times when a number has to be coerced to another type for satisfying operator requirements or function parameters.

int(x) – Converting number x to plain integer

long(x) – Converting number x to long

float (x) - Converting number x to float

complex(x,y) - Converting number x and y to complex numbers where x is the real part and y is the imaginary part.

Mathematical Functions – Common mathematical functions on Python include the following –

abs(x) – The distance (positive) between 0 and x; x's absolute value

fabs(x) – x's absolute value

ceil(x) – x's ceiling

exp(x) - ex; x's exponential value

log(x) – For x>0, natural log of x

log10(x) - For x>0, base log of x

pow(x,y) – x**y

sqrt(x) – x's square root

Python – Strings

In order to understand what a string is, Characters must be explained. Characters are anything you can type on your keyboard; letters, numbers, and other characters e.g.: ‘i’,‘2’,’@’,’ and are denoted by enclosing the character in single quotation marks.

String, essentially, are a sequence/list/array of characters and are denoted by being enclosed in double quotation marks. So, for example, the string “apple” consists of the characters ‘a’ + ‘p’ + ‘p’ + ‘l’ + ‘e’.

At this point you might wonder why such a strange paradigm was chosen for Python (and other programming languages) to deal with textual information. Why not just have a primitive type called text for all forms and shapes of text?

To answer this question we have to consider what storing text on a computer actually requires. Fundamentally, any information processed on a computer consists of a sequence of bit states. Specifically speaking, each specific piece of information on your computer has a physical counterpart representing that information with a unique state.

So, essentially, all words that run through a computer are equal to a unique number which represent those words. More specifically, each character has a unique number representing that character, and so, String, are essentially very large numbers. Knowing this, it may be clearer why Python modulates text information first into characters which can be combined into Strings, so we do not have to store two versions of the same text in different memory locations.

Manipulating Strings includes algorithms like concatenating different String together, flipping Strings, extracting characters of Strings and many more!

String manipulation is a very essential programming skill and is therefore often used as an introductory topic for beginners. Make sure that you are familiar with Strings, their uses and how to manipulate them to your needs.

Python Lists

Lists are one of the most widely used data structures in Python. Lists, unlike tuples, are dynamic and can expand and contract. Think of a list as a named container that can expand and contract to contain other objects. Lists can contain tuples and strings.

Syntax and Structure

Lists are ordered objects contained in square brackets. For example:

list_1 = [“A”, “b”, “C”, “White House”, “Grover”]

If you type “List1” at the Python interactive parser, it will return

[‘A’, ‘b’, ‘C’, ‘White House’, ‘Grover’]

Pulling Data from Lists

The order of items in lists in Python is important. You can use name of the list and the number of the item in the list to return just the value of that specific item. For example:

list_1[3]

‘White House’

Why did “list_1[3]” return ‘White House’ and not ‘C’? Because lists (and tuples) use zero-based indexing - the first item in the list numbered “0” when returned.

You can also use negative indexing for a list. Python will always grab the last item in the list on an index of -1, the second-to-last on -2, the third-to-last on -3 and so on.

You can pull multiple values out of a list with the “slice” operator, which can take two numbers separated by a colon “:”. If you don’t use a leading number, the slice starts from the front of the list and goes for as many items as the number after the colon.

For example

list_1[:3]

[‘A’, ‘b’, ‘C’]

Notice that it only returned the first three items?

If you wanted to get the last three items of the list and you knew its length, you could do the following:

list_1[2:4]

[“C”, “White House”, “Grover”]

If you don’t know the length of the list, but want to start from the last item and work your way to the front of the list for a certain number of positions, you can omit the second number before the slice operator:

list_1[3:]

[‘White House’, ‘Grover’]

Once again, zero-based indexing means that you’re going to end up getting one value fewer than the digit you typed in. If you use [:n] and [n:], you will always slice out the entire list, but at an arbitrary point within it. This is a handy trick to have when parsing database information.

An added trick - all of these operators for lists, and grabbing data from them also work with tuples.

Python - Tuples

A tuple in Python is a comma delimited data structure that's immutable, and is one of the "Big Three" data structures, the other two being lists and strings. Tuples, lists and strings are all examples of sequence data types in Python.

Samples and Syntax

The following are examples of tuples:

tuple 1 = ('lemonade', 'ginger ale', 1848, 1969);

tuple 2 = (1, 2, 3, 4, 5);

tuple 3 = "a", "b", "c", "d";

Tuples are typically contained by parentheses "()", but as the third example above shows, they can be treated as a series of text strings separated by commas. When a tuple contains a single object, there must be a comma after the item.

Overview of the Data Type

Tuples can be thought of as fixed value lists. All of Python's standard indexing techniques work with tuples, and it's possible to name a tuple, and use a function to call the nth item from the tuple. Where tuples differ from lists (and dictionaries) is that lists are assumed to be mutable, meaning items within the list can be changed, and the order of items within the list are also changeable. Tuples do not change, though data pulled from tuples can be sorted into other tuples.

Programmatically, tuples are used to store values that remain constant through the part of the program you're using. Where lists and strings can be thought of as fungible data storage tools within the program, use tuples when you need a bit of permanence. Because of the presumption of immutability for tuples, arecomputationally less intensive on indexing operations. In the background, Python uses tuples for most sequential data that isn't specified as a list-type data structure.

When you're using Python in the workplace, tuples are most commonly used for things like coordinate pairs, or regularizing data (like first name, last name, date of hire) that gets pulled from or written into a database or other data store. Tuples can be included as elements of a list (make sure you watch your parentheses for nesting purposes) and vice versa.

Conversion of Data Types

Python has a lot of functions — notably the "list" and "tuple" functions to convert data from one format to another. The standard slicing and concatenation commands work on both data types. Python includes tuples and ways to manipulate them, to maintain syntactic consistency and clarity.

Dictionaries are effectively a tuple made up of multiple lists.

Python - Dictionary

Python dictionary is a mutable key-value pairing feature. It is mutable because it can be changed. It pairs a key to a value, in each pairing. The keys in the Python dictionary are immutable, like numbers or strings and cannot be changed. They also have to be unique. The dictionary in Python can contain unordered items. In other programming languages this is referred to as a hash table or an associative array.

Creating a dictionary in Python can be done in more than one way. One method, is using literal codification, where the key-values are contained in curly brackets and the pairs are separated by commas.

months = { “Jan”: “January”, “Feb”: “February” }

The first value, “Jan” is the key, followed by a colon and then the value, in this instance the value is “January.”

Python dictionaries can also be created using dict (), the dictionary function.

vals = dict (uno=1, dos=2)

Another method of creating a dictionary in Python is by first creating an empty dictionary, followed by one or more instances. In this case, 3 pairings are added.

clothing { }

clothing[“hat”] = “bonnet”

clothing[“shoes”] = “hush puppies”

clothing[“jacket”] = “leather bomber”

The keys are assigned first, inside the square brackets. The values are on the right side of the equals sign. The dictionary is put together using a comprehension, which is defined with 2 parts. The first part, as shown, is known as the object. The second part is referred to as the, for i in range(4) loop. 4 pairs have now been created in the example above.

dic = { i: object() for i in range(4) }

Operations can be performed using dictionaries. Values can be added or removed from one or more dictionary using specific functions. The pop function will remove pairs from the dictionary. It is also possible to create new dictionaries from lists using the fromkeys() function. Using the setfault() function, if a key is available then a value is created. Dictionaries can be joined using the update() function.

The clear() function can remove all items from a dictionary.

Python - Date & Time

When you are planning to create date and time stamps, you need to remember that there are many modules for creating them. When you use these modules correctly, you can put the stamps anywhere you want. Also, you will be able to create time and date stamps that people can use themselves. Consider each way of using the module in your programming work.

A Clock

You can create a clock that fits on the screen easily. These clocks can carry the date and time, and you can populate them with a calendar that people can check for many years to come. Also, these clocks can be designed in a number of styles. Some of them look better in certain colors, and you can change the colors to create a special clock that is bespoke for your current project.

Stamps

You can create time and date stamps that will go on a number of applications. They can be used to mark word processing documents, and they can be used to mark messages in a messenger service. Also, these stamps can be used when people are editing documents.

Timers

When you want to create a timer, you will be able to offer people a stop watch and timer. These timers can count down from any number you want, or you can create a stop watch that counts down to the hundredth of a second. People use these timers to work, and they can use stop watches to measure processes that they are working on.

The timers and stop watches are wonderful for people who are timing athletic events, and they can be used for people who work on their computers. Also, these people need to make sure that they are setting up the clocks to work for athletics or work purposes.

The Clock Setup

The time and date module allows people to set up the clock to run on 12 hour or 24 hour timing. These times are easier to read for many people, and they allow people to read a clock that they are used to reading. This will help with European users, and it will help with people who prefer to read modern American military time.

The Python time and date module is going to help people create clocks and calendars that will work well on all the projects they work on. These modules can provide a time and date for each item on the computer.

Python – Functions

Python has many pre-made functions that a programmer can use. However, programmers can also define their own functions. Such functions are known as user-defined functions. To start using a function, you only need to call the function. Calling a function means giving a function input. The function will respond by giving you an output. Those values that you give the function to tell it what it should do are called parameters.

For example if a function multiplies any number by 2, the stuff in parameter will tell the function the number it should multiply by 2. Let's assume you have put 30 in the parameters. The function will in turn do 30x2. One of the most important characteristic of parameters is that they have a positional behavior. Thus, you should always inform them in the same sequence that they were defined.

There are a few rules that you should follow when defining a function. First, you need to note that a function set always starts with keyword def which is then followed by the name of function and parenthesis. The programmer should always place all input parameters within these parenthesis. The parenthesis can also be used for defining parameters. The other function rule that you should note is that the code block in every function should start with a colon. You should also note that the first statement of a function can be an optional statement.

A programmer can call a function using four different types of formal arguments. These arguments include variable-length arguments, keyword arguments, default arguments and required arguments. A required argument is an argument that is normally passed to the function in a correct positional order. Here, the programmer should make sure that all the arguments in the function call perfect match with the function definition.

Default arguments assume a default value if the programmer fails to provide a function call for that argument. In variable length argument, the programmer is allowed to process more arguments that the ones specified when defining the function. Unlike default and required arguments that should be named while a function is defined, variable-length arguments do not need to be defined. The programmer should place an asterisk (*) before the variable name that will hold these arguments. The last type of argument is the keyword argument. This is an argument that is related to the function call. When used in a function call, the caller easily identifies that argument by the parameter name. This means that the programmer can place arguments out-of-order or skip them altogether.

Python - Modules

Python Modules carry information for different parts of the programming language known as Python. You can use these modules to learn about the coding from many different points of view. You will not be able to use the programming language until you have learned all the different aspects of the programming language.

The Basics

When you get into the basic modules, you will be able to learn quickly how to set up all your programming code. When you do not know the basic formatting for the code, you will not be able to create complicated versions of the code. Your coding begins and ends with your understanding of the basics of the Python language.

The Combinations

You can learn through the modules how to put together combinations of the code. These combinations will help you to create unique bits of programming. People who have progressed past the basics need to be sure that they understand how to combine different parts of the code to make a better product.

Graphics

When people create graphics with the Python language, they need to have training in the graphics portion of the language. Graphics are the most important part of a computer program, and they need to be handled properly. When you are planning to create a computer game or user interface, you need to make sure you read all the modules for graphics.

Security

You can create security protocols with Python if you have read up on the coding. You can code security into all your programs, and you can create a user interface that allows people to sign in to their accounts. If you have learned the security features of Python, you will be able to protect your programs and the people who use them.

Advanced Coding

There are advanced versions of the code that people have created over the years. These advanced versions of the code are unique, and they are often designed to do just one thing. You can learn the simple things that you need to know to make your program even more viable. Also, you will be able to manage your coding in such a way that it is much more efficient.

Learning how to handle the Python coding language will help you to create graphics, security and advanced coding that will help you and your customers. You can create programs that look professional when you use the coding language correctly.

Python File I/O Functions

Python file I/O (input/output) methods describe code instructions responsible for managing basic file and directory manipulation. For example, when a file is created all of the data and variables are stored in RAM. When a computer is powered off all of the data that is stored in RAM is immediately erased. This is one of the areas where Python file I/O operations are helpful. It can use programmatic methods to save data in a file and save the file to the hard drive.

Let's review the basic Python options that enable the execution of file input/output instructions.

The print screen function is an often used operation that allows data to be displayed on the user's screen. This is accomplished by using the print statement followed by the text or variable intended for display (i.e. print "this text").

When it is necessary to read information entered from a keyboard, Python has two available functions; raw_input and input. The former prompts the user to enter a string and displays(on the screen) the exact string that was entered by the user. The latter assumes the data entered is a Python expression (e.g. math equation) and calculates the data to produce a result.

Python, input/output functions also includes opening and closing files. To open files, the open() method can be called with a variety of parameters that allow files to be opened as read or write only (other options exist). To close a file, the close() method is executed.

Python has built-in options to read or write data to files. This is accomplished by using the read() and write() methods respectively. These methods are capable of handling text and binary string formats.

Python, input/output operations can also identify and display the current position of the pointer within a file. This is accomplished by using the tell() method.

Python, has a powerful mechanism to execute existing operating system functions for Windows, Mac and Linux using the Python os module. This module is capable of interfacing with operating system functions to rename and delete files. The methods that interface with the os module (for these tasks) are rename() and remove(), respectively.

The os module, also has the ability to interface with directory management operations. These methods are mkdir(), chdir(), getcwd() and rmdir(). These methods are responsible for making a directory, changing the current directory, displaying the current directory and deleting a directory, respectively.

Python – Exceptions

Python provides two basic features of handling unexpected error in your python programs; exception handling and assertions. An exception is an error that occurs during the running of a program. These errors disrupt the normal running of a program's instructions. When a python program encounters an exception, it must handle the error immediately. However, most programs cannot handle these errors and the result is usually an error message. This message shows what happened, the type of exception and details of the exception. The last line of the error message shows what happened. There are different types of exceptions that a program can encounter including NameError,TypeError and ZeroDivisionError.

How to handle exceptions

You can write programs that handle selected exceptions. These programs normally ask the user for input until a valid integer has been entered. The programs may also allow the user to interrupt it using control-C or any other feature provided by the operating system. This user-generated interruption is indicated by raising the KeyboardInterrupt exception.

For a try statement to work, the following steps should be followed. The first step involves execution of the try statement. If an exception doesn't occur after that step, then the except clause is skipped.

This marks the end of the execution of the try statement. The rest of the clause is skipped if an exception clause is detected during execution stage. The except clause is executed if the exception named after the except keyword matches with the type of exception detected. If the exception that occurred does not match with the exception identified by the except clause, it is passed to the outer try statements. If there is no handler in the try statements, the exception becomes an unhandled exception and the execution stops there with a message.

A try statement can have more than one clause. This specifies the handlers for the different exceptions. Handlers can only handle exceptions occurring in the corresponding try clause and not in other handlers from the same try statement. The except clause can name more than one exceptions. These exceptions are normally shown in the error message.

The try except statement can have an extra clause; else clause. This clause is very important for clauses that should be executed if the try clause does not raise an exception. The clause avoids detecting any exception that was not raised by the try except statement protected code. This means that there is no need of adding an additional code to the try clause.

Acknowledgments

We support copyright of all intellectual property. Copyright protection continues to spark the seed of creativity in content producers, ensures that everyone has their voice heard through the power of words and the captivity of a story. Uniqueness of culture and content has been passed down through generations of storytelling and is the DNA of every intelligent species on our planet.