



Python Basics

Python Fundamentals

Name of presenter

Date

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Welcome to Python Basics.

What you will learn

At the core of the lesson

You will learn how to:

- Install Python on a Linux computer
- Define basic Python terminology
- Declare variables in a Python script and perform operations on those variables
- Explain statements, functions, and exceptions



In this module, you will learn how to:

- Install Python on a Linux computer
- Define basic Python terminology
- Declare variables in a Python script and perform operations on those variables
- Explain statements, functions, and exceptions



System requirements for Python

**Microsoft
Windows**

macOS

Linux

Demo: Install Python



Check the current installation of Python and install a newer version of Python.

1. By default, CentOS includes Python V2.7.
2. You can check which version you have by using the following command:
`# python --version`
3. You can observe that Python V2.7 is running. (Note: For the purpose of this course, you will use Python V3.7. You will learn how to update Python versions.)
4. Run the following command:
`# yum install gcc openssl-devel bzip2-devel libffi-devel`
(This command installs a series of packages. At the end, the installer will ask if it is ok. Respond by entering: `y`)
5. Run the following commands:
`# cd /usr/src`
`# wget https://www.python.org/ftp/python/3.7.2/Python3.7.2.tgz`
(The second command tells CentOS to *get* the specific file from ***python.org***.)
6. Run the following command:
`# tar xzf Python-3.7.2.tgz`
(This command tells CentOS to extract the file.)
7. Run the following command:
`# cd Python-3.7.2 # ./configure --enable-optimizations # make altinstall`
These commands take some time to run.
8. Now, remove the downloaded file from the system:

```
# rm /usr/src/Python-3.7.2.tgz
```

9. You will be asked if you want to remove this file. Enter **y** and press **ENTER**.

10. Verify the installation by running this command:

```
# python3.7 -v
```

Python syntax basics

Python uses indentation and spacing to group blocks of code together.

- If you get runtime errors, check your spacing first.
- Next, check your indentation for any missing punctuation, such as colons.

Python is also case sensitive. Capitalization matters.



Identifiers

In Python, *identifier* is the name for entities like class, functions, and variables. It helps differentiate one entity from another entity.

When you name objects in Python, you must observe some rules:



An identifier cannot start with a digit. **1variable** is invalid, but **variable1** is valid.



Keywords cannot be used as identifiers.



You cannot use special symbols like **!**, **@**, **#**, **\$**, and **%** in your identifiers.



Identifiers can be any length.



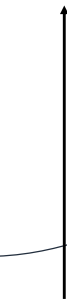
File extensions: .py files

The extension `.py` is a script file extension that is used in Python.

Files with the `.py` extension can be created in any text editor, but they need a Python interpreter to be run.

Functions

```
>>> print ()  
>>> print (10)
```



argument

- A function tells the computer to do a specific task.
- Functions have a *name* and are invoked by adding parentheses after the name.
- Many functions take *arguments*, which are information that the function uses to do its specific task.
- One useful function is the **print** function.
- You can write your own functions.
- Python also has native functions (such as **print**) that are already built in.

Vim review

The text editor that you will use is Vim (Vi improved).

- To open it on the command line, enter `vim <file name>` or `vim`
- Vim opens in normal mode. To write to Vim, enter `i` (the *i* stands for *insert*)
- To get back to normal mode, press ESC
- When you are in normal mode, you can save and quit, or continue editing
- To save a file, enter `:w <file name>` or `:w` (the *w* stands for *write*)
- To close Vim, enter `:q` (the *q* stands for *quit*)
- To save and close, enter `:wq`

Demo: Hello World



10

1. Create a file in CentOS that is called *helloworld.py*.
2. Open that file and write to it:
`print("Hello World")`
3. Save the file and return to your command line.
4. Run the file with Python and observe what happens.



1. Enter the following command:
`$ vim helloworld.py`
(This command creates the file. Enter *i* to insert text.)
2. Enter **`print("Hello World")`** and press ENTER.
3. Press ESC, then enter: **`:w`**
(`:w` writes the file.)
4. Then enter: **`:q`**
(`:q` quits and puts you back on the command line.)
5. At the command line, enter this command:
`$ python3.7 helloworld.py`
(This command runs the file with the Python interpreter.)

Comments

```
# This is a comment
```

- Comments are notes to yourself and to other developers.
- Comments describe the contents of a program and how it works so that a person who reviews the source code can understand it.
- Use the pound (#) symbol to start writing a comment.
- By starting a line of text with #, you tell the interpreter *not* to run this line as code.

Demo: Hello World and Commenting



1. Open your *helloworld.py* file.
2. Insert more text by entering *i*
3. On another line, enter some comments.
 - For example: `# I love Python`
4. Write and exit.
5. Run your file again by using the following command:
`python3.7 helloworld.py`
6. Did anything change? Why or why not?

Data types

Data types determine what kind of data is stored in an object.

Python stores the type of an object with the object. When the operation is performed, it checks whether that operation makes sense for that object. (This technique is called *dynamic typing*.)

Example: Some functions or methods in Python expect a certain data type. You cannot pass a string into a function that is expecting a float.

Basic data types

Integer (int):

A whole number. Can be negative.

Examples:
4
3000
-29

Float:

A number that contains a decimal. Can also be negative.

Examples:
3.390
8.090
-0.001

Boolean (bool):

A condition of *True* or *False*.

Capitalization matters.

String (str):

A set of characters that are enclosed in quotation marks (" " or ' ' or "" ""). Can include numbers, letters, and special characters.

Examples:
"I am a string"
'23456'
""I have 56 sheep""

Note: Numbers inside a string are still a string. Python does not consider them to be integers or floats.

Data types: Mutable versus immutable

Mutable means *changeable*. In Python, some data types are *mutable*, and others are *immutable*.

Mutable

- List
- Set
- Dictionary
- Byte array

Immutable

- Integers, floats, complex
- Strings
- Tuples
- Frozen sets

Converting data types

Python has specific commands that you can use to change an object from one data type to another.

- **float()**: In the example, `4` is assigned to the variable `x`. By its nature, `4` is an integer. However, `x` must be identified as a float, so the **float()** command was used. The result is `4.0`.
- **int()**: By passing `x` back into the **int(x)** command, it receives `4` back. This command truncates the decimals.
- **Note**: Using **int()** is not the same as rounding. It only removes any decimal digits that the number has. It does not round the decimal numbers up or down, so be careful.

Lists, sets, tuples, and other data types will be covered later.

```
>>> x = 4
>>> float(x)
4.0
>>>
```

```
>>>
>>> int(x)
4
>>>
```

Strings

```
>>> "I'm a string!"
>>> 'Me too'
>>> '''Me too!'''
>>> "I have written 3 strings"
```

- Strings contain *text*, which can be a single character or paragraphs of text.
- Strings are characters that are enclosed between quotation marks.
- Strings can contain numbers.
- Three ways of notating a string are:
 - Single quotation marks (' ')
 - Double quotation marks (" ")
 - Triple quotation marks (" " " ")

The triple quote notation allows a string to span multiple lines and include non-printable formatting characters such as Newline and Tab.

String concatenation

```
>>> "I'm a string!"
>>> 'Me too'
>>> "I'm a string" + "Me too"
>>> "I'm a stringMe too"
```

Note: If you want a space between the words of your new string, you must add it into one of the original strings. Otherwise, Python adds them as-is, without inserting a space.

- Strings are *immutable*. When you manipulate them, you create a new string.
- You can *concatenate* or *add* strings together, which creates a new string.
- It is possible to create an empty string.
 - Example: `x = ""`

A simple Python program

```
>>> 1 + 1  
>>> 2
```

- This simple Python program adds the numbers *1* and *1* together.
- The result is 2, as displayed on the following line.
- The angle brackets (`>>>`) are not part of the program. The Python interpreter displays them.

Variables

```
>>> x = 1
>>> x + x
2
```

- In this example, *x* is a variable that holds the number *1*.
- When you do something with variables, Python looks up the values to understand what to do.

Variables cont.

```
>>> apples = 2
>>> oranges = 3
>>> apples = oranges
>>> car_color = red
```

- You can have as many variables as you want.
- You can name your variables whatever you want, within certain restrictions.
- Restrictions are:
 - A variable name can only contain letters, numbers, and underscores (_).
 - A variable name cannot begin with a number.
 - A variable name cannot be a keyword in Python.

Demo: Variables



22

Instructions

1. Create a new file that is called: *variables.py*
2. Make some variable assignments of your choice. Use strings, floats, or ints.
3. Examples:
`x = 33`
`y = 7.0`
`variable_challenge = "I love Python"`
4. Use the **print()** function to tell Python to print your variables. Otherwise, when you call the file, you will not see any output.
Examples:
`print(x)`
`print(y)`
`print(variable_challenge)`
5. Write and exit your file. Call the file from the command line. What happened?



Example code:

```
x = 33
y = 7.0
variable_challenge = "I love Python"
```

```
print(x)
print(y)
print(variable_challenge)
```

Operators

Arithmetic operators	<code>+, -, *, /, %, **, //</code>
Comparison (relational) operators	<code><, >, <=, >=, ==, !=</code>
Assignment operators	<code>=, +=, -=, *=, /=, %=, //=, **=, &=, =, ^=, >>=, <<=</code>
Logical operators	<code>and, or, not</code>
Membership operators	<code>in, not in</code>
Identity operators	<code>is, is not</code>

Python also has bitwise operators that allow the manipulation of a number at the bit level.

Operator precedence

Operator	Description
(expressions...), [expressions...], {key: value...}, {expressions...}	Parenthesized expression, list, dictionary, set
**	Exponentiation
+, -, ~	Positive, negative, bitwise NOT (Unary operators)
*, /, %, //	Multiplication, division, remainder, floor division
+, -	Addition, subtraction
<<, >>	Left and right bitwise shift
in, not in, is, is not, <, <=, >, >=, !=, ==	Membership operators, identity operators, comparison operators
and, or, not	Logical (Boolean) operators
=, +=, -=, *=, /=, %=, //=, **=, &=, =, ^=, >>=, <<=	Assignment operators

This table summarizes the order of precedence for the major operators in Python.

Note that membership, identity, and comparison operators have the same order of precedence and are, therefore, evaluated from left to right.

Demo: Basic Math



25

Instructions

1. Create a new file that is called: *basicmath.py*
2. Insert text into the file. Try writing some basic math functions.
Examples:
1+1
100/33
2**2
3. Now, tell Python to print these expressions by using the **print()** function.
4. Write and exit the file.
5. Run it. What happened?



Create a new file with the following code:

```
print(1 + 1)
print(100/33)
print(2**2)
```

Demo: Variables + Basic Math



26

Instructions

What happens when you combine variable assignment with math?

1. Open your *basicmath.py* file.
2. Edit the file to assign variable names to your basic math functions.

Examples:

```
easy_math = 1+1  
one_third = 100/33  
powerful = 2**2
```

3. Tell Python to print those variable names.
4. You can also do math with only the variable names.

Examples:

```
x = 2  
y = 4  
z = x+y  
print(x + y)  
print (z)
```

5. Do you expect printing *z* and *x+y* to display the same result? Why or why not?

Demo: Strings and Concatenation



27

Instructions

1. Create a new file: *funwithstrings.py*
2. Create a variable and assign a string to it.
Example:
`first_str = "I love dogs"`
3. Create a second variable and assign a different string to it.
Example:
`second_str = " and cats"`
4. Create a third variable that is called *newstr* and assign it the concatenation of strings 1 and 2.
Example:
`newstr = first_str + second_str`
5. Print *newstr*. What happened?

Statements

```
>>> apples = 2
>>> oranges = 3
>>> apples = oranges
```

- A *statement* is usually a line of code, and each line of code that you have seen so far is a single statement.
- A statement is an individual instruction to Python.
- This code has three statements.
- By assigning *apples* to *oranges*, you are changing the value of *apples*. If you `print(apples)`, what is the response?

Functions, strings, and variables

```
>>> name = "Kwesi"  
>>> print (name)
```

- Store strings in variables.

Exceptions

- An exception raises an error.
- Examples include:
 - Referencing a variable by the incorrect name
 - Dividing a number by zero
 - Trying to read a file that does not exist on your computer
- Exceptions result in a *stack trace*, which is a listing of the various ways that something went wrong.

Demo: Find the Exceptions



31

Instructions

- Do some research online for Python exceptions. Find as many different kinds of exceptions as you can. Be prepared to discuss your findings with the class.

Checkpoint questions



Does Python limit you from using specific characters when you declare a variable?



How are functions called in Python?



Which data type do you use to store the value 1.7—an integer or a float?



What is the `+=` operator used for?

Answers:

1. Yes, you cannot use symbols such as `!`, `@`, `#`, `$`, and others in variable declarations.
2. Functions are called by name.
3. A float is used to store decimal values.
4. The plus equals (`+=`) operator adds the *rvalue* (the value to the right of the operator) to the *lvalue* (the value to the left of the operator). The *lvalue* is usually a variable. For example, if `x` is 5, consider the following code:
`x+=10`
After that code is run, the value of `x` is 15.

Key takeaways



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

33

- Python can be installed on Microsoft Windows, macOS, and Linux computers.
- Python files have the `.py` extension.
- Python uses indentation and spaces, and is case sensitive.
- Functions are groups of commands that can be called by name.
- Data types identify the type of data (integer, float, and so on) that a variable contains.
- Operators in Python are used for math, equivalence tests, and string operations.
- Exceptions are errors that the Python interpreter raises when an error occurs in the code.

aws re/start

Some key takeaways from this lesson include:

- Python can be installed on Microsoft Windows, macOS, and Linux computers.
- Python files have the `.py` extension.
- Python used indentation and spaces, and is case sensitive.
- Functions are groups of commands that can be called by name.
- Data types identify the type of data (integer, float, and so on) that a variable contains.
- Operators in Python are used for math, equivalence tests, and string operations.
- Exceptions are errors that the Python interpreter raises when an error occurs in the code.