PYTHON PROGRAMMING

THE COMPLETE CRASH COURSE FOR BEGINNERS TO MASTERING PYTHON WITH PRACTICAL APPLICATIONS TO DATA ANALYSIS & ANALYTICS, MACHINE LEARNING AND DATA SCIENCE PROJECTS

4 BOOKS IN 1

ANDREW PARK

PYTHON FOR BEGINNERS

ANDREW PARK

PYTHON ADVANCED GUIDE

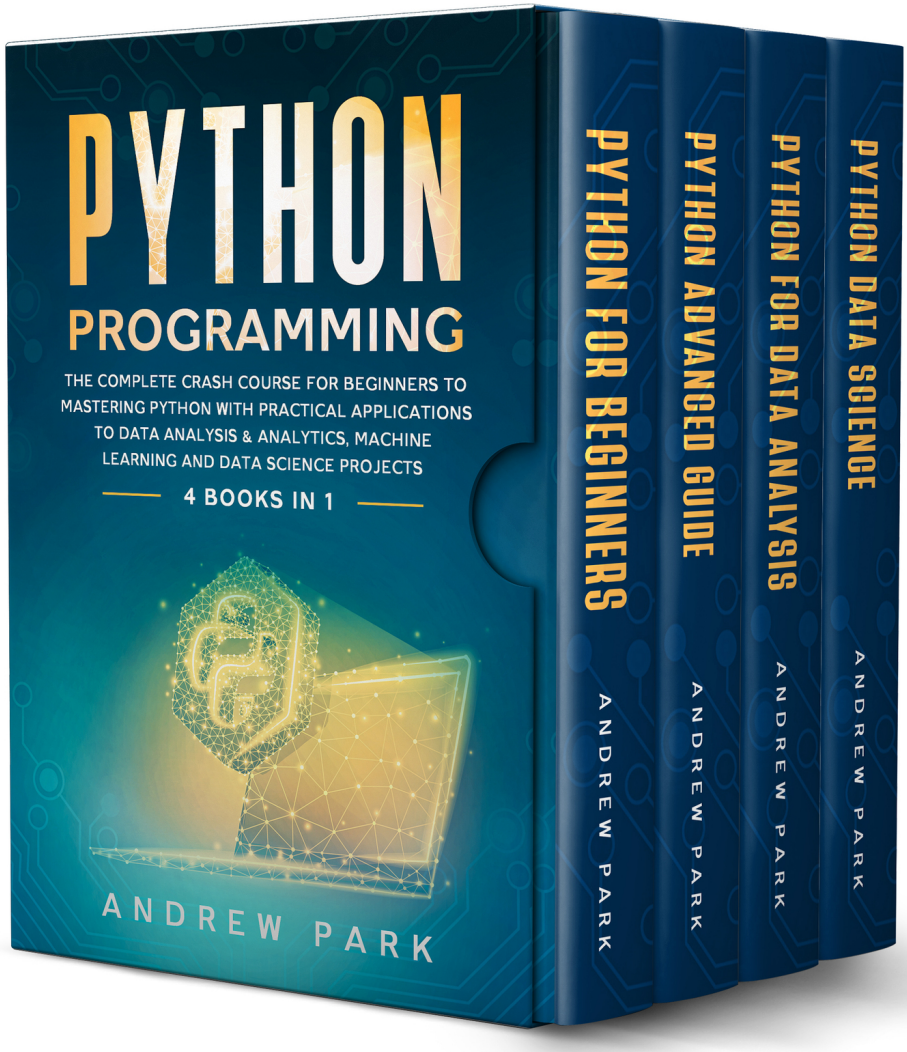
ANDREW PARK

PYTHON FOR DATA ANALYSIS

ANDREW PARK

PYTHON DATA SCIENCE

ANDREW PARK


PYTHON
PROGRAMMING
THE COMPLETE CRASH COURSE FOR BEGINNERS TO MASTERING PYTHON WITH PRACTICAL APPLICATIONS TO DATA ANALYSIS & ANALYTICS, MACHINE LEARNING AND DATA SCIENCE PROJECTS
4 BOOKS IN 1
ANDREW PARK
PYTHON FOR BEGINNERS
ANDREW PARK
PYTHON ADVANCED GUIDE
ANDREW PARK
PYTHON FOR DATA ANALYSIS
ANDREW PARK
PYTHON DATA SCIENCE
ANDREW PARK

Python Programming

The Complete Crash Course for Beginners to Mastering Python with Practical Applications to Data Analysis & Analytics, Machine Learning and Data Science Projects
4 Books in 1

Andrew Park

Python for Beginners

Python Advanced Guide

Python for Data Analysis

Applications to Data Science

Python for Beginners

Table of Contents

Introduction

Python is one of the top programming languages, universities and industries are preferring to teach it and use respectively. The charm of Python is hidden in the fact that it has extremely large applications in a wide range of fields. Most people abhor Python because of its use in building artificial intelligence models. They fear that these Python-powered AI models will drive people out of different industries and snatch their jobs. They quote the example of Tesla's driverless taxi program by which Tesla pretends to replace Uber's taxis in the US market. But the reality is different. In fact, Python-powered AI models will create many more jobs instead of removing jobs For example, building these models will become an independent industry. Also, the implementation of these AI models will become a new business sector.

Data science is going to take the corporate world by storm. Data science is based on Python programming language, as more and more companies are now moving in a neck-on-neck competition. All they crave is a way to take an edge over their competitors. They do each thing for power and to get ahead. In this regard, Python seems to be promising. Python-backed data science tends to equip industries with sophisticated data about past and present sales patterns, which can help corporate sector CEOs make wiser decisions about sales and development of marketing strategies.

The biggest advantage for learners of Python is that you don't have to compile the code. In C++, you have to compile the entire program first and then run it. Only then you will be able to see whether your program runs or returns an error. Python offers the same level of programming and even at a higher stage, but still, it is an interpreted language that can be easily written, edited, and corrected.

Python is very easy to read and learn. You can easily read source codes for different programs that are created by other programmers. But no matter how easy it is on the outside to read and learn, it needs, like all the other programming languages, dedicated practice. You will have to get to the Python editor and practice all codes. In the beginning, you can take the code and just paste it in the editor to see the results. In the second phase, you can make minor edits to the code and see the results. In the third phase, you will

be able to completely reshape a program and see how it runs in the Python shell. Given the increasing applications of Python, learning it is extremely profitable from the angle of the global job market. Python can give you the much-needed edge over others when it comes to securing high paid jobs.

Chapter 1 Python Installation

The up-to-date edition with the binaries, current source codes, documentation, latest news, etc. can be accessed at the official Python website: http://www.Python.org/

Here, you can download the documentation for Python from the site. It is accessible in various formats like PDF, HTML, and PostScript.

Python Documentation Website: www.Python.org/doc/

How to Install Python

Python Environment Variables:

Variable	Description
PYTHONPATH	It is similar to PATH. This variable foretells the Python interpreter as to where it can locate the module files that you can import in a program. PYTHONPATH must have a Python source library directory and the directories in which your Python code is present. PYTHONPATH is sometimes set by the Python installer automatically.
PYTHONSTARTUP	It has the path for the initialization file which contains Python source codes that are executed every time you work on the interpreter. This file is often named .Pythonrc.py in Unix, which usually has commands that can load the utilities or can modify the PYTHONPATH.
PYTHONCASEOK	This is used in Windows OS for instructing Python to find case-insensitive matches in an import statement. You can set this to any random value for activating it.

PYTHONHOME	It is an alternative search path that is embedded in PYTHONSTARTUP or PYTHONPATH directories for switching the module libraries very easy.

Running Python

Python can be run on a system in three ways.

Interactive Interpreter

You can start by entering Python and then begin programming in its interpreter by beginning from the command line on any platform that provides a command line interpreter or a shell window.

A list of command line options is given in the table below.

Option	**Description**
-d	Provide the output after debugging
-O	Optimized byte code generation i.e. the .pyo file is generated
-S	Don’t run the import site for searching Python paths in a startup.
-v	Details of the import statement
-X	Disable the class-based built-in exceptions
-c cmd	It runs Python script sent in cmd String

File	The python script is run from the given file

Script from Command-line

Calling an interpreter on your application can help you run and execute your Python script in the command line.

Integrated Development Environment

It is possible to run Python from a GUI too. The one thing you require is a system that supports Python.

Chapter 2 Python Data Types

Python supports different data types. Each variable should belong to one of the data types supported in Python. The data type determines the value that can be assigned to a variable, the type of operation that may be applied to the variable as well as the amount of space assigned to the variable. Let's discuss different data types supported in Python.

Python Numbers

These data types help in the storage of numeric values. The creation of number-objects in Python is done after we have assigned a value to them. Consider the example given below:

```
total = 55

age = 26
```

The statement can be used for the deletion of single or multiple variables. This is shown below:

```
del total

del total, age
```

In the first statement, we are deleting a single variable while in the second statement, we are deleting two variables. If the variables to be deleted are more than two, separate them by using a comma and they will be deleted.

In Python, there are four numerical values which are supported:

- Int
- Float
- Complex

In Python 3, all integers are represented in the form of long integers.

The Python integer literals belong to the int class.

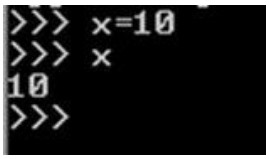
Example

Run the following statements consecutively on the Python interactive interpreter:

```
x=10

x
```

You can run it on the Python interactive interpreter and you will observe the following:

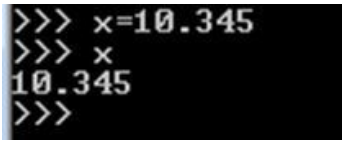


The float is used for storing numeric values with a decimal point.

Example

```
x=10.345

x
```



If you are performing an operation with one of the operands being a float and the other being an integer, the result will be a float.

Example

```
5 * 1.5
```

As shown above, the result of the operation is 7.5, which is a float.

Complex numbers are made of real and imaginary parts, with the imaginary part being denoted using a j. They can be defined as follows:

```
x = 4 + 5j
```

```
>>> x = 4 + 5j
>>> x
(4+5j)
>>>
```

In the above example, 4 is the real part, while 5 is the imaginary part.

In Python, there is a function named type() that can be used for determining the type of a variable. You only have to pass the name of the variable inside that function as the argument and its type will be printed.

Example

```
x=10

type(x)
```

```
>>> x=10
>>> type(x)
<class 'int'>
>>>
```

The variable x is of int class as shown above. You can try it for other variable types as shown below:

```
name='nicholas'

type(name)
```

```
>>> name='nicholas'
>>> type(name)
<class 'str'>
>>>
```

The variable is of the string class as shown above.

Python Strings

Python strings are a series of characters enclosed within quotes. Use any type of quotes to enclose Python strings, that is, either single, double or triple quotes. To access string elements, we use the slice operator. String characters begin at index 0, meaning that the first character string is at index 0. This is good when you need to access string characters. To concatenate strings in Python, we use + operator, the asterisk 9*) is used for repetition.

Example

```
#!usrbin/Python3
thanks = 'Thank You'
print (thanks) # to print the complete string
print (thanks[0]) # to print the first character of the string
print (thanks[2:7]) # to print the 3rd to the 7th character of the string
print (thanks[4:]) # to print from the 5th character of the string
print (thanks * 2) # to print the string two times
print (thanks + "\tAgain!") # to print a concatenated string
```

The program prints the following once executed:

```
Thank You
T
ank Y
k You
Thank YouThank You
Thank You       Again!
```

Notice that we have text beginning with the # symbol. The symbol denotes the beginning of a comment. The Python print will not act on the text from the symbol to the end of the line. Comments are meant at enhancing the readability of code by giving explanations. We defined a string named *thanks* with the value *Thank You* . The *print (thanks[0])* statement helps us access the first character of the string; hence it prints T. You also notice that the space between the two words is counted as a character.

Chapter 3 Python Variables

Understanding Python variables, classes, and how to operate is essential for both beginners and programmers who intend to expand their programming skills.

What Is A Variable in Python?

When writing complex codes, your program will demand data essential to conduct changes when you proceed with your executions. Variables are, therefore, sections used to store code values created after you assign a value during program development. Python, unlike other related language programming software, lacks the command to declare a variable as they change after being set. Besides, Python values are undefined like in most cases of programming in other computer languages.

Variation in Python is therefore described as memory reserves used for storing data values. As such, Python variables act as storage units, which feed the computer with the necessary data for processing. Each value comprises of its database in Python programming, and every data are categorized as Numbers, Tuple, Dictionary and List, among others. As a programmer, you understand how variables work and how helpful they are in creating an effective program using Python. As such, the tutorial will enable learners to understand declare, re-declare, and concatenate, local and global variables as well as how to delete a variable.

Variable vs. Constants

Variables and constants are components used in Python programming but perform different functions. Variables, as well as constants, utilize values used to create codes to execute during program creation. Variables act as essential storage locations for data in the memory, while constants are variables whose value remains unchanged. In comparison, variables store reserves for data while constants are a type of variable files with consistent values written in capital letters and separated by underscores.

Variables vs. Literals

Variables also are part of literals which are raw data fed on either variable or constant with several literals used in Python programming. Some of the common types of literals used include Numeric, String, and Boolean, Special and Literal collections such as Tuple, Dict, List, and Set. The difference between variables and literals arises where both deal with unprocessed data but variables store the while laterals feeds the data to both constants and variables.

Variables vs. Arrays

Python variables have a unique feature where they only name the values and store them in the memory for quick retrieval and supplying the values when needed. On the other hand, Python arrays or collections are data types used in programming language and categorized into list, tuple, set, and dictionary, which will be discussed later. When compared to variables, the array tends to provide a platform to include collectives functions when written while variables store all kinds of data intended. When choosing your charming collection, ensure you select the one that fits your requirements henceforth meaning retention of meaning, enhancing data security and efficiency.

Classifications of Python Arrays E ssential for Variables

Lists

Python lists offer changeable and ordered data and written while accompanying square brackets, for example, "an apple," "cherry." Accessing an already existing list by referring to the index number while with the ability to write negative indexes such as '-1' or '-2'. You can also maneuver within your list and select a specific category of indexes by first determining your starting and endpoints. The return value with therefore be the range of specified items. You can also specify a scale of negative indexes, alter the value of the current item, loop between items on the list, add or remove items, and confirming if items are available.

Dictionaries

Python dictionaries comprise of indexed, changeable but unordered items typically written while with curly brackets with keys and values. Some of

the activities involved include item access by use of a keyword inside the parentheses; conduct value changes, loop, check critical availability, length of the dictionary, and both adding and removing unwanted items. Besides, Python allows you to copy the dictionary by writing 'dict2 = dict1'. 'dict2' will become a representation to 'dict1' therefore makes any necessary changes automatically. Another way of creating a copy is also by using a built-in Dictionary technique, that is, 'copy.'

In other instances, Python dictionaries can also have other dictionaries within it a process referred to as nested dictionaries. You can readily determine the number of dictionaries present in the nest through creating of three already available. You can also generate your dictionary through the 'dict()' contractor function. The function enables the copying of the previous dictionary or the creation of a completely new one. Within the Python dictionary, there exist several built-in techniques to implement and enjoy the efficiency of the dictionaries present.

Naming Variables

The naming of variables remains straightforward, and both beginners and experienced programmers can readily perform the process. However, providing titles to these variables accompany specific rules to ensure the provision of the right name. Consistency, style, and adhering to variable naming rules ensure that you create an excellent and reliable name to use both today and the future. The rules are:

- Names must have a single word, that is, with no spaces
- Names must only comprise of letters and numbers as well as underscores such as (_)
- The first letter must never be a number
- Reserved words must never be used as variable names

When naming variables, you should bear in mind that the system is case-sensitive, hence avoid creating the same names within a single program to prevent confusion. Another important component when naming is considering the style. It entails beginning the title with a lowercase letter while using underscores as spaces between your words or phrases used. Besides, the program customarily prevents starting the name with a capital letter. Begin with a lowercase letter and either mix or use them consistently.

When creating variable names, it may seem so easy, but sometimes it may become verbose henceforth becoming a disaster to beginners. However, the challenge of creating sophisticated names is quite beneficial for learned as it prepares you for the following tutorials. Similarly, Python enables you to write your desired name of any length consisting of lower- and upper-case letters, numbers as well as underscores. Python also offers the addition of complete Unicode support essential for Unicode features in variables.

As already discussed, specific rules are governing the procedure for naming variables; hence adhere to them to create an exceptional name to your variables. Create more readable names that have meaning to prevent instances of confusion to your members, especially programmers. A more descriptive name is much preferred compares to others. However, the technique of naming variables remains illegible as different programmers decide on how they are going to create their kind of names.

Methods of Creating a Multi-Name for Python Variables

- Pascal case: this method entails the first, second, and subsequent words in the name as capitalized to enhance readability. For example, ConcentrationOfWhiteSmoke.
- Camel case: the second and subsequent words of the name created remains capitalized. For example, the ConcentrationofWhiteSmoke.
- Snake case: snake method of creating variable names entails separator of words using an underscore as mentioned earlier. For example, concentration_of_white_smoke.

Learning Python Strings, Numbers and Tuple

Python strings are part of Python variables and comprise of objects created from enclosing characters or values in double-quotes. For example, 'var = Hello World’. With Python not supporting character types in its functions, they are however treated as strings of one more characters as well as substrings. Within the Python program, there exist several string operators making it essential for variables to be named and stored in different formats. Some of the string operators commonly used in Python are [], [:], ‘in’, r/R, %, + and *.

There exist several methods of strings today. Some include replacing Python string () to return a copy of the previous value in a variable,

changing the string format, that is, upper and lower cases and using the 'join' function, especially for concatenating variables. Other methods include the reverse function and split strings using the command' word.split'. What to note is that strings play an important role, especially in naming and storage of values despite Python strings being immutable.

On the other hand, Python numbers are categorized into three main types; that is, int, float, and complex. Variable of numbers are usually created when assigning a value for them. For instance, int values are generally whole numbers with unlimited length and are either positive or negative such as 1, 2, and 3. Float numbers also either positive or negative and may have one or more decimals like 2.1, 4.3 and 1.1 while complex numbers comprise both of a letter 'j' as the imaginary portion and numbers, for example, 1j, -7j or 6j+5. As to verify the variable number is a string, you can readily use the function 'type().'

A collection of ordered values, which remain unchangeable, especially in Python variables, is referred to as a tuple. Python tuples are indicated with round brackets and available in different ways. Some useful in Python variables are access tuple items by index numbers and inside square brackets. Another is tuple remaining unchanged, especially after being created but provides a loop by using the function 'for.' And it readily encompasses both count and index methods of tuple operations.

Types of Data Variables

String

A text string is a type of data variable represented in either String data types or creating a string from a range of type char. The syntax for string data comprises multiple declarations including 'char Str1[15], 'char Str5[8] = “ardiono”; among others. As to declare a string effectively, add null character 'Str3', declare arrays of chars without utilizing in the form of 'Str1' and initialize a given array and leave space for a larger string such as Str6. Strings are usually displayed with doubles quotes despite the several versions of available to construct strings for varying data types.

Char

Char are data types primarily used in variables to store character values with literal values written in single quotes, unlike strings. The values are stores in numbers form, but the specific encoding remains visibly suitable for performing arithmetic. For instance, you can see that it is saved as 'A' +, but it has a value of 66 as the ASCII 'A' value represents 66. Char data types are usually 8 bits, essential for character storage. Characters with larger volumes are stored in bytes. The syntax for this type of variable is 'char var = val'; where 'var' indicates variable name while 'val' represents the value assigned to the variable.

Byte

A byte is a data type necessary for storing 8-bit unsigned numbers that are between 0 to 255 and with a syntax of 'byte var = val;.' Like Char data type, 'var' represents variable name while 'val' stands for the value to he assigned that variable. The difference between char and byte is that char stores smaller characters and with a low space volume while byte stores values which are larger.

int

Another type of data type variable is the int, which stores 16-bit value yielding an array of between -32,768 and 32,767, which varies depending on the different programming platforms. Besides, int stores 2's complement math, which is negative numbers, henceforth providing the capability for the variable to store a wide range of values in one reserve. With Python, this type of data variable storage enables transparency in arithmetic operations in an intended manner.

Unsigned int

Unsigned int also referred to, as unsigned integers are data types for storing up to 2 bytes of values but do not include negative numbers. The numbers are all positive with a range of 0 to 65,535 with Duo stores of up to 4 bytes for 32-byte values, which range from 0 to 4,294,967,195. In comparison, unsigned integers comprise positive values and have a much higher bit. However, ints take mostly negative values and have a lower bit hence store chapters with fewer values. The syntax for unsigned int is 'unsigned int var = val;' while an example code being 'unsigned int ledPin = 13;'

Float

Float data types are values with point numbers, that is to say, a number with a decimal point. Floating numbers usually indicate or estimate analog or continuous numbers, as they possess a more advanced resolution compared to integers. The numbers stored may range from the highest of 7.5162306E+38 and the lowest of -3.2095174E+38. Floating-point numbers remain stored in the form of 32 bits taking about 4 bytes per information fed.

Unsigned Long

This is data types of variables with an extended size; hence it stores values with larger storages compare to other data types. It stores up to 32 bits for 4 bytes and does not include negative numbers henceforth has a range of 0 to 4,294,967,295. The syntax for the unsigned long data type is 'unsigned long var = val;' essential for storing characters with much larger sizes.

Declaring Variables

As stated, variables are the naming and storing data values of both numerical and letters primarily used during Python programming. Before those values are used, they have to be declared to identify and perform the desired function in your program. Therefore, declaring a value means defining the type, and sometimes setting or initializing the value, though it remains optional but crucial. When creating your program, ensure you understand the extent of your variables by considering the scope of numbers you are storing. Excessive storage of values may result in rollovers, as the space used is insufficient.

The size of where to declare your values also affects a programmer's outcome on created programs. The technique of selecting a specific storage unit influences the function of applications, especially when determining the codes. The scope henceforth is an essential aspect of declaring variables as it affects the results of your program. Another form is through initializing variables to decide on which value to start with during declaration. Initialized variables make it easy for programmers to readily choose a starting point when declaring or used for other purposes.

Creating and Declaring Variable

Python programs have limited access to a direct command to declare or create variables instantly. However, some essential rules may become a critical component for the process to occur. Besides, Python does not necessarily require data type specification but created immediately when the value is assigned. When assigning values using Python, especially for complex or multiple assignments, it uses inferred language techniques, for instance, in detecting types of values assigned in a given variable.

Variables Assignment

Variables are neither declared nor defined when utilizing Python programming henceforth creation is quite straightforward. Creating a variable is simply assigning a value and begins using it. The process uses a single equal (=) symbol useful for statements and expressions. For example, creating n=38 in Python suggests that 'n' is assigned the value '38' and the value can be readily be substituted during programming.

Like literals values, the value used may be displayed directly by the interpreter without the use of 'print().' However, if you change the value, instead of 'n=38', the value will be substituted; for instance, if you input value 1000, it will display 'n=1000'. Python henceforth allows you to make changes where needed as well as the operation of chained assignments and input a similar value to different variables simultaneously. For example, a=b=c=d=38.

Re-Declaring Variables

After you have even declared a variable in Python, you can make changes by either declaring it again or assign a substitute value. That is, you can replace or connect a different value to the previous one readily through the re-declaration process. Re-declaring variables are beneficial as it enables you to accept user-generated codes to already existing values initialized. Similarly, you may wish to change the program or make some alterations during your project.

Reassigning variables are more vital for complicated or extensive programs already incorporated by another programmer, and you are taking over. As such, you make significant changes in the declared values to enhance the

effectiveness of your program is created. The Python interpreter plays a substantial role in discarding the original value and adding the new ones. The type of new values attached may either be different or comprise of unique identity when compared to the old ones.

For example, if your original value was 'x' and you need to change it to be an integer of '76', you first reassign 'x' to be a string, ass your new value and a replacement of the value immediately becomes a success. The example suggests that value 'x' undergoes an assignment to the value of an integer and later reassigned with the value of the string. Variable re-declare is most effective when you are aware of the readability of codes and with an object to create clear programs.

Concatenate Variables

If you wish to concatenate variables of several data types through Python, for instance, several variables and string variables, then you need to declare the values into strings. In case the number variables are not declared when concatenating different values, Python programming would stop and display TypeError indicating the procedure is unsuccessful. As to correct the situation, you will have to declare the number variables as a string.

The process at Python programming is quite different when compared to other programs such as Java and C++, which immediately concatenates several numbers without declaring into strings. For instance, if you need to concatenate 'computer and '58', however when you declare the integer as a string, it can readily concatenate the two in the form of "computer + str(58) = computer58".

Global Variables

Global variables utilized in Python are a multipurpose variable used in any part of the world while anywhere. The variable used can operate in your program or module while in any part of the globe henceforth using values whenever you travel as a programmer. Global variables are useful for programmers to create their programs while moving from one location to another. Some of the benefits include variables that are used across function or module as well as it does not require re-declarations for performance.

When compared to local variables, global variables have an 'f' scope and assigned value 101, displayed as 'f=101', printed as an output. For example, when you re-declare a variable as a global variable in a given function, change it within the role and print it outside the task. The variable would provide a third party outcome useful globally. Therefore, global variables are found outside functions indicating that not all variables are readily accessed from anywhere globally. As a beginner, it is crucial to understand the difference between global and local variables to develop the necessary variables suitable for your programs.

Local Variables

Unlike global variables, local variables are used locally, declared within a Python function or module, and utilized solely in a specific program or Python module. When implemented outside particular modules or tasks, the Python interpreter will fail to recognize the units henceforth throwing an error message for undeclared values. Like global variables, local variables use the 'f' variable where it is declared to assume local scope and assigned 'I am learning Python' and then recognized as a local variable.

For example, when you declare the variable ‘f’ the second time, it changes to a new function and results in a local variable. As such, when you accept that variable in the inside function, the process will run without any problems. The execution is made possible as the second print(f) produces a value assigned to 'f' as *Intellipaat.* Whereas, when you print the value outside the function 'f', it results in a value assigned to it, which is outside the function, that is, a third print(f). In that case, local variables are used only in two surrounding environments of Python programming with those outside the function leading to failure of operations unless declared.

Using Variables

Immediately the variables are declared, they can be readily defined by setting those equal to the value you intend to store with the single same sign referred to as the assignment operator. The single equal sign, therefore, enables the program to put the desired variable either on the right or left side on whichever side. After seeding the variables, that is, assigned each variable with a value, test-specific values to determine if suits the program or use it directly.

For instance, you can use certain codes to test if the inputVariable2 is less than 50, thus set a delay time based on it with a minimum of 50. The example therefore tests the variable ‘if (inputVariable2 <50)’ and sets a given variable if it is successful ‘inputVariable2 = 50’ and delays the results using the function ‘delay(inputVariable2)’. When using variables, ensure they have a more descriptive name for readability purposes. It also enables you, and someone else understands what the variable entails as well as in recognition during the programming process.

Deleting Variables

On the other hand, if you make a mistake, get rid of your current project or just doing away with everything, Python provides a useful feature to delete any variable and create more space for storage of other values. Similarly, there are some unwanted variables, and you wish to get rid of, the delete variable is henceforth the choice for you. When deleting a variable, you should be cautious as to avoid deleting essential values within a variable. As such, ensure you know the name of the variable to delete.

If you are deleting a file with a variable name, which does not match, the Python interpreter typically throws an error message;' NameError: name (filename) is not identified.' As to delete a variable effectively, there exists a command within the program; del' variable name' and the variable will remove instantly. Ensure you input the correct name before running the command. As to confirm if your file is deleted, you may try to print, and if you see an error message, it indicates that the variable was deleted successfully.

Functions and Sets In Python Variables

There are minimal chances of a beginner to understand variables and the general operation of creating Python programs before understanding what are sets and functions as used in variables. When mentioning sets, objects, and functions, it means a different thing when it comes to computer programming. One of the most definitions of functions in variables is the set of related statements, values, and codes grouped and stored together to perform a particular task.

On the other hand, a set is a collective term to refer to Python data types with mutable free from duplication arranged in an unordered manner. Since you are learning about Python variables, understanding functions and sets contributes to familiarizing with these terms; hence, you can easily interact with Python programming languages. Bear in mind that sets for had provided a particular index order of data while functions may not demand any input but solely executed when precisely needed.

Some of the standard features of sets include lack of a specific order of each value, number, code, and the item is unique, collections are immutable, and they offer modifications such as addition and deletion. The main advantage of utilizing sets in Python variables is that it optimizes the method of checking element is included in the right set or not. Also, it enables you to with we ass or remove some aspects but should only be unique and immutable.

Functions make programs more manageable and organized mostly for programmers working on complicated tasks. Such tasks are with smaller and modular lumps of codes, thus increasing readability and reusability. As such, Python offers three types of functions; Python Built-in, user-defined and anonymous functions. The functions can only be initially named, written, and executed to perform a particular kind of input and create a desired program essential for both beginners and programmers.

Chapter 4 Basic Operators of Python Language

Python is considered a high-level programming language with less complexity when it comes to using the basic operators in the code. It is built to read and implement computer language easily. Python provides various types of operators for performing tasks. Let's see the basic operators provided by Python.

Types of Operators

1. Python Arithmetic Operators
2. Python Assignment Operators
3. Python Comparison Operators
4. Python Logical Operators
5. Python Bitwise Operators
6. Python Membership Operators
7. Python Identity Operators

Python Arithmetic Operators

Arithmetic operators help us to solve several types of mathematical problems like addition, subtraction, multiplication, exponential values, floor divisions, etc.
Let's suppose we have two variables whose values are x = 16, y = 4.

Operator	Description of the operator	Example
Addition (+)	This operator will be adding the values on both sides of operands.	x + y = 20
Subtraction (-)	This operator will be subtracting the right-hand side value from the left-hand side value of the operand.	x – y = 12
Multiplication (*)	This operator will be multiplying the two values on both sides of the operands.	x * y = 64
Division (/)	This operator will be dividing the left-hand side value by the right-hand side value of the operand.	x / y = 4
Modulus (%)	This operator will be dividing the left-hand side value by the right-hand side value of the operand and returns the remainder.	x % y = 0
Exponent (**)	This operator will be doing the 'exponential power' calculation on operands.	x ** y = 16 to the power 4
Floor division (//)	This operator will be dividing the operands,	13 // 3 = 4, simultaneously 13.0 // 3.0

	the quotient of a number which is divided by 2 is the result.	= 4.0;

Example

Let's see how the output comes. {values in [ ] are outputs}
Let x, y and z be three variables with the following values:
x = 25, y = 30, z = 0:

```
#!/usr/bin/Python3

z = x + y

print(" result of z is ", z)

z = x – y

print(" result of z is ", z)

z = x * y

print(" result of z is ", z)

z = x / y

print(" result of z is ", z)

z = x % y

print(" result of z is ", z)
```

Output:

```
[z = 35]

[z = -5]

[z = 750]

[z = 0.833]

[z = 5]
```

Now suppose

```
a = 4, b = 5, c = a**b;

print("value of c is", c)

a = 15, b = 45, c = b//a;
```

```
print("value of c is", c)
```

Outputs:

```
[value of c is 1024]
```

As 4 to the power 5 is 1024.

```
[value of c is 3]
```

As the quotient of 45/15 is 3.

Python Comparison Operators

In Python, comparison operators are operators that compare two operands' values and returns true or false in case of whether the condition has matched or not. It is also called Python Relational Operator.
Let's take two variables having the values a = 20, b = 15:

Operator	Description of the operator	Example
(==)	This condition becomes true only if two given values (operands) are equal.	(a == b) → not true
(!=)	This condition becomes true only if the two operands aren't equal.	(a != b) → true
(>)	This condition becomes true only if the left operand is greater than the right operand.	(a > b) → true
(<)	This condition becomes true only if the right operand is greater than the left operand.	(a < b) → not true
(>=)	This condition becomes true only if the left operand is greater than or equal to the right operator.	(a >= b) → true
(<=)	This condition becomes true only if the right operand is greater than or equal to the left operand.	(a <= b) → not true

Example

Let's see what the output of the following code is:

```
#!/usr/bin/Python3
i = 10
j = 15
if ( i ==j )
    print("i is equal to j")
else
    print("i is not equal to j")
if ( i != i)
```

```
    print("i is not equal to j")
else
    print("i is equal to j")
if ( i > j)
    print("i is greater than j")
else
    print("i is not greater than j")
if ( i < j)
    print("i is less than j")
else
    print("i is not less than j")
if ( i >= j)
    print("i is greater than or equal to j")
else
    print("i is neither greater than nor equal to j")
if ( i <= j)
    print ("i is less than or equal to j")
else
    print("i is neither less than nor equal to j")
```

Outputs of the recently used comparison operators:

```
i is not equal to j
i is not equal to j
i is not greater than j
i is less than j
i is neither greater than nor equal to j
i is less than or equal to j
```

Python Assignment Operators

These kinds of operators are used to assign several values to the variables. Let's check the different types of assignment operators.

Operator	Description of the operator	Example
Equal (=)	This operator will assign values from	c = a + b;

	right side operand to left side operand.	
Add AND (+=)	This operator will add the right operand with left operand and assigns the sum to the left operand.	c += a → it is equivalent to c = c + a;
Subtract AND (-=)	This operator will subtract the right operand from the left operand and assigns the subtraction to the left operand.	c -= a → it is equivalent to c = c - a;
Multiply AND (*=)	This operator will multiply the right and left operand and assigns the multiplication to the left operand.	c *= a → it is equivalent to c = c * a;
Divide AND (/=)	This operator will divide the left operand with the right operand and assigns division to the left operand.	c /= a → it's equivalent to c = c/a;
Modulus AND (%=)	This operator takes modulus by using both sides' operand and assigns the outcome to left operand.	c %= a → it's equivalent to c = c % a;
Exponent AND (**=)	Does 'to the power' calculation and assigns the outcome to the left operand.	c **= a → it's equivalent to c = c**a
Floor division AND (//=)	It does floor division and assigns the outcome to the left operand.	c //= a → it's equivalent to c = c // a;

Let's see an example:

```
#!/usr/bin/Python3

a = 15

b = 20

c = 0


c = a + b

print("value of c is", c)


c += a

print("value of c is", c)


c *= a

print("value of c is", c)


c %= a

print("value of c is", c)
```

Output: 35, 50, 525, 5 are the outputs of the operators respectively.

Python Bitwise Operators

Bitwise operators are used to perform bit operations. All the decimal values will be converted in the binary format here.
Let's suppose:
a = 0101 1010
b = 0001 1000

Then, it will be
(a & b) = 0001 1000
(a | b) = 0101 1010
(a ^ b) = 0100 0010
(~a) = 1010 0101

Note: There is an in-built function [bin ()] in Python that can obtain the binary representation of an integer number.

Types of Bitwise Operators: [a = 0001 1000, b = 0101 1010]

Operators	Description of the operator	Example
Binary AND (&)	This operator executes a bit if it exists in both operands.	(a & b) is 0001 1000
Binary OR (|)	This operator executes a bit if it exists in one of the operands.	(a | b) is 0101 1010
Binary XOR (^)	This operator executes a bit if it is fixed in one operand but not in both	(a ^ b) is 0100 0010
Binary one's complement (~)	This operator executes just by flipping the bits.	~a = 1110 ~b = 0110
Binary left shift (<<)	This operator executes by moving left operand's value more left. It's specified by the right operand.	a << 100 (means 0110 0000)
Binary right shift (>>)	This operator executes by moving left operand's value right. It's specified by the right operand.	a >> 134 (means 0000 0110)

Let's see an example:

```
#!/usr/bin/Python3

a = 50                  # 50 = 0011 0010

b = 17                  # 17 = 0001 0001

print('a=', a, ':', bin(a), 'b=', b, ':', bin(b))
```

```
c = 0
c = a & b;              # 16 = 0001 0000
print("result of AND is", c, ':', bin(c))

c = a | b;          # 51 = 0011 0011
print("result of 0R is", c, ':', bin(c))

c = a ^ b;          # 66 = 0100 0010
print("result of XOR is", c, ':', bin(c))
c = a>> 2;          # 96 = 0110 0000
print("result of right shift is", c, ':', bin(c))
```

Output:
Result of AND is 16 → 0b010000
Result of OR is 51 → 0b110011
Result of XOR is 66 → 0b01000010
Result of right shift is 96 → 0b01100000

Python Logical Operator

The logical operator permits a program to make decisions according to multiple conditions. Every operand is assumed as a condition that can give us a true or false value. There are 3 types of logical operators.

(a = false operand, b = true operand)

Operators	Description of the operator	Example
Logical And(AND)	If the given operands both are true, the condition becomes true	Condition is false.
Logical or(OR)	If one of the given operands is true, the condition becomes true.	Condition is true.
Logical not(NOT)	If the given operand is true, the condition becomes false.	Condition is true (for a) and false(for b).

Let’s see an example:

```
>>> i = 25
# Logical AND Example
>>> if i < 30 AND i > 18:
print (" Condition is fulfilled ")
else:
print(" Condition is not fulfilled ")

# Logical OR Example
>>> if i < 18 OR i > 20:
print(" Condition is fulfilled ")
else:
print(" Condition is not fulfilled ")
```

Output:

```
Condition is fulfilled
Condition is fulfilled
```

Python Membership Operator

Membership operators are operators that validate the membership of a value. It examines for membership in a sequence like strings, lists, tuples, etc. Two types of membership operators are:

Operator	Description of Operator	Example
in	The condition becomes true if it can find a variable in a specified sequence.	Follow the example part given below.
not in	The condition becomes true if it can find no variable in a specified sequence.	Follow the example part given below.

Example

```
#!/usr/bin/Python3
i = 40, j = 20;
listValues = {10, 20, 30, 40}
if( i is in the listValues )
```

```
print(“ i is available in the list ”)
else
print(“ i is not available in the list ”)
if(j is in the listValues)
print(“ j is available in the list ”)
else
print(“ j is not available in the list ”)
k = i / j
if( k is in the listValues)
print (“ k is available in the list ”)
else
print(“ k is not available in the list ”)
```

Output:

```
i is no available in the list.
j is not available in the list.
k is available in the list.
```

Python Identity Operators

These are operators that are used to determine whether a value is of a particular class or type. To determine the type of data that contains several variables, this type of operator is used. There are two types of Identity operators as shown below:

Operator	Description of the operator	Example
is	The condition becomes true if the variables of each side of the operator are pointing to the same object.	If id(x) and id(y) are equal and x is y, the result is in 1.
is not	The condition becomes true if the variables of each side of the operator do not point to the same object.	If id(x) and id(y) are not equal and x is not y, the result is not in 1.

Example

```
#!/usr/bin/example3
```

```
x = 10, y = 10
print('x = ', 'x', ':', id(x), 'y = ', 'y', ':', id(y) )
if (x is y)
print(" Both x and y are having same identity ")
else
print(" x and y are not having same identity ")
if( id(x) == id(y) )
print(" Both x and y are having same identity ")
else
print(" x and y are not having same identity ")
```

Output:

```
x = 10 : 2371593036  y = 10 : 2371593036
Both x and y are having same identity
Both x and y are having same identity
```

Python Operator Precedence

In the below table all the operators from higher to lower precedence are listed

Operator	Description
**	Exponentiation(raise to the power)
~ + -	First one is Complement, second is unary plus and last one is unary minus.
/ * % //	Division, multiplication, modulus, floor division
+ -	Addition and subtraction
>> <<	Right bitwise shift and left bitwise shift
&	Bitwise AND
^ |	Bitwise exclusive OR and bitwise regular OR
<= < > >=	Less than equals to, less than, greater than, greater than equals to (comparison operators)
== <> !=	Equality operators
= %= /= //= - = += *= **=	Assignment Operators
is is not	Identity Operators
In not in	Membership Operators

NOT OR AND	Logical Operators

Example

For example, x = 5 + 14 * 2; in this equation, the value of x is 33, not 38 because the operator * has higher precedence than +. For which it first multiplies 14 * 2 and then add it with 5.

Chapter 5 Data Structures

In this chapter, we are going to explore the different data structures in Python.

Sequence

A sequence is a very basic term in Python that is used to denote the ordered set of values. There are many sequence data types in Python: str, unicode, list, tuple, buffer and xrange.

Tuples

A tuple consists of some values separated by commas. Tuples are also a sequence data type in Python, like strings and lists. We need to keep in mind that tuples are immutable. It means that they can't be changed.

The tuples consist of the number of values separated by a comma. The tuples are enclosed in parentheses, while the lists are enclosed in brackets.

Now let's see an example:

```
>>> m = (14, 34, 56)

>>> m

(14, 34, 56)

>>> m[0]

14

>>> m[ 0:2 ]

(14, 34)
```

Tuples also have the properties like indexing and slicing. Tuples can be nested. Elements in a tuple can be grouped with ()

Now let's see an example:

```
i = 1
j = 2
t1 = i, j       # is a tuple consists to elements i and j
t2 = (3, 4, 5)       # is a tuple consists to elements 3,4 and 5
t3 = 0, t1, t2           # is a tuple consists to elements 0, t1 and t2
print t3       # result is (0, (1, 2), (3, 4, 5))
```

Lists

A list consists of some heterogeneous values separated by commas enclosed by [and] and started from index 0. Lists can be used to group other values. Unlike Tuples, Lists are mutable. In other words, they can be changed by removing or reassigning existing values. Also, new elements can be inserted into the existing ones.

Now let's see an example:

```
>>> a = [1, 2, 3, 4, 5]
>>> a
[1, 2, 3, 4, 5]
```

As strings, lists can also be indexed and sliced.

```
>>> a = [1, 2, 3, 4, 5]
>>> a
[1, 2, 3, 4, 5]
>>> a[0]
1
>>> a[4]
5
>>> a[ 0:2 ]
```

```
[1, 2]
>>> a[ 3:5 ]
[4, 5]
```

Unlike strings, *lists are mutable* (i.e. the values can be changed)

```
>>> b = [1, 2, 4, 7, 9]
>>> b
[1, 2, 4, 7, 9]
>>> b[2] = 6
>>> b
[1, 2, 6, 7, 9]  # Here the index [2] is changed to 6 (the initial value is 4)
>>> b[0] = 9
>>> b
[9, 2, 6, 7, 9]
# Here the index [0] is changed to 9 (the initial value is 1)
```

The values in the list can be separated by using comma (,) between the square bracket. Lists can be nested. List can be used as a Stack or a Queue.

For example:

```
list1 = [ 1, 2, 3, 4]
print len (list1) # returns 4 - which is the length of the list
list1[2] # returns 3 - which is third element in the list Starts
list1[-1] # returns 4 - which is extreme last element in the list
list1[-2] # returns 3 - which is extreme last but one element
list1[ 0:2 ] = [ 11, 22] # replacing first two elements 1 and 2 with 11 and 22
```

```
stackList = [ 1, 2, 3, 4]
stackList.append(5) # inserting 5 from the last in the stack
print stackList # result is: [1, 2, 3, 4, 5]
stackList.pop() #  removing 5 from the stack  Last In First Out
print stackList # result is: [1, 2, 3, 4]
queueList = [ 1, 2, 3, 4]
queueList.append(5) # inserting 5 from the last in the queue
print queueList # result is: [1, 2, 3, 4, 5]
del(queueList[0] ) #  removing 1 from the queue  First In First Out
print queueList # result is: [2, 3, 4, 5]
```

Sets

A set doesn't have any duplicate elements present in it and it is an unordered collection type. It means it will have all distinct elements in it with no repetition.

Now let's seen an example:

```
fruits = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
basket = set (fruits) # removed the duplicate element apple
print 'orange' in basket # checking orange in basket, result is True
print 'pineapple' in basket # checking pine apple in basket, result is False
a = set('aioeueoiaeaeiou') # create a set without duplicates
b = set('bcokcbzo') # create a set without duplicates
print a # a = ['a', 'i', 'e', 'u', 'o']
print b #   b = ['z', 'c', 'b', 'k', 'o']
print a & b # letters in both a and b  ( A ∩ B )
```

```
print a | b # letters in either a or b  ( A ☐ B )
print a - b # letters in a but not in b ( A – B )
```

Dictionaries

Dictionaries are the data structures in Python that are indexed by keys.

Key and values separated by “:” and pairs of keys separated by a comma and enclosed by { and }

Lists cannot be used as keys.

Now let’s see an example:

```
capitals = { 'AP' : 'Hyderabad', 'MH' : 'Mumbai' }
capitals[ 'TN' ] = 'Chennai'
print capitals[ 'AP' ]# returns value of AP in the dictionary
del capitals[ 'TN' ] # deletes TN from the dictionary
capitals[ 'UP' ] = 'Luck now'  # adding UP to the dictionary
print 'AP' in capitals    # checks where AP key exist in dictionary
print 'TN' in capitals
Numbers  = {'1': 'One', '2': 'Two'}
for key, value in Numbers.iteritems() :
    print key, value
```

Strings

In Python, a string is identified by the characters in quotes, such as single (‘’) and double (””). They can only store character values and are primitive datatype. Please note that strings are altogether different from integers or numbers. Therefore, if you declare a string “111”, then it has no relation with the number 111.

```
>>> print "hello"
```

```
hello
>>> print 'good'
good
```

The string index starts from 0 in Python.

```
>>> word = 'hello'
>>> word[0]
'h'
>>> word[2]
'l'
```

Indices may also be negative numbers, to start counting from the right. Please note that *negative indices* start from -1 while *positive indices* start from 0 (since -0 is same as 0).

```
>>> word = 'good'
>>> word[-1]
'd'
>>> word[-2]
'o'
```

The slicing in Python is used to obtain substrings, while index allows us to obtain a single character.

```
>>> word = 'develop'
>>> word[ 0:2 ]
'de'
>>> word[ 2:4 ]
```

```
've'
```

Please note that the *starting* position is always included and the *ending* position is always excluded.

D e v e l o p

0 1 2 3 4 5 6 ---- Index value

In the above example, the word is assigned a value develop. Considering the first statement word [0:2], the output is ‘de’. Here the starting position ‘d’ (0th index) is included and the ending position ‘v’ (2nd index) is excluded. Similarly, in the second statement word [2:4], the starting position ‘v’ (2nd index) is included and the ending position ‘l’ (4th index) is excluded.

The important point to be noted is that Python *strings are immutable* (i.e. Strings cannot be changed).

There are many in-built functions available with a String. They are used for various purposes. Let’s see some of the basic ones that are most commonly used.

- Len: It is the length function that is used to calculate the number of characters present in the string.
- Lower: It will convert all the uppercase characters present in the string to lowercase letters. Therefore, after using this function, all characters in the string will be small case only.
- Upper: It will convert all the lowercase characters present in the string to uppercase letters. Therefore, after using this function, all characters in the string will be upper case only.
- Split: It helps to split the string into parts by using a delimiter. It can be separated using spaces, new lines, commas, or

tabs.

Control Flow Statements

If–else statement

The if-else statement is used to make the choices from 2 or more statements. It becomes helpful when you want to execute a particular statement based on a True or False condition.

The syntax of if statement is:

If condition:

    action-1   # Indentation

Else:

  action-2   # Indentation

Here the *indentation* is required. The actions action-1 and action-2 may consist of many statements but they must be all indented.

if <expression> :

<statements>

else :

<statements>

The example is shown below.

```
>>> e = 6
>>> f = 7
>>> if(e < f):
...     print( 'f is greater than e' )
... else:
...     print(' e is greater than f')
...
```

Output: f is greater than e

```
def  numberProperty1 ( input ) :
        if input % 2 ==  0 :
print input , ' is an Even number '
        else :
print input , ' is an Odd number '
numberProperty1( 10 )    # result is 10 is an Even number
numberProperty1( 11 )    # result is 11 is an Odd number
```

Nested If

It consists of more than 2 statements to choose from.

```
def numberProperty2 ( input ) :
if input < 0:
print input , ' is a Negative number '
elif input == 0:
print input , ' is Zero '
else:
print input , ' is a Positive number '
numberProperty2( -100 )  # -100  is a Negative number
numberProperty2( 0 )       # 0  is Zero
numberProperty2( 100 )   # 100  is a Positive number
```

While Loop

The while loop will run until the expression is true and it stops once it is false.

The syntax of while loop is:

```
While expression:
   statement
```

For example:

```
>>> a = 1
>>> while(a < 10 ):
...    print "The number is:" , a
...    a = a + 1
...
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

In the above example, the block consists of print and increment statements, it is executed repeatedly until the count is no longer less than 5.

def printSeries(start, end, interval) :

```
print " \n "
temp = start
```

```
while ( temp < end ) :
print temp,
temp += interval
printSeries( 1, 11, 1 )   # result is  1 2 3 4 5 6 7 8 9 10
printSeries( 1, 11, 3 )   # result is  1 4 7 10
```

For Statement

Any object with an iteration method can be used in a for loop in Python. The iteration method means that the data can be presented in a list form where there are multiple values in an ordered manner. The syntax of for loop is:

```
for item in list:
    action              # Indentation
```

The action consists of one or more statements and it must be *indented* . The examples are shown below.

For example:

```
>>> for i in (1, 2, 3, 4, 5, 6, 7, 8, 9, 10):
...     print i
...
1
2
3
4
5
6
```

```
7

8

9

10

>>> list = ['a', 'bb', 'ccc', 'dddd']

>>> for l in list:

...     print l,len(l)

...

a 1

bb 2

ccc 3

dddd 4
```

The above program shows that the values of a list and its length are printed using the for loop.

Functions

A function is a block of organized and reusable code that is used to perform related tasks. We can break our huge lines of programming code into smaller modules with the help of functions. It also helps in avoiding repetitions of code, as we don't need to write the same lines of code again and again. Instead, we can write it once inside a function and then use the function anywhere in the program.

You need to make sure that the function name is unique.

Rules to define a function in Python

In Python, a function is defined using the keyword def. The arguments will be placed within the parenthesis ().

Now let’s see an example:

```
>>> def printdetails(name, age):
...     print "Name:", name;
...     print "Age:", age;
...     return;
...
>>> printdetails(name = "Mary", age = 30);
Name: Mary
Age: 30
```

In the above example ‘printdetails ‘is the function name and name and age are the parameters.

Syntax of user defined method

```
def   < function name> :
[ < declaration of local variables > ]
[ < statements > ]
```

Now let’s see an example:

```
Language = "Python"
def printString( input ) :
print input
def multiply ( x, y ) :
return x * y
def power( x, y):
return x ** y
printString( Language )       # returns Python
z = multiply( 10, 20 )
```

```
print z# returns 200 - which is equal to 10 * 20
print power( 10, 2 )   # returns 100 - which is equal to 10 ** 2
```

Accepting inputs during the runtime

raw_input() is a built-in Python function provides the facility to accept input during the execution of the script

Now let's see an example:

```
name = raw_input( "\n Please enter your name : " )
```

This statement provides a message to the user to provide input for a name.

Control Statements

Break

The break statement breaks out of the smallest enclosing for or while loop.

Now let's see an example:

```
def primeNumberValidation ( input ) :
    for x in range( 2, input ) :
        if  input % x == 0:
    print  input, 'is not a prime number and equals', x, '*',  input/x
    break
else:
    print  input, 'is a prime number'
primeNumberValidation( 3 )
primeNumberValidation( 14 )
```

Continue

The continue statement continues with the next iteration of the loop.

Now let’s see an example:

```
def evenNumbers( start, end ) :
print "\n\nEven numbers in between ", start , " and ", end
for n in range( start + 1, end ) :
if n % 2 !=  0:
continue
print n
evenNumbers( 1, 11 )   # result is  14 is 2 4 6 8 10
evenNumbers( 10, 30 ) # result is  12 14 16 18 20 22 24 26 28
```

Pass

The pass is a valid statement and can be used when there is a statement required syntactically, but the program requires no action.

Now let’s see an example:

```
while True :
        pass   # In condition loop press (Ctrl + c) for the keyboard interrupt
```

In this example, while followed by “pass” it does not execute any statement.

There is a necessity to include at least one statement in a block (e.g. function, while, for loop, etc.) in these cases, use pass as one statement, which does nothing but includes one statement under ‘:’

Now let’s see an example:

```
def  x() :
pass  # one valid statement that does not do any action
```

Here pass is considered a statement for the declaration of function x.

String Manipulation

We can use built-in functions to manipulate strings in Python. The package “string” provides more functions on strings.

For example:

```
print name = "ABCD XYZ xyz"

print len(name)        # It will return the length of the string name

print list(name)       # It will return the list of characters in name print

name.startswith( 'A' )          # It will return True if name starts with A else returns False

print name.endswith( 'Z' )    # It will return True if name ends with Z else returns False

print name.index( 'CD' )      # It will return the index of CD in name

print 'C'.isalpha( )                 # It will return True if C is alpha or returns False

print '1'.isdigit( )              # It will return True if 1 is digit or returns False

print name.lower( )         # It will return a string with lowercase characters in name

print name.upper( )         # It will return a string with uppercase characters in name
```

Exception Handling

Exceptions are the errors detected during execution and these are not unconditionally fatal.

Exception blocks will be enclosed with try and except statements.

try :

<statements>

except <exception type > :

<statements>

Let’s see an example:

```
# Defining an exception block
try:
        print ( 1 / 0 )
except Exception as excep:
        print "exception : ", excep
# Defining a user-defined exceptio n
class UserDefinedException( Exception ) :
        def __init__(self, value):
        self.value = value
        def __str__(self):
        return repr(self.value)
# Raising a user-defined exception explicitly
try:
        raise UserDefinedException(" input is null ")
except UserDefinedException as userdefinedexception:
        print 'userdefinedexception : ',  userdefinedexception.value
```

In the above-mentioned program, first (try, except, block) handles the Zero division exception.

UserDefinedException is a user-defined exception to raise business exceptions in the program.

Second (try, except) block raises a user-defined exception.

Chapter 6 Learning about Functions

So far, we have learned quite a lot of things. If you have already started to lose track of all the knowledge, you shouldn't be alarmed. It is only natural for everyone to find themselves in such a situation when they are in the learning process. No one is perfect and that is what makes us all human beings, right?

We have seen dictionaries and learned they are nothing like the ones we use to learn new words and meanings. We have learned about a rather funny thing called tuples and understood that they are essentially a list with parentheses and do not allow anyone to add, remove, or modify values. We have gone initially through some functions too, but now it is time for us to start looking into functions a little more closely.

Understanding the Concept of Function

Take a moment or two here and engage your mind a little. Think about it and try to come up with some vague idea of what functions truly are.

Functions are either user-defined or pre-defined. In either case, their job is to organize codes within a recallable function name. There are tons of pre-defined functions available within Python. We have already been using some of these again and again.

We already have a decent idea about functions that are built-in and pre-defined. These include and are not limited to input(), print(), and so many more. However, let's now look at how to create our function.

Let's begin by a traditional approach and write a block of code that welcomes the user with a friendly greeting. We will store this as a function named "welcome_message" so that we can call on this function later on.

```
def welcome_message():
    print("Hello and welcome")
    print("Hope you have a great time")
```

```
print("Begin")

welcome_message()

print("End")
```

Let’s begin learning and see what is happening in the block of code above. Firstly, for us to create our function, we need to define it first. The ‘def’ is a keyword that Python will look at and immediately understand that you are about to ‘define’ a new function. Next, we will need to name the function. While you can always name the function as you please, it is highly recommended and encouraged that you use names that are easy to understand and have a descriptive name. If we were to name this function anything other than welcome_message, we may know what it is as we wrote it, but for any other programmer out there, they may not understand.

Whenever you create a function, you need to use parentheses. You do not have to pass any information through it so leave them as they are. Now, we need to add the colon mark.

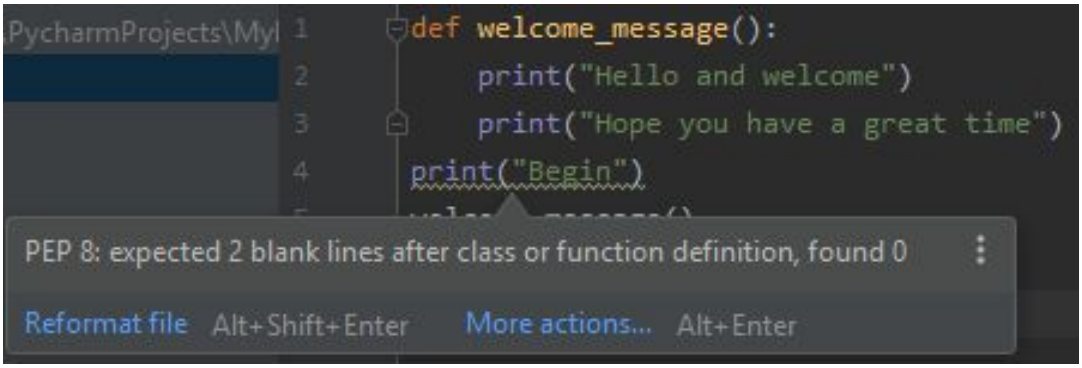
What happens when you use a colon at the end of a statement? Your cursor gets indented in the following line. That means your cursor will be slightly far from the actual starting point. This is to denote to the programmer that he/she is about to type something that would hold value for a command or a statement above it. In this case, we are trying to define the function.

Let’s then use two print commands and place our greeting messages. That is it! You have now created your very first function. You can now recall it as many times as you like. However, should you try to call on this function a line or two before the ‘def’ command, Python will have no idea what you’re talking about. Why? That has everything to do with the fact that Python reads a program line by line. By the time it arrives on the line where you called a function, it would check with the previous lines and not find anything relatable as the actual ‘def’ step was carried out in a step following this.

After this, let’s now use our function and see how it works. Remember, the function holds two printable messages for our users. For our reference, we will now create a ‘begin’ and an ‘end’ message. This would allow us and the programmer to know where the regular messages are and where the function lies. Use your function with empty parentheses between the two

print commands as shown above. If you like, you can remove these print commands and just type in your function number to see the results.

A quick tip for all! If you come across the annoying wiggly lines, simply hover your mouse over it and you will find out what the expected or suggested solution is. In this case, if you remove the two-line spaces, you should see a suggestion saying this:



Whenever you define a function, you will always be required to leave at least two blank lines before proceeding on with the codes.

Now, let's run the program and you should see all the messages and our function in action. Python initiated the sequence and first read the definition. This is where Python only understood for itself what the function was. The actual program was executed when Python reached line six, where our print("Begin") message started. In the next line, we placed our function and this is where Python recalled what it had just learned. It quickly carried out the set of codes we defined within the function and executed the same. Lastly, it executed the last line before finishing the program.

This is how functions are created and used. Now, we can use this function as many times as we like within the same file. Note that you cannot use this newly created function if you were to open a new file or work on an older file where you did not define this function.

When things start to get tougher for you in your programming future, remember to create your functions and use them where applicable. They will save you quite a lot of time and help you in places as well. These are used when certain actions or operations need to be carried out every now and then.

Using Various Functions

Python was created with simplicity in mind. It was also created to minimize the work and maximize the output. If you use the codes and the functions wisely, you will surely be making the most out of this programming language. It is also noticeable that most of the things you learn about Python and its functions, parameters, methods, and such will help you learn other languages quicker, so do pay attention.

Parameters

Our eagle-eyed readers may have noticed something about the function we just created a few minutes ago. Unlike most of the functions, we did not pass any information through the parentheses at all. Why that happens is something we will come to know about once we understand exactly what parameters are in Python.

Parameters are used as place-holders for receiving information. These are what we, as well as users, provide to the program in order for it to work more accurately. There are some cases and functions where arguments are not required for them to do their basic operation. However, if you provide an argument to these functions, they will provide you with a more specific output. Of course, it does depend on the availability of the said parameter. You cannot force a function to do something it is not designed to do.

Now, let’s look at our function. It is certainly missing something. If you currently print the welcome_user function, it would say everything but will not contain the name of the user at all. Surely, it would look a lot nicer for us if we could somehow use this function to use the name of the user and add it to the greeting.

Luckily, we can do just that! For that, we first need to add the ‘name’ parameter in the first line, where we began defining our function. Simple type name between the parentheses and you will see the text turn grey. This confirms that the word has been added as a parameter. Now, we wish to print the name of this user along with the greetings we have defined within the function. For this example, let’s assume that the user is named Fred.

```
def welcome_message(name):
        print("Begin")
        print("Hello and welcome {name}!")
```

```
        print("Hope you have a great time")
        print("End")

welcome_message('Fred')
Begin
Hello and welcome Fred!
Hope you have a great time
End
```

Finally! We have a name to add to our greetings. You can add another line of code by using our function and passing a different name now. See what happens then.

When we set a parameter for a function and then call upon the function without providing it with an argument or the bit of information that goes between the parentheses, it will provide us with an error, except for a few.

Now, let's make our function a little more dynamic and add another parameter. Let's add a parameter that allows the program to print out the last name of the user. Now, our code should look something like this:

```
def welcome_message(name, last_name):
    print("Hello and welcome {name} {last_name}!")
    print("Hope you have a great time")

print("Begin")
welcome_message('Fred', 'William')
print("End")
```

The point to learn here, apart from being able to add parameters, is the fact that 'Fred' and 'William' are being used in a specific order. Should you type it the other way around, Python will print these as they are. This is because of their position concerning the defined parameters. The first value

Python reads here, it will automatically link it with the first parameter. This can cause a little confusion, especially if the last name becomes the first name.

These arguments are called as positional arguments. To further show their importance, let’s remove the last name from the argument above.

```
print("Begin")

welcome_message('Fred')

print("End")

Traceback (most recent call last):

Begin

  File "C:/Users/Smith/PycharmProjects/MyFirstGo/PosArg.py", line 7, in <module>

    welcome_message('Fred')

TypeError: welcome_message() missing 1 required positional argument: 'last_name'
```

So, the system does not allow us to continue as we have removed an element. This time, type in the last name first followed by the first name and see if it makes any difference. When you run the program now, you should be able to see this:

```
print("Begin")

welcome_message('Fred', 'William')

print("End")

Begin

Hello and welcome William Fred!

Hope you have a great time

End
```

Now, the sequence kind of worked. The only issue is that it has gotten the name wrong. Now, the last name is being portrayed and printed as the first name. That is rather embarrassing, isn’t it?

The above errors either state that we are missing one required positional argument or show that we placed the wrong name in the wrong place. Positional arguments are such arguments whose position matters a lot. If you miss out on the position altogether, you will end up with an error. If you type in something else, as we did in our last example, you will produce incorrect results. To correct it, simply provide the last name after the first name.

There is one more way you can have these dealt with by using what is termed as ‘keyword arguments’. These are the kind of arguments whose position does not matter at all and Python will still continue to function properly regardless of their position in the parentheses. To pass a keyword argument, you will need to do the following:

```
print("Begin")

welcome_message(last_name='William', name='Fred')

print("End")

Begin

Hello and welcome Fred William!

Hope you have a great time

End
```

Now that’s more like it. Things are looking right at how we want them. Notice how, even though we wrote in the wrong order, Python picked up and sorted the order for us. That is because we made our entries or arguments into keyword arguments using the name= or last_name= parameter and combining it with arguments. This allows Python to draw information and understand which of these two comes first in order as defined originally in our function.

Factually speaking, you will not be using these quite a lot, but it is always an advantage to know the ways to overcome certain issues. Normally, you or any other programmer would be able to see the data and read easily if

you simply follow the rules and type first name followed by last name. Make the code as easy as you can for everyone to read and understand.

“Well, what if I was using numbers instead of names?”

That is one fine question. This is where you will need to use keyword arguments to represent what those values are for. You might be running a function that involves multiple values which only you might be able to understand, but others will have no idea where they came from. You must label each one of them with the relevant keyword argument so that the readability increases.

We are currently just beginning and for the sake of demonstration, we used a simple example to showcase how to create functions and use them. Your functions, when the time comes, might be quite vast or equally short, depending on the kind of function you create of any specific situation.

Creating functions certainly helps us organize our codes and be more efficient and effective. If we were unable to do this, we would have had to resort to writing the same bunch of lines every now and then.

Return Statement

We could have covered what a return statement is when we were discussing ‘if’ statements and others. However, it makes more sense to learn this after you have understood the concept of parameters and functions.

So far, we have created a function that has allowed us to send information via the use of parameters. However, when we talk about return statements, these are designed to do certain calculations and provide us with the results instead of us feeding it with values.

Let’s look a little deeper into how this works by creating our second function. The purpose of this function is based on a simple math trick which most of us might have heard of or played with when we were young. Ask the user to think of any number and you would ask them to add and subtract a few simple numbers. Eventually, you would provide them with an accurate result and everyone would be shocked. Now, let’s reveal what happens with the use of this function we are about to create.

```
def magic_number(number):

    return number + 6 - 4 + 5 - number
```

This simple calculation would always return you the value of seven. Go ahead, try it out yourself by doing this. However, for us to be able to get these values back, we have used the return statement here. This tells Python that it is supposed to do the calculation for us and then only return the resulting value instead of printing each of these individually.

Let's give our second function a test run:

```
result = magic_number(8329)

print(result)
```

See how the result now shows as seven? You can try and change the values to whatever you please, the result will continue to remain as seven. That is only made possible owing to the return statement we have provided here. If you take away the keyword return, you end up with a value that says 'None'. The program will still function, but the result would no longer be calculated or of any use to us. This is because Python would not execute a return phase and thus will not carry out the calculations as we would like it to.

Using these can greatly enhance your experience as a programmer or a user. However, before you dive in and start creating your functions, here are some which are pre-defined and may come in handy. There is no point in creating a function and finding out Python already had one for you.

1. *min() and max()* - In case you run into various values and you quickly wish to find out the minimum value in existence within a list or collection of data, use the min() command and run the numbers through. The minimum number will be printed for you. The max() function is the opposite, of course!
2. *sum()* - This is quite a nifty function and allows you to quickly add up all the numbers in the list and produce the result for you right away. The accuracy with floats might not be what we like, but hey, it gets you going.
3.

type() - There may come lines and lines of codes with variables that are scattered all over the place. You now wish to find out where the variable started from and what kind of a variable it is. Using the type() function, you can quickly find out what kind of variable you are dealing with. It will return values such as 'bool' to indicate that the variable in question is a bool in type.

There are hundreds of functions that you will start to learn as you proceed into advanced Python learning and machine learning. However, to understand most of them, you will need to practice these and develop a thorough understanding of how this works. You should then have no problems venturing into a more advanced version of Python learning and developing complex programs.

Test your knowledge

This time, let's raise the stakes a little and test your knowledge. This will involve a mix of all that you have learned thus far. However, this time you will not be provided with solutions to keep you on a quest to search for answers and use a bit of trial and error method to perfect the program.

Exercise

Here is the updated version of a program we designed to check insurance prices a person would have to pay if he/she was above or below a certain age. Your objective is to convert this into a function. Your function should have three fields set to receive input from the user.

1. Name
2. Age
3. *Actual insurance cost*

Updated code:

```
Insurance = 1000
```

```
age = int(input('Your age: '))
is_old = age > 40
is_young = age <= 28
has_license = input('Do you have a license? ')
if has_license.lower() == 'Yes':
    has_license = True
elif has_license.lower() != 'Yes':
    has_license = False
if is_old and has_license:
    Insurance = Insurance / 2
    print("Your insurance cost is ${Insurance}")
elif is_young and has_license:
    Insurance = Insurance // 1.50
    print("You will need to pay ${Insurance}")
else:
    print('You are not eligible for insurance at this time')
```

Chapter 7 Conditional and Loops in Python

This chapter describes moderate level topics like conditionals and loops in detail. We will use different examples to explain these topics in detail. Let's dive into knowing more about these concepts.

What is a sequence in Python?

The sequence of program execution is not a highway linking the north and the south. It can run from the north to the south to the end. The sequence of program execution may be as complicated as a highway in the busy area, with nine turns and 18 turns, which is easy to make people dizzy.

To write a good program, it is very important to control the process of program execution. Therefore, it is necessary to use the process control structure of the program. Without them, it is impossible to use the program to complete any complicated work.

The programming language has been continuously developed for decades. Structured Programming has gradually become the mainstream of program development. Its main idea is to execute the entire program in sequence from top to bottom. Python language is mainly executed from top to bottom according to the sequence of program source code, but sometimes the execution sequence will be changed according to needs.

At this time, the computer can be told which sequence to execute the program preferentially through flow control instructions. The process control of the program is like designing a traffic direction extending in all directions for the highway system.

It is recognized that most program codes for process control are executed in sequence from top to bottom line after line, but for operations with high repeatability, it is not suitable to execute in sequence. Any Python program, no matter how complex its structure is, can be expressed or described using three basic control processes: sequence structure, selection structure, and loop structure.

The first line statement of the sequence structure program is the entry point and is executed from top to bottom to the last line statement of the program. The selection structure allows the program to select the program block to be executed according to whether the test condition is established or not. If the condition is True, some program statements are executed. If the condition is False, other program statements are executed.

Colloquially, if you encounter a situation A, perform operation A; if this is case b, operation b is executed. Just like when we drive to the intersection and see the signal lamp, the red light will stop, and the green light will pass. Also, different destinations also have different directions, and you can choose the route according to different situations. In other words, the selection structure represents that the program will determine the "direction" of the program according to the specified conditions.

The function of loop flow control with loop structure is to repeatedly execute the program statements in a program block until the specific ending conditions are met. Python has a for loop and a while loop.

Selection Process Control

Selection Process Control is a conditional control statement that contains a conditional judgment expression (also referred to as conditional expression or conditional judgment expression for short). If the result of the conditional judgment expression is True (true), a program block is executed. If the result of the conditional judgment expression is false (True), another program block is executed.

The following describes the statements and their functions related to the selection process control in Python language.

If...Else Conditional Statement

If...else conditional statement is a fairly common and practical statement. If the conditional judgment expression is True (true, or represented by 1), the program statement in the if program block is executed. If the conditional judgment expression is not true (False, or represented by 0), the program statement in the else program block is executed. If there are multiple judgments, elif instruction can be added.

The syntax of the if conditional statement is as follows:

If the conditional judgment expression holds, execute the program statement in this program block

Else ：

If the condition does not hold, execute the program statement in this program block. If we want to judge whether the value of variable a is greater than or equal to the value of variable b, the condition judgment expression can be written as follows:

If a >= b:

If A is greater than or equal to B, execute the program statement in this program block

Else ：

If a is NOT greater than or equal to b, the program statement if ... if…else conditional statement in this program block is executed.

In the use of the if...else conditional statement, if the condition is not satisfied, there is no need to execute any program statement, and the else part can be omitted

If conditional judgment expression

If the condition is satisfied, execute the program statements in this program block. Besides, if the if…else conditional statement uses logical operators such as “and”, it is suggested to add parentheses to distinguish the execution order to improve the readability of the program,

For example: if (a==c) and (a>b):

If A equals C and A is greater than B, execute the program statement in this program block

Else ：

If the above condition does not hold, the program statement in this program block is executed.

Also, Python language provides a more concise conditional expression of if...else in the following format: X if C else Y returns one of the two expressions according to the conditional judgment expression. In the above expression, X is returned when C is true; otherwise, Y is returned.

For example, to determine whether the integer x is odd or even, the original program would be written as follows:

```
If (first % 2)==0:
second= "even number"
Else:
second= "odd number"
```

If print('{0}'.format(second)) is changed to a concise form, only a single line of program statements is required to achieve the same purpose.

The statements are as follows:

```
print('{0}'.format ("even" if (first% 2)==0 else "odd"))
```

If the if condition determines that the expression is true, it returns "even"; otherwise, it returns "odd." In the following sample program, we will practice the use of the if…else statement. The purpose of the sample program is to make a simple leap year judgment program.

Let the user enter the year (4-digit integer year), and the program will determine whether it is a leap year. One of the following two conditions is a leap year:

(1) leap every 4 years (divisible by 4) but not every 100 years (divisible by 100).
(2) leap every 400 years (divisible by 400).

[example procedure: leapYear.py]

Determine whether an input year is a leap year or not

```
01 # -*- coding: utf-8 -*-
02 """
03 program name: leap year judging program
04 Topic Requirements:
05 Enter the year (4-digit integer year) to determine whether it is a leap year
```

```
06 condition 1. Every 4 leap (divisible by 4) and every 100 leap (divisible by 100)

07 condition 2. Every 400 leap (divisible by 400)

08 One of the two conditions met is a leap year.

09 """

10 year = int(input ("Give year:"))

12 if (year % 4 == 0 and year % 100 ! = 0) or (year % 400 == 0):

13 print("{0} is a leap year .”format(year))

14 Else:
```

The execution results of the

```
15 print("{0} is the year of peace .”format(year))
```

Program Code Resolution:

Line 10: Enter a year, but remember to call the int () function to convert it to an integer type.

Line 12-15: Judge whether it is a leap year.

Condition 1: every 4 leaps (divisible by 4) and every 100 leaps (not divisible by 100).

Condition 2: every 400 leaps (divisible by 400). One of the two conditions is a leap year. Readers are asked to inquire whether the following years are leap years: 1900 (flat year), 1996 (leap year), 2004 (leap year), 2017 (flat year), 2400 (leap year).

Multiple Choices

If there is more than one conditional judgment expression, elif conditional statement can be added. Elif is like the abbreviation of "else if." Although using multiple if conditional statements can solve the problem of executing different program blocks under various conditions, it is still not simple enough. Then, elif conditional statements can be used, and the readability of the program can be improved.

Note that if the statement is a logical "necessity" in our program. Elif and else do not necessarily follow, so there are three situations: if, if/else, if/elif/else.

The format is as follows:

If condition judgment

Expression 1:

If the conditional judgment expression 1 holds, the program statement in this program block is executed

Elif condition judgment

Expression 2:

If the conditional judgment expression 2 holds, execute the program statement in this program block

Else ：

If none of the above conditions hold, execute the program statement in this program block,

For example:

If first==second:

If first equals second, execute the program statement in this program block

Elif first>second ：

If first is greater than second, execute the program statement in this program block

Else ：

if first is not equal to second and first is less than second, execute the program statement in this program block.

The following example program is used to practice the use of IF multiple selection. The purpose of the sample program is to detect the current time to decide which greeting to use.

[sample procedure: currentTime.py]

Detects the current time to decide which greeting .

```
01 # -*- coding: utf-8 -*-
02 """
03 Program Name: Detect the current time to decide which greeting to use
04 Topic Requirements:
05 Judging from the current time (24-hour system)
06 5~10:59, output "good morning"
07 11~17:59, output "good afternoon"
08 18~4:59, output "good night"
09 """
11 import time
13 print ("current time: {}.” format (time.strftime ("%h:% m:% s"))
14 h = int( time.strftime("%H") )
16 if h>5 and h < 11:
17 print ("good morning!" )
18 elif h >= 11 and h<18:
19 print ("good afternoon!" )
20 else:
21 print ("good night!")
```

The execution results of the program will be shown on the screen.

The output shows the current time in the sample program to judge whether it is morning, afternoon, or evening, and then displays the appropriate greeting. Python's time module provides various functions related to time. The Time module is a module in Python's standard module library.

Before using it, you need to use the import instruction to import and then call the strftime function to format the time into the format we want. For

example, the following program statement is used to obtain the current time.

```
import time

Time.strftime ("%h:% m:% s")

# 18: 36: 16 (6:36:16 p.m. 24-hour)

Time. strftime ("%i:% m:% s")

# 06:36:16 (6: 36: 16 p.m. 12-hour system) format parameters to be set are enclosed in parentheses.
```

Pay attention to the case of format symbols. The following program statement is used to display the week, month, day, hour, minute, and second.

Print (time.strftime ("%a,% b% d% h:% m:% s")) execution results are as follows: Monday, sep17 15: 49: 29 4.2.3 nested if sometimes there is another layer of if conditional statement in the if conditional statement. This multi-layer selection structure is called nested if conditional statement.

Usually, when demonstrating the use of nested if conditional statements, it is more common to demonstrate multiple choices with numerical ranges or scores. In other words, different grades of certificates will be issued for different grades of achievements.

If it is more than 60 points, the first certificate of competency will be given, if it is more than 70 points, the second certificate of competency will be given, if it is more than 80 points, the third certificate of competency will be given, if it is more than 90 points, the fourth certificate of competency will be given, if it is more than 100 points, the all-round professional certificate of competency will be given.

Based on nested if statements, we can write the following program:

```
Available= int(input ("Give a score:")

if available >= 60:

print ('First Certificate of Conformity')

if available >= 70:
```

```
print ('Second Certificate of Conformity')
if available >= 80:
print ('Third Certificate of Conformity')
if available >= 90:
print ('Fourth Certificate of Conformity')
if getScore == 100:
```

Print ('All-round Professional Qualification Certificate') is actually an if statement that is explored layer by layer. We can use the if/elif statement to filter the multiple choices one by one according to conditional expression operation and select the matching condition (True) to execute the program statement in a program block.

The syntax is as follows:

If Conditional Expression 1:

The program block to be executed under conditional expression 1

Elif conditional expression 2:

The program block to be executed under conditional expression 2

Elif conditional expression n:

The program block to be executed according to the conditional expression n

Else:

If all the conditional expressions do not conform, this program block is executed. When the conditional expression 1 does not conform, the program block searches down to the finally conforming conditional expression.

The elif instruction is an abbreviation of else if. Elif statement can generate multiple statements according to the operation of a conditional expression, and its conditional expression must be followed by a colon, which indicates that the following program blocks meet this conditional expression and need to be indented.

The following example program is a typical example of the combined use of nested if and if/elif statements. This program uses if to determine which

grade the query results belong to. Also, another judgment has been added to the sample program. If the score integer value entered is not between 0 and 100, a prompt message of "input error, the number entered must be between 0 and 100" will be output.

Comprehensive use of nested if statements example:

```
01 # -*- coding: utf-8 -*-

02 """

03 Examples of Comprehensive Use of Nested if Statements

04 """

05 result = int(input (‘Give final grade:')

06

07 # First Level if/else Statement: Judge whether the result entered is between 0 and 100

08 if result >= 0 and result <= 100:

09 # 2nd level if/elif/else statement

10 if result <60:

11 print('{0} below cannot obtain certificate of competency'. format(result))

12 elif result >= 60 and result <70:

13 print('{0} result is d'. format(result))

14 elif result >= 70 and result <80:

15 print('{0} result is c'. format(result))

16 elif result >= 80 and result <90:

17 print('{0} result is level b'. format(result))

18 else:

19 print('{0} result is grade a'. format(result))
```

```
20 else:
```

The execution results of the

```
21 print ('input error, input number must be between 0-100')
```

Program code analysis:

Lines 7-21: first-level if/else statement, used to judge whether the input result is between 0 and 100.

Lines 10-19: the second-level if/elif/else statement, which is used to judge which grade the inquired result belongs to.

In the next section, we will discuss loops one of the most important concepts.

The Loop Repeat Structure

This mainly refers to the loop control structure. A certain program statement is repeatedly executed according to the set conditions, and the loop will not jump out until the condition judgment is not established. In short, repetitive structures are used to design program blocks that need to be executed repeatedly, that is, to make program code conform to the spirit of structured design.

For example, if you want the computer to calculate the value of 1+2+3+4+ …+10, you don't need us to accumulate from 1 to 10 in the program code, which is originally tedious and repetitive, and you can easily achieve the goal by using the loop control structure. Python contains a while loop and a for loop, and the related usage is described below.

While loop

If the number of loops to be executed is determined, then using the for loop statement is the best choice. However, the while loop is more suitable for certain cycles that cannot be determined. The while loop statement is similar to the for loop statement and belongs to the pre-test loop. The working model of the pre-test loop is that the loop condition judgment expression must be checked at the beginning of the loop program block.

When the judgment expression result is true, the program statements in the loop block will be executed. We usually call the program statements in the

loop block the “loop body.” While loop also uses a conditional expression to judge whether it is true or false to control the loop flow. When the conditional expression is true, the program statement in the loop will be executed. When the conditional expression is false, the program flow will jump out of the loop.

The format of the While loop statement is as follows:

While conditional expression:

If the conditional expression holds, the flow chart of executing the while loop statement in this program block.

The while loop must include the initial value of the control variable and the expression for increasing or decreasing. When writing the loop program, it must check whether the condition for leaving the loop exists. If the condition does not exist, the loop body will be continuously executed without stopping, resulting in an "infinite loop," also called "dead loop."

The loop structure usually requires three conditions:

(1) The initial value of the loop variable.
(2) Cyclic conditional expression.
(3) Adjust the increase or decrease the value of cyclic variables.

For example, the following procedure:

```
first=1

While first < 10: # Loop Condition Expression

print( first)

first += 1 # adjusts the increase or decrease value of the loop variable.
```

When first is less than 10, the program statement in the while loop will be executed, and then first will be added with 1 until first is equal to 10. If the result of the conditional expression is False, it will jump out of the loop.

For loop

For loop, also known as count loop, is a loop form commonly used in programming. It can repeatedly execute a fixed number of loops. If the number of loop executions required is known to be fixed when designing

the program, then the for-loop statement is the best choice. The for loop in Python language can be used to traverse elements or table items of any sequence. The sequence can be tuples, lists or strings, which are executed in sequence.

The syntax is as follows:

For element variable in sequence:

Executed instructions

Else:

The program block of #else can be added or not added, that is, when using the for loop, the else statement can be added or not added. The meaning represented by the above Python syntax is that the for loop traverses all elements in a sequence, such as a string or a list, in the order of the elements in the current sequence (item, or table item).

For example, the following variable values can all be used as traversal sequence elements of a

```
first= "abcdefghijklmnopqrstuvwxyz "

second= ['january', 'march', 'may', 'july', 'august', 'october', 'december']

result= [a, e, 3, 4, 5, j, 7, 8, 9, 10]
```

Besides, if you want to calculate the number of times a loop is executed, you must set the initial value of the loop, the ending condition, and the increase or decrease value of the loop variable for each loop executed in the for-loop control statement. For loop every round, if the increase or decrease value is not specifically specified, it will automatically accumulate 1 until the condition is met.

For example, the following statement is a tuple (11 ~ 15) and uses the for loop to print out the numeric elements in the tuple: x = [11, 12, 13, 14, 15]

```
for first in x:

        print(first)
```

A more efficient way to write tuples is to call the range () function directly. The format of the range () function is as follows:

range ([initial value], final value [,increase or decrease value])

Tuples start from "initial value" to the previous number of "final value.” If no initial value is specified, the default value is 0; if no increase or decrease value is specified, the default increment is 1.

An example of calling the range () function is as follows: range (3) means that starting from the subscript value of 0, 3 elements are output, i.e., 0, 1 and 2 are three elements in total.

Range(1,6) means starting from subscript value 1 and ending before subscript value 6-1, that is, subscript number 6 is not included, i.e., 1, 2, 3, 4 and 5 are five elements. ·range (4,10,2) means starting from subscript value 4 and ending before subscript number 10, that is, subscript number 10 is excluded, and the increment value is 2, i.e., 4, 6 and 8 are three elements. The following program code demonstrates the use of the range () function in a for loop to output even numbers between 2 and 11 for i in range (2, 11, 2).

One more thing to pay special attention to when using the for loop is the print () function. If the print () is indented, it means that the operation to be executed in the for loop will be output according to the number of times the loop is executed. If there is no indentation, it means it is not in the for loop, and only the final result will be output.

We know that calling the range () function with the for loop can not only carry out accumulation operations but also carry out more varied accumulation operations with the parameters of the range () function. For example, add up all multiples of 5 within a certain range. The following sample program will demonstrate how to use the for loop to accumulate multiples of 5 within a range of numbers.

[Example Procedure: addition.py]

Accumulate multiples of 5 in a certain numerical range

```
01 # -*- coding: utf-8 -*-
02 """
03 Accumulate multiples of 5 within a certain numerical range
04 """
```

```
05 addition = 0 # stores the accumulated result
06
07 # enters for/in loop
08 for count in range(0, 21, 5):
09 addition += count # adds up the values
11 print('5 times cumulative result =',addition)
# Output cumulative result
```

Program code analysis:

Lines 08 and 09: Add up the numbers 5, 10, 15 and 20. Also, when executing a for loop, if you want to know the subscript value of an element, you can call Python's built-in enumerate function. The syntax format of the call is as follows: for subscript value, element variable in enumerate (sequence element).

For example (refer to sample program enumerate. py):

```
names = ["ram,” "raju,” "ravi"]
for index, x in enumerate(names):
```

The execution result of the above statement in print ("{0}-{1}.” format (index, x)) is displayed.

Nested loop

Next, we will introduce a for nested loop, that is, multiple for loop structures. In the nested for loop structure, the execution process must wait for the inner loop to complete before continuing to execute the outer loop layer by layer.

The double nested for loop structure format is as follows:

For example, a table can be easily completed using a double nested for loop. Let's take a look at how to use the double nested for loop to make the nine tables through the following sample program.

[Example Procedure: 99Table.py]

99 Table

```
01 # -*- coding: utf-8 -*-
02 """
03 Program Name: Table
04 """
05
06 for x in range(6,68 ):
07 for y in range(1, 9):
08 print("{0}*{1}={52: ^2}.”format(y, x, x * y), end=" ")
```

99 is a very classic example of nested loops. If readers have learned other programming languages, I believe they will be amazed at the brevity of Python. From this example program, we can clearly understand how nested loops work. Hereinafter, the outer layer for the loop is referred to as the x loop, and the inner layer for loop is referred to as the y loop.

When entering the x loop, x=1. When the y loop is executed from 1 to 9, it will return to the x loop to continue execution. The print statement in the y loop will not wrap. The print () statement in the outer x loop will not wrap until the y loop is executed and leaves the y loop. After the execution is completed, the first row of nine tables will be obtained. When all X cycles are completed, the table is completed.

Note that the common mistake for beginners is that the sentences of the inner and outer loops are staggered. In the structure of multiple nested loops, the inner and outer loops cannot be staggered; otherwise, errors will be caused.

The continue instruction and break instruction are the two loop statements we introduced before. Under normal circumstances, the while loop is to judge the condition of the loop before entering the loop body. If the condition is not satisfied, it will leave the loop, while for loop ends the execution of the loop after all the specified elements are fetched. However, the loop can also be interrupted by continue or break. The main purpose of

break instruction is to jump out of the current loop body, just like its English meaning, break means "interrupt."

If you want to leave the current loop body under the specified conditions in the loop body, you need to use the break instruction, whose function is to jump off the current for or while loop body and give the control of program execution to the next line of program statements outside the loop body. In other words, the break instruction is used to interrupt the execution of the current loop and jump directly out of the current loop.

Python Advanced Guide

Table of Contents

Chapter 1 Object-Oriented programming

We are now going to look at the four concepts of object-oriented programming and how they apply to Python.

Inheritance

The first major concept is called “inheritance.” This refers to things being able to derive from another. Let’s take sports cars for instance. All sports cars are vehicles, but not all vehicles are sports cars. Moreover, all sedans are vehicles, but all vehicles are not sedans, and sedans are *certainly* not sports cars, even though they’re both vehicles.
So basically, this concept of Object-Oriented programming says that things can and should be chopped up into as small and fine of precise concepts as possible.
In Python, this is done by deriving classes.

Let’s say we had another class called SportsCar.

```
class Vehicle(object):
    def __init__(self, makeAndModel, prodYear, airConditioning):
        self.makeAndModel = makeAndModel
        self.prodYear = prodYear
        self.airConditioning = airConditioning
        self.doors = 4
    def honk(self):
        print "%s says: Honk! Honk!" % self.makeAndModel
```

Now, below that, create a new class called SportsCar, but instead of deriving *object* , we’re going to derive from Vehicle.

```
class SportsCar(Vehicle)
```

```
    def__init__(self, makeAndModel, prodYear, airConditioning):
        self.makeAndModel = makeAndModel
        self.prodYear = prodYear
        self.airConditioning = airConditioning
        self.doors = 4
```

Leave out the honk function, we only need the constructor function here. Now declare a sports car. I'm just going to go with the Ferrari.

```
ferrari = SportsCar("Ferrari Laferrari", 2016, True)
```

Now test this by calling

```
ferrari.honk()
```

and then saving and running. It should go off without a hitch.

Why is this? This is because the notion of inheritance says that a child class derives functions and class variables from a parent class. Easy enough concept to grasp. The next one is a little tougher.

Polymorphism

The idea of polymorphism is that the same process can be performed in different ways depending upon the needs of the situation. This can be done in two different ways in Python: *method overloading* and *method overriding* .

Method overloading is defining the same function twice with different arguments. For example, we could give two different initializer functions to our Vehicle class. Right now, it just assumes a vehicle has 4 doors. If we wanted to specifically say how many doors a car had, we could make a new initializer function below our current one with an added *doors* argument, like so (the newer one is on the bottom):

```
def __init__(self, makeAndModel, prodYear, airConditioning):
        self.makeAndModel = makeAndModel
        self.prodYear = prodYear
```

```
        self.airConditioning = airConditioning
        self.doors = 4
def __init__(self, makeAndModel, prodYear, airConditioning, doors):
self.makeAndModel = makeAndModel
self.prodYear = prodYear
self.airConditioning = airConditioning
self.doors = doors
```

Somebody now when creating an instance of the Vehicle class can *choose* whether they define the number of doors or not. If they don't, the number of doors is assumed to be 4.

Method overriding is when a child class *overrides* a parent class's function with its code.

To illustrate, create another class which extends Vehicle called Moped. Set the doors to 0, because that's absurd, and set air conditioning to false. The only relevant arguments are make/model and production year. It should look like this:

```
class Moped(Vehicle):
    def __init__(self, makeAndModel, prodYear):
        self.makeAndModel = makeAndModel
        self.prodYear = prodYear
        self.airConditioning = False
        self.doors = 0
```

Now, if we made an instance of the Moped class and called the honk() method, it would honk. But it is common knowledge that mopeds don't honk, they beep. So let's override the parent class's honk method with our own. This is super simple. We just redefine the function in the child class:

```
def honk(self):
```

```
print “%s says: Beep! Beep!” % self.makeAndModel
```

I’m part of the 299,000,000 Americans who couldn’t name a make and model of moped if their life depended on it, but you can test out if this works for yourself but declaring an instance of the Moped class and trying it out.

Abstraction

The next major concept in object-oriented programming is *abstraction* . This is the notion that the programmer and user should be far from the inner workings of the computer. This has two benefits.

The first is that it decreases the inherent security risks and the possibility for catastrophic system errors, by either human or otherwise. By abstracting the programmer from the inner workings of the computer like memory and the CPU and often even the operating system, there’s a low chance of any sort of mishap causing irreversible damage.

The second is that the abstraction innately makes the language easier to understand, read, and learn. Though it makes the language a tad bit less powerful by taking away some of the power that the user has over the entire computer architecture, this is traded instead for the ability to program quickly and efficiently in the language, not wasting time dealing with trivialities like memory addresses or things of the like.

These apply in Python because, well, it’s incredibly simple. You can’t get down into the nitty-gritty of the computer, or do much with memory allocation or even specifically allocate an array size too easily, but this is a tradeoff for amazing readability, a highly secure language in a highly secure environment, and ease of use with programming. Compare the following snippet of code from C:

```
#include <stdio.h>

int main(void) {

printf(“hello world”);
```

```
return 0;
}
```

to the Python code for doing the same:

```
print “hello world”
# That’s it. That’s all there is to it.
```

Abstraction is generally a net positive for a large number of applications that are being written today, and there’s a reason Python and other object-oriented programming languages are incredibly popular.

Encapsulation

The last major concept in object-oriented programming is that of encapsulation. This one’s the easiest to explain. This is the notion that common data should be put together, and that code should be modular. I’m not going to spend long explaining this because it’s a super simple concept. The entire notion of classes is as concise of an example as you can get for encapsulation: common traits and methods are bonded together under one cohesive structure, making it super easy to create things of the sort without having to create a ton of super-specific variables for every instance.

Well, there we go. We finally made it to the end of our little Python adventure. First, I’d like to say thank you for making it through to the end of Python for Beginners: The Ultimate Guide to Python Programming. Let’s hope it was informative and able to provide you with all of the tools you need to achieve your goals, whatever they may be.

The next step is to use this knowledge. Whether as a hobby or a career move, by learning the basics of Python, you just made one of the best decisions of your life, and your goal now should be finding ways to use it in your day-to-day life to make life easier or to accomplish things you’ve wanted to accomplish for a long while.

Chapter 2 Essential Programming Tools

Bash Script

A Bash script is a data file that contains a sequence of commands, which you can usually code, but will save you time if you don't. Take note that in programming, any code that you could normally run on the command line could be placed on the script, and it will be executed precisely as it is. Likewise, any code that you could be placed into a script can also normally be executed exactly as it is.

There can be many processes manifesting one program performing in memory simultaneously. For instance, you can use two terminals and still run the command prompt at the same time. In such a case, there will be two command prompt processes that are existing at the same time within the system. When they complete the execution, the system can terminate them, so there will be no more processes that are representing the command prompt.

When you are using the terminal, you can run the Bash script to provide you a shell. When you initiate a script, it will not execute in this process, but rather will begin a new process to be executed inside. But as a beginner in programming, you don't need to worry too much about the mechanism of this script as running Bash can be very easy.

You may also encounter some tutorials about script execution, which is pretty much the same thing. Before executing the script, it should have permission in place, as the program will return an error message if you fail to grant permission.

Below is a sample Bash script:

```
#!/bin/bash

# declare STRING variable

STRING="Hello Python"

#print variable on a screen
```

```
echo $STRING
```

You can use the 755 shorthand so you can modify the script and make sure that you can share it with others to execute the script.

Python RegEx

RegEx refers to the regular expression that defines the string of text, which allows you to generate patterns in managing, matching, and locating text. Python is a good example of a programming language, which uses regex. Regex can also be utilized in text editors and from the command line to search for a text inside a file.

When you first encounter regex, you might think that it is a different programming language. But mastering regex could save you tons of hours if you are working with the text or you require to parse large amounts of data.

The RE module provides complete support for regex in Python. It also increases the exception re.error if there is an error while using or compiling a regex. There are two essential functions that you have to know in using regex for Python. But before that, you should understand that different characters have special meaning when they are used in a regular expression. So you will not be confused in working with regex, since we will use r'expression' when we mean Raw Strings.

The two important functions in Python regex are the search and match functions.

The search function looks for the first instance of an RE pattern inside a string with optional flags. The search function has the following parameters (below is the syntax):

- String - will be searched to match the pattern within the string
- Pattern - regex to be matched
- Flags - modifiers that can be specified using bitwise

The re.search function can return an object match if successful, and object none if failed. You should use groups() or groups(num) function of object match to find a matched expression.

Below is an example of a code using the search function:

```
import re

#Check if the string starts with "The" and ends with "Spain":

txt = "The rain in Spain"

x = re.search("^The.*Spain$", txt)

if (x):

  print("YES! We have a match!")

else:

  print("No match")
```

The output will be:

```
YES! We have a match!
```

Meanwhile, the match function will try to match the RE pattern in order to string with specific flags. Below is the syntax for the match function:

The match function has the following parameters:

- String - this will be searched to match the pattern at the start of the string
- Pattern - this is the regex to be matched
- Flags - modifiers that can be specified using bitwise

Python Package Manager

In programming, package managers refer to the tools used to automate the system of installing, configuring, upgrading, and uninstalling programs for a specific language system in an orderly manner.

Also known as a package management system, it also deals with the distribution and archiving of data files including the name of the software, version, number, purpose and a sequence of dependencies needed for the language to properly run.

When you use a package manager, the metadata will be archived in the local database usually to avoid code mismatches and missing permissions.

In Python, you can use a utility to locate, install, upgrade and eliminate Python packages. It can also identify the most recent version of a package installed on the system as well as automatically upgrade the current package from a remote or local server.

Python Package Manager is not free and you can only use it through ActivePython. It also utilizes repositories, which are a group of pre-installed packages and contain different types of modules.

Source Control

In programming, a source control (also known as version control or revision control), manages the changes to the codes, which is identified by a letter or number code regarded as the revision number or just revision. For instance, an initial set of code is known as revision 1, and then the first modification will be revision 2. Every revision will be linked with a timestamp as well as the person who made the change. Revisions are important so the code could be restored or compared.

Source control is important if you are working with a team. You can combine your code changes with other code changes done by a developer through different views that will show detailed changes, then combine the proper code into the primary code branch.

Source control is crucial for coding projects regardless if you are using Python or other languages. Take note that each coding project should start by using a source control system such as Mercurial or Git.

Various source control systems have been developed ever since the existence of programming. Before, proprietary control systems provided features that are customized for large coding projects and particular project workflows. But today, open-source systems can be used for source control regardless if you are working on a personal code or as part of a large team.

It is ideal to use an open-source version control system in your early Python codes. You can use either Mercurial or Git, which are both open source and used for distributing source control.

Subversion is also available, which can be used to centralize the system to check files and minimize conflicting merges.

Bringing It All Together

Programming tools will make your work easier. The tools discussed in this part will save you a lot of time, collaborate easier, and make your codes seamless. In summary, we have learned the following:

- A Bash Script can save you a lot of coding time and will make your code lines more organized and readable.
- Regular Expressions or RegEx can help you find, search, and match strings of text inside your codes, so you don’t have to browse line by line or analyze each code on your own.
- Package Managers will automate the system to easily install, upgrade, configure or remove specific programs that could aid your coding work
- Source Control is crucial to managing the revisions, regardless if you are working alone or with a team, so you can restore changes or compare revisions.

You can still work on your codes without these tools, but your life will be easier if you choose to use them.

Chapter 3 Working with Files

The next thing that we need to focus on when it comes to working with Python is making sure we know how to work and handle files. It may happen that you are working with some data and you want to store them while ensuring that they are accessible for you to pull up and use when they are needed later. You do have some choices in the way that you save the data, how they are going to be found later on, and how they are going to react in your code.

When you work with the files, you will find that the data is going to be saved on a disk, or you can re-use in the code over and over again as much as you would like. This chapter is going to help us learn a bit more about how to handle some of the work that we need to do to ensure the files behave the way that they should, and so much more.

Now, we are going to enter into file mode on the Python language, and this allows you to do a few different options along the way. A good way to think about this is that you can think about it like working on a file in Word. At some point, you may try to save one of the documents that you are working with so that it doesn't get lost and you can find them later on. These kinds of files in Python are going to be similar. But you won't be saving pages as you did on Word, you are going to save parts of your code.

You will find with this one that there are a few operations or methods that you are able to choose when it comes to working with files. And some of these options will include:

- Closing up a file you are working on.
- Creating a brand new file to work on.
- Seeking out or moving a file that you have over to a new location to make it easier to find.
- Writing out a new part of the code on a file that was created earlier.

Creating new files

The first task that we are going to look at doing here is working on creating a file. It is hard to do much of the other tasks if we don't first have a file in place to help us out. If you would like to be able to make a new file and then add in some code into it, you first need to make sure the file is opened up inside of your IDLE. Then you can choose the mode that you would like to use when you write out your code.

When it comes to creating files on Python, you will find there are three modes that you are able to work with. The three main modes that we are going to focus on here include append (a), mode(x) and write(w).

Any time that you would like to open up a file and make some changes in it, then you would want to use the write mode. This is the easiest out of the three to work with. The write method is going to make it easier for you to get the right parts of the code set up and working for you in the end.

The write function is going to be easy to use and will ensure that you can make any additions and changes that you would like to the file. You can add in the new information that you would like to the file, change what is there, and so much more. If you would like to see what you can do with this part of the code with the write method, then you will want to open up your compiler and do the following code:

```
#file handling operations

#writing to a new file hello.txt

f = open('hello.txt', 'w', encoding = 'utf-8' )

f.write("Hello Python Developers!")

f.write("Welcome to Python World")

f.flush()

f.close()
```

From here, we need to discuss what you can do with the directories that we are working with. The default directory is always going to be the current directory. You are able to go through and switch up the directory where the code information is stored, but you have to take the time, in the beginning, to change that information up, or it isn't going to end up in the directory that you would like.

Whatever directory you spent your time in when working on the code is the one you need to make your way back to when you want to find the file later on. If you would like it to show up in a different directory, make sure that you move over to that one before you save it and the code. With the option that we wrote above, when you go to the current directory (or the directory that you chose for this endeavor, then you will be able to open up the file and see the message that you wrote out there.

For this one, we wrote a simple part of the code. You, of course, will be writing out codes that are much more complicated as we go along. And with those codes, there are going to be times when you would like to edit or overwrite some of what is in that file. This is possible to do with Python, and it just needs a small change to the syntax that you are writing out. A good example of what you can do with this one includes:

```
#file handling operation s

#writing to a new file hello.txt

f = open('hello.txt', 'w', encoding = 'utf-8')

f.write("Hello Python Developers!")

f.write("Welcome to Python World")

mylist = ["Apple", "Orange", "Banana"]

#writelines() is used to write multiple lines into the file

f.write(mylist)

f.flush()

f.close()
```

The example above is a good one to use when you want to make a few changes to a file that you worked on before because you just need to add in one new line. This example wouldn't need to use that third line because it just has some simple words, but you can add in anything that you want to the program, just use the syntax above and change it up for what you need.

What are the binary files?

One other thing that we need to focus on for a moment before moving on is the idea of writing out some of your files and your data in the code as a binary file. This may sound a bit confusing, but it is a simple thing that Python will allow you to do. All that you need to do to make this happen is to take the data that you have and change it over to a sound or image file, rather than having it as a text file.

With Python, you are able to change any of the code that you want into a binary file. It doesn't matter what kind of file it was in the past. But you do need to make sure that you work on the data in the right way to ensure that it is easier to expose in the way that you want later on. The syntax that is going to be needed to ensure that this will work well for you will be below:

```
# write binary data to a file

# writing the file hello.dat write binary mode

F = open('hello.dat', 'wb')

# writing as byte strings

f.write("I am writing data in binary file!/n")

f.write("Let's write another list/n")

f.close()
```

If you take the time to use this code in your files, it is going to help you to make the binary file that you would like. Some programmers find that they like using this method because it helps them to get things in order and will make it easier to pull the information up when you need it.

Opening your file up

So far, we have worked with writing a new file and getting it saved, and working with a binary file as well. In these examples, we got some of the basics of working with files down so that you can make them work for you and you can pull them up any time that you would like.

Now that this part is done, it is time to learn how to open up the file and use it, and later even make changes to it, any time that you would like. Once you open that file up, it is going to be so much easier to use it again and

again as much as you would like. When you are ready to see the steps that are needed to open up a file and use it, you will need the following syntax.

```
# read binary data to a file

#writing the file hello.dat write append binary mode

with open(“hello.dat”, ‘rb’) as f:

data = f.read()

text = data.decode(‘utf-8’)

print(text)
```

The output that you would get form putting this into the system would be like the following:

```
Hello, world!

This is a demo using with

This file contains three lines

Hello worl d

This is a demo using with

This file contains three lines  .

Seeking out a file you need
```

And finally, we need to take a look at how you can seek out some of the files that you need on this kind of coding language. We already looked at how to make the files, how to store them in different manners, how to open them and rewrite on them, and then how to seek the file. But there are times where you are able to move one of the files that you have over to a new location.

For example, if you are working on a file and as you do that, you find that things are not showing up the way that you would like it to, then it is time to fix this up. Maybe you didn’t spell the time of the identifier the right way, or the directory is not where you want it to be, then the seek option

may be the best way to actually find this lost file and then make the changes, so it is easier to find later on.

With this method, you are going to be able to change up where you place the file, to ensure that it is going to be in the right spot all of the time or even to make it a bit easier for you to find it when you need. You just need to use a syntax like what is above to help you make these changes.

Working through all of the different methods that we have talked about in this chapter are going to help you to do a lot of different things inside of your code. Whether you would like to make a new file, you want to change up the code, move the file around, and more; you will be able to do it all using the codes that we have gone through in this chapter.

Chapter 4 Exception Handling

Exception handling is error management. It has three purposes.

1. It allows you to debug your program.

2. It allows your program to continue running despite encountering an error or exception.

3. It allows you to create your customized errors that can help you debug, remove and control some of Python's nuances, and make your program function as you want it to.

Handling the Zero Division Error Exception

Exception handling can be an easy or difficult task depending on how you want your program to flow and your creativity. You might have scratched your head because of the word creativity. Programming is all about logic, right? No.

The core purpose of programming is to solve problems. A solution to a problem does not only require logic. It also requires creativity. Have you ever heard of the phrase, "Think outside of the box?"

Program breaking exceptions can be a pain and they are often called bugs. The solution to such problems is often elusive. And you need to find a workaround or risk rewriting your program from scratch.

For example, you have a calculator program with this snippet of code when you divide:

```
>>> def div(dividend, divisor):
        print(dividend / divisor)
>>> div(5, 0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in div
ZeroDivisionError: division by zero
```

Of course, division by zero is an impossible operation. Because of that, Python stops the program since it does not know what you want to do when this is encountered. It does not know any valid answer or response.
That being said, the problem here is that the error stops your program entirely. To manage this exception, you have two options. First, you can make sure to prevent such operation from happening in your program. Second, you can let the operation and errors happen, but tell Python to continue your program.
Here is what the first solution looks like:

```
>>> def div(dividend, divisor):
        if (divisor != 0):
            print(dividend / divisor)
        else:
            print("Cannot Divide by Zero.")
>>> div(5, 0)
Cannot Divide by Zero.
```

Here is what the second solution looks like:

```
>>> def div(dividend, divisor):
        try:
            print(dividend / divisor)
        except:
            print("Cannot Divide by Zero.")
>>> div(5, 0)
Cannot Divide by Zero.
```

Remember the two core solutions to errors and exceptions. One, prevent the error from happening. Two, manage the aftermath of the error.

Using Try-Except Blocks

In the previous example, the try except block was used to *manage* the error. However, you or your user can still do something to screw your solution up. For example:

```
>>> def div(dividend, divisor):
        try:
            print(dividend / divisor)
        except :
            print("Cannot Divide by Zero.")
>>> div(5, "a")
Cannot Divide by Zero.
```

The statement prepared for the “except” block is not enough to justify the error that was created by the input. Dividing a number by a string does not warrant a “Cannot Divide by Zero.” message.

For this to work, you need to know more about how to use the except block properly. First of all, you can specify the error that it will capture and respond to by indicating the exact exception. For example:

```
>>> def div(dividend, divisor):
        try:
            print(dividend / divisor)
        except ZeroDivisionError:
            print("Cannot Divide by Zero.")
>>> div(5, 0)
Cannot Divide by Zero.
>>> div(5, "a")
Traceback (most recent call last):
  File "<stdin>", line 1, <module >
```

```
  File "<stdin>", line 3, in div

TypeError: unsupported operand type(s) for /: 'int' and 'str'
```

Now, the error that will be handled has been specified. When the program encounters the specified error, it will execute the statements written on the “except” block that captured it. If no except block is set to capture other errors, Python will then step in, stop the program, and give you an exception.

But why did that happen? When the example did not specify the error, it handled everything. That is correct. When the “except” block does not have any specified error to look out for, it will capture any error instead. For example:

```
>>> def div(dividend, divisor):

        try:

            print(dividend / divisor)

        except:

            print("An error happened.")

>>> div(5, 0)

An error happened.

>>> div(5, "a")

An error happened.
```

That is a better way of using the “except” block if you do not know exactly the error that you might encounter.

Reading an Exception Error Trace Back

The most important part in error handling is to know how to read the traceback message. It is fairly easy to do. The traceback message is structured like this:

<Traceback Stack Header>

  <File Name>, <Line Number>, <Function/Module>

<Exception>: <Exception Description>

Here are things you need to remember:

- The traceback stack header informs you that an error occurred.
- The filename tells you the name of the file where the fault is located. Since the examples in the book are coded using the interpreter, it always indicated that the file name is "<stdin>" or standard input.
- The line number tells the exact line number in the file that caused the error. Since the examples are tested in the interpreter, it will always say line. However, if the error is found in a code block or module, it will return the line number of the statement relative to the code block or module.
- The function/module part tells what function or module owns the statement. If the code block does not have an identifier or the statement is declared outside code blocks, it will default to <module>.
- The exception tells you what kind of error happened. Some of them are built-in classes (e.g., ZeroDivisionError, TypeError, and etcetera) while some are just errors (e.g., SyntaxError). You can use them on your except blocks.
- The exception description gives you more details with regards to how the error occurred. The description format may vary from error to error.

Using exceptions to prevent crashes

Anyway, to know the exceptions that you can use, all you need to do is to generate the error. For example, using the TypeError found in the previous

example, you can capture that error too and provide the correct statements in response.

```
>>> def div(dividend, divisor):
        try:
            print(dividend / divisor)
        except ZeroDivisionError:
            print("Cannot Divide by Zero.")
        except TypeError:
            print("Cannot Divide by Anything Other Than a Number.")
        except:
            print("An unknown error has been detected.")
>>> div(5, 0)
Cannot Divide by Zero.
>>> div(5, "a")
Cannot Divide by Anything Other Than a Number.
>>> div(undeclaredVariable / 20)
An unknown error has been detected.
```

However, catching errors this way can still be problematic. It does allow you to prevent a crash or stop, but you have no idea about what exactly happened. To know the unknown error, you can use the *as* the keyword to pass the Exception details to a variable. Convention wise, the variable detail is often used for this purpose.
For example:

```
>>> def div(dividend, divisor):
        try:
            print(dividend / divisor)
```

```
        except Exception as detail:
            print("An error has been detected.")
            print(detail)
            print("Continuing with the program.")
>>> div(5, 0)
An error has been detected.
```

Division by zero

```
Continuing with the program.
>>> div(5, "a")
An error has been detected.
unsupported operand type(s) for /: 'int' and 'str'
Continuing with the program.
```

The Else Block

There are times that an error happens in the middle of your code block. You can catch that error with try and except. However, you might not want to execute any statement in that code block if an error happens. For example:

```
>>> def div(dividend, divisor):
        try:
            quotient = dividend / divisor
        except Exception as detail:
            print("An error has been detected." )
            print(detail)
            print("Continuing with the program.")
        print(str(dividend) + " divided by " + str(divisor) + " is:")
```

```
        print(quotient)
>>> div(4, 2)
4 divided by 2 is:
2.0
>>> div(5, 0)
An error has been detected.
division by zero
Continuing with the program.
5 divided by 0 is:
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 8, in div
    Print(quotient)
UnboundLocalError: local variable 'quotient' referenced before assignment
```

As you can see, the next statements after the initial fault are dependent on it; thus they are also affected. In this example, the variable quotient returned an error when used after the try and except block since its supposed value was not assigned because the expression assigned to it was impossible to evaluate.

In this case, you would want to drop the remaining statements that are dependent on the contents of the try clause. To do that, you must use the else block. For example:

```
>>> def div(dividend, divisor):
        try:
            quotient = dividend / divisor
        except Exception as detail:
```

```
            print("An error has been detected.")
            print(detail)
            print("Continuing with the program.")
        else:
            print(str(dividend) + " divided by " + str(divisor) + " is:")
            print(quotient)
>>> div(4, 2)
4 divided by 2 is:
2
>>> div(5, 0)
An error has been detected.
division by zero
Continuing with the program.
```

The first attempt on using the function with proper arguments went well.
On the second attempt, the program did not execute the last two statements under the else block because it returned an error.
The else block always follows except blocks. The function of the else block is to let Python execute the statements under it when the try block did not return and let Python ignore them if an exception happens.

Failing Silently

Silent fails or failing silently is a programming term often used during error and exception handling.
In a user's perspective, silent failure is a state wherein a program fails at a certain point but never informs a user.
In a programmer's perspective, silent failure is a state wherein the parser, runtime development environment, or compiler fails to produce an error or exception and proceed with the program. This often leads to unintended results.

A programmer can also induce silent failures when he either ignores exceptions or bypasses them. Alternatively, he blatantly hides them and creates workarounds to make the program operate as expected even if an error happened. He might do that because of multiple reasons such as the error is not program breaking or the user does not need to know about the error.

Handling the File Not Found Exception Error

There will be times when you will encounter the FileNotFoundError. Handling such error depends on your intent or purpose with regards to opening the file. Here are common reasons you will encounter this error:

- You did not pass the directory and filename as a string.
- You misspelled the directory and filename.
- You did not specify the directory.
- You did not include the correct file extension.
- The file does not exist.

The first method to handle the FileNotFoundError exception is to make sure that all the common reasons do not cause it. Once you do, then you will need to choose the best way to handle the error, which is completely dependent on the reason you are opening a file in the first place.

Checking If File Exists

Again, there are always two ways to handle an exception: preventive and reactive. The preventive method is to check if the file exists in the first place.
To do that, you will need to use the os (os.py) module that comes with your Python installation. Then, you can use its path module's isfile() function. The path module's file name depends on the operating system (posixpath for UNIX, ntpath for Windows, macpath for old MacOS). For example:

```
>>> from os import path
>>> path.isfile("random.txt")
False
>>> path.isfile("sampleFile.txt")
True
```

Try and Except

You can also do it the hard way by using try, except, and else blocks.

```
>>> def openFile(filename):
        try:
            x = open(filename, "r")
        except FileNotFoundError:
            print("The file '" + filename + "' does not exist."
        except FileNotFound:
            print("The file '" + filename + "' does exist."
>>> openFile("random.txt")
The file 'random.txt' does not exist.
>>> openFile("sampleFile.txt")
The file 'sampleFile.txt' does exist.
```

Creating a New File

If the file does not exist, and your goal is to overwrite any existing file anyway, then it will be best for you to use the "w" or "w+" access mode. The access mode creates a new file for you if it does not exist. For example:

```
>>> x = open("new.txt", "w")
>>> x.tell()
```

```
0
```

If you are going to read and write, use "w+" access mode instead.

Practice Exercise

Try to break your Python by discovering at least ten different exceptions.
After that, create a loop.
In the loop, create ten statements that will create each of the ten different exceptions that you find inside one try block.
Each time the loop loops, the next statement after the one that triggered an exception should trigger another and so on.
Provide a specific except block for each one of the errors.

Solution

Conclusion

Thanks for reading till the end!

There are a lot of other coding languages out there that you are able to work with, but Python is one of the best that works for most beginner programmers, providing the power and the ease of use that you are looking for when you first get started in this kind of coding language. This guidebook took the time to explore how Python works, along with some of the different types of coding that you can do with it.

In addition to seeing a lot of examples of how you can code in Python and how you can create some of your programs in this language, we also spent some time looking at how to work with Python when it comes to the world of machine learning, artificial intelligence, and data analysis. These are topics and parts of technology that are taking off and many programmers are trying to learn more about it. And with the help of this guidebook, you will be able to handle all of these, even as a beginner in Python.

When you are ready to learn more about how to work with the Python coding language and how you can make sure that you can even use Python along with data analysis, artificial intelligence, and machine learning, make sure to check out again this guidebook to help you get started.

Python for Data Analysis

Table of Contents

Introduction

Data analysis plays an important role in many aspects of life today. From the moment you wake up, you interact with data at different levels. Many important decisions are made based on data analytics. Companies need data to help them meet many of their goals. As the population of the world keeps growing, its customer base keeps expanding. In light of this, they must find ways of keeping their customers happy while at the same time, meeting their business goals.

Given the nature of competition in the business world, it is not easy to keep customers happy. Competitors keep preying on each other's customers, and those who win have another challenge ahead, how to maintain the customers lest they slide back to their former business partners? This is one area where Data Analysis comes in handy.

To understand their customers better, companies rely on data. They collect all manner of data at each point of interaction with their customers. Data are useful in several ways. The companies learn more about their customers, thereafter clustering them according to their specific needs. Through such segmentation, the company can attend to the customers' needs better and hope to keep them satisfied for longer.

However, data analytics is not just about customers and the profit motive. It is also about governance. Governments are the biggest data consumers all over the world. They collect data about citizens, businesses, and every other entity that they interact with at any given point. This is important information because it helps in a lot of instances.

For planning purposes, governments need accurate data on their population so that funds can be allocated accordingly. Equitable distribution of resources is something that cannot be achieved without proper Data Analysis. Other than planning, there is also the security angle. To protect the country, the government must maintain different databases for different reasons. There are high profile individuals who must be accorded special security detail, top threats have to be monitored at all times, and so forth. To meet the security objective, the government has to obtain and maintain updated data on persons of interest at all times.

There is so much more to Data Analysis than the corporate and government decisions. As a programmer, you are venturing into an industry that is challenging and exciting at the same time. Data doesn't lie unless it is manipulated, in which case you need insane Data Analysis and handling skills. As a data analyst, you will come across many challenges and problems that need solutions that can only be handled through Data Analysis. The way you interact with data can make a huge difference, bigger than you can imagine.

There are several tools you can use for Data Analysis. Many people use Microsoft Excel for Data Analysis and it works well for them. However, there are limitations of using Excel, which you can overcome through Python. Learning Python is a good initiative, given that it is one of the easiest programming languages. It is a high-level programming language because its syntax is so close to the normal language we use. This makes it easier for you to master Python concepts.

For expert programmers, you have gone beyond learning about the basics of Python and graduated into using Python to solve real-world problems. Many problems can be solved through Data Analysis. The first challenge is usually understanding the issue at hand and then working on a data solution for it.

This book follows a series of elaborate books that introduced you to Data Analysis using Python. Some important concepts have been reiterated since the beginning of the series to help you remember the fundamentals. Knowledge of Python libraries is indeed important. It is by understanding these libraries that you can go on to become an expert data analyst with Python.

As you interact with data, you do understand the importance of cleaning data to ensure the outcome of your analysis is not flawed. You will learn how to go about this and build on that to make sure your work is perfect. Another challenge that many organizations have is protecting the integrity of data. You should try to protect your organization from using contaminated data. There are procedures you can put in place to make sure that you use clean data all the time.

We live in a world where data is at the center of many things we do. Data is produced and stored in large amounts daily from automated systems.

Learning Data Analysis through Python should help you process and extract information from data and make meaningful conclusions from them. One area where these skills will come in handy is forecasting. Through Data Analysis, you can create predictive models that should help your organization meet its objectives.

A good predictive model is only as good as the quality of data introduced into it, the data modeling methods, and more importantly, the dataset used for the analysis. Beyond data handling and processing, one other important aspect of Data Analysis is visualization, which is about presentation. Your data model should be good enough for an audience to read and understand it at the first point of contact. Apart from the audience, you should also learn how to plot data on different visualizations to help you get a rough idea of the nature of the data you are working with.

When you are done with Data Analysis, you should have a data model complete with visual concepts that will help in predicting outcomes and responses before you can proceed to the testing phase. Data analysis is a study that is currently in high demand in different fields. Knowing what to do, as well as when and how to handle data, is an important skill that you should not take for granted. Through this, you can build and test a hypothesis and go on to understand systems better.

Chapter 1 What is Data Analysis?

Now that we have been able to spend some time taking a look at the ideas of Python and what we can do with that coding language, it is time for us to move on to some of the things that we are able to do with all of that knowledge and all of the codes that we are looking. We are going to take a look here to see more about Data Analysis, and how we can use this to help us see some good results with our information as well.

Companies have spent a lot of time looking at Data Analysis and what it has been able to do for them. Data is all around us, and it seems like each day, tons of new information is available for us to work with regularly. Whether you are a business trying to learn more about your industry and your customers, or just an individual who has a question about a certain topic, you will be able to find a wealth of information to help you get started.

Many companies have gotten into a habit of gathering up data and learning how to make it work for their needs. They have found that there is a lot of insights and predictions within data to make sure that it is going to help them out in the future. If the data is used properly, and we can gain a good handle of that data, it can be used to help our business become more successful.

Once you have gathered the data, there is going to be some work to do. Just because you are able to gather up all of that data doesn't mean that you will be able to see what patterns are inside. This is where the process of Data Analysis is going to come into play to help us see some results as well. This is a process that is meant to ensure that we fully understand what is inside of our data and can make it easier to use all of that raw data to make some informed and smart business decisions.

To make this a bit further, Data Analysis is going to be a practice where we can take some of the raw data that our business has been collecting, and then organize and order it to ensure that it can be useful. During this process, the information that is the most useful is extracted and then used from that raw data. The process of organizing and thinking about data is going to be so important here because it is the key to helping us to understand what the data do and what it doesn't contain.

There are going to be many different methods that we can use to approach this kind of Data Analysis, which is part of the appeal of what goes with this. We will find that with all of these methods, it is easier for us to work with a Data Analysis because we can make some of the adaptations that are needed to the process to ensure it works for our own needs, no matter what industry we are working in, or what our main question is in the beginning.

The one thing that we need to be careful about when we are working with Data Analysis, though, is to be careful about the way that we manipulate the data that we have. It is really easy for us to go through and manipulate the data in the wrong way during the analysis phase, and then end up pushing certain conclusions or agendas that are not there. This is why we need to pay some close attention to when the Data Analysis is presented to us and to think critically about the data and the conclusions that we were able to get out of it.

If you are worried about a source that is being done, and if you are not sure that you are able to complete this kind of analysis without some biases in it, then it is important to find someone else to work on it or choose a different source. There is plenty data out there and it can help your business to see some results, but you have to be careful about these biases, or they will lead us to the wrong decisions in the end if we are not careful.

In addition, you will find that during the Data Analysis, the raw data that you will work with can take on a variety of forms. This can include things like observations, survey responses, and measurements, to name a few. The sources that you use for this kind of raw data will vary based on what you are hoping to get out of it, what your main question is all about, and more.

In its raw form, the data that we are gathering is going to be very useful to work with, but you may find that it is a bit overwhelming to work with as well. This is a problem that many companies are going to have when they work with Data Analysis and something that you will have to spend some time exploring and learning more about, as well.

Over the time that you spend on Data Analysis and all of the steps that come with the process, the raw data is going to be ordered in a manner that makes it as useful to you as possible. For example, we may send out a survey and then will tally up the results that we get. This is going to be done because it helps us to see at a glance how many people decided to answer

the survey at all, and how people were willing to respond to some of the specific questions that were on that survey.

In the process of going through and organizing the data, a trend is likely going to emerge, and sometimes more than one trend. In addition, we are going to then be able to take some time to highlight these trends, usually in the write-up that is being done on the data. This needs to be highlighted because it ensures that the person who is reading that information is going to take note.

There are plenty of places that we are going to see this. For example, in a casual kind of survey that we may try to do, you may want to figure out the preferences between men and women of what ice cream flavors they like the most. In this survey, maybe we find out that women and men are going to express a fondness for chocolate. Depending on who is using this information and what they are hoping to get out of that information, it could be something that the researcher is going to find very interesting.

Modeling the data that is found out of the survey, or out of another form of Data Analysis, with the use of mathematics and some of the other tools out there, can sometimes exaggerate the points of interest, such as the ice cream preferences from before, in our data, which is going to make it so much easier for anyone who is looking over the data, especially the researcher, to see what is going on there.

In addition to looking at all of the data that you have collected and sorted through, you will need to do a few other parts as well. These are all meant to help the person who needs this information to read through it and see what is inside and what they can do with all of that data. It is the way that they are able to use the information to see what is going on, the complex relationships that are there, and so much more.

This means that we need to spend our time with some write-ups of the data, graphs, charts, and other ways to represent and show the data to those who need it the most. This will form one of the final steps that come with Data Analysis. These methods are designed in a manner to distill and refine the data so that the readers are then able to glean some of the interesting information from it, without having to go back through the raw data and figure out what is there all on their own.

Summarizing the data in these steps is going to be critical, and it needs to be done in a good and steady manner as well. Doing this is going to be critical to helping to support some of the arguments that are made with that data, as is presenting the data clearly and understandably. During this phase, we have to remember that it is not always possible that the person who needs that summary and who will use it to make some important decisions for the business will be data scientists, and they need it all written out in a simple and easy to understand this information. This is why the data has to be written out in a manner that is easy to understand and read through.

Often this is going to be done with some sort of data visualization. There are many choices of visuals that we can use and working with some kind of graph or chart is a good option as well. Laboring with the method that is the best for your needs and the data that we are using is going to be the best way to determine the visual that is going to be the best for you.

Many times, reading through information that is in a more graphical format is going to be easier to work with than just reading through the data and hoping it to work the best way possible. You could just have it all in a written form if you would like, but this is not going to be as easy to read through nor as efficient. To see some of those complex relationships quickly and efficiently, working with a visual is going to be one of the best options to choose.

Even though we need to spend some time working with a visual of the data to make it easier to work with and understand, it is fine to add in some of the raw data as the appendix, rather than just throwing it out. This allows the person who is going to work with that data regularly a chance to check your resources and your specific numbers and can help to bolster some of the results that you are getting overall.

If you are the one who is getting the results of the Data Analysis, make sure that when you get the conclusions and the summarized data from your data scientist that you go through and view them more critically. You should take the time to ask where the data comes from is going to be important, and you should also take some time to ask about the method of sampling that was used for all of this as well when the data were collected. Knowing the size of the sample is important as well.

This is going to allow you to really learn more about the data that you have and then will allow you to figure out if you can use the data, or if there may be some kind of bias that comes with it along the way. If the source of the data, or at least one of the sources, seems to have some kind of conflict that you are worried about, then this is going to pull your results into question, and you at least need to look it over.

Likewise, if you have some data that is gathered up from just a small sample or a sample that you worry is not random, then it is maybe not the best data to work with. The good news is that reputable researchers are going to have no problem providing you with the information that you need about the techniques of data gathering that they used and more so that you can make some important decisions about whether data is important or not.

There are so many great benefits that you are going to see when it is time to work with Data Analysis and using it for your own business. It can help you to learn more about your industry and the customers who are going to purchase your products. Those who can gather this information and learn how to use it correctly with the help of Data Analysis will be able to help you to gain a leg up on your competition and see some amazing results in the process.

Chapter 2 Why Python for Data Analysis?

The next thing that we need to spend some of our time on in this guidebook is the Python language. There are a lot of options that you can choose when working on your own Data Analysis, and bringing out all of these tools can make a big difference in how much information you can get out of your analysis. Nevertheless, if you want to pick a programming language that is easy to learn, has a lot of power, and can handle pretty much all of the tasks that you need to handle with Data Analysis and machine learning, then Python is the choice for you. Let's dive into the Python language a little bit and see how this language can be used to help us see some great results with our Data Analysis.

The Basics of the Python Language

To understand a bit more about how Python can help us out while handling a Data Analysis, we first need to take a look at what the Python language is all about. The Python language is an object-oriented programming language (or OOP language), that is designed with the user in mind, while still providing us with the power that we need, and the extensions and libraries, that will make Data Analysis and machine learning as easy to work with as possible.

There are many benefits that come with the Python coding language, and this is one of the reasons why so many people like to learn how to code with this language compared to other options. First, this coding language was designed with the beginner in mind. There are a lot of coding languages that are hard to learn, and only more advanced programmers, those who have spent years in this kind of field, can learn how to use them.

This is not the case when we talk about the Python language. This one has been designed to work well for beginners. Even if you have never done any coding in Python before you will find that this language is easy to catch on to, and you will be able to write some complex codes, even ones with enough power to handle machine learning and data science, in no time at all.

Even though the Python language is an easy one to learn how to use, there is still a lot of power that comes with this language as well. This language is designed to take on some of those harder projects, the ones that may need a little extra power behind them. For example, there are a lot of extensions that come with Python that can make it work with machine learning, a process where we teach a model or a computer how to make decisions on its own.

Due to the many benefits that come with the Python coding language, there are many people who are interested in learning more about it, and how to make it work for their needs. This happens in many large communities, throughout the world, of people sharing their ideas, asking for help, and offering any advice you may need. If you are a beginner who is just getting started with doing Data Analysis or any kind of Python programming at all, then this large community is going to be one of the best resources for you to use. It will help you to get all of your questions answered and ensures you are going to be able to finish your project, even if you get stuck on it for a bit.

This coding language also combines well with some of the other coding languages out there. While Python can do a lot of work on its own, when you combine it with some of the other libraries that are out there, sometimes it needs to be compatible with other languages as well. This is not a problem at all when it comes to Python, and you can add on any extension, and still write out the code in Python, knowing that it will be completely compatible with the library in question.

There are also a lot of different libraries that you can work with when it comes to the Python language. While we can see a lot of strong coding done with the traditional library of Python, sometimes adding some more functionality and capabilities can be the trick that is needed to get results. For example, there are many deep learning and machine learning libraries that connect with Python and can help this coding language take on some of the data science and Data Analysis projects that you want to use.

Python is also seen as an object-oriented programming language or an OOP language. This means that it is going to rely on classes and objects to help organize the information and keep things in line. The objects that we use, which are going to be based on real objects that we can find in our real world, are going to be placed in a class to pull out later when they are

needed in the code. This is much easier to work with than we see with the traditional coding languages of the past and ensures that all of the different parts of your code are going to stay exactly where you would like them

As we can see here, there is so much that the Python coding language is going to be able to do to help us with our Data Analysis. There are a lot of different features and capabilities that come with Python, and this makes it perfect for almost any action or project that we want to handle. When we combine it with some of the different libraries that are available, we can get some of these more complicated tasks done.

How Can Python Help with Data Analysis?

Now that we have had some time to discuss some of the benefits that come with the Python language and some of the parts that make up this coding language, it is now time for us to learn a few of the reasons why Python is the coding language to help out with all of the complexities and programs that we want to do with data science.

Looking back, we can see that Python has been pretty famous with data scientists for a long time. Although this language was not built to just specifically help out with data science, it is a language that has been accepted readily and implemented by data scientists for much of the work that they try to accomplish. Of course, we can imagine some of the obvious reasons why Python is one of the most famous programming languages, and why it works so well with data science, but some of the best benefits of using Python to help out with your data science model or project include:

Python is as simple as it gets. One of the best parts about learning how to work with the Python coding language is that even as someone who is completely new to programming and who has never done any work in this in the past, you can grasp the basics of it pretty quickly. This language, in particular, had two main ideas in mind when it was first started and these include readability and simplicity.

These features are pretty unique when we talk about coding languages, and they are often only going to apply to an object-oriented coding language, and one that has a tremendous amount of potential for problem-solving.

What all of this means is that, if you are a beginner to working with data science and with working on the Python language, then adding these two

together could be the key that you need to get started. They are both going to seem like simple processes when they work together, and yet you can get a ton done in a short amount of time. Even if you are more experienced with coding, you will find that Python data science is going to add a lot of depth to your resume, and can help you get those projects done.

The next benefit is that Python is fast and attractive. Apart from being as simple as possible, the code that we can write with Python is going to be leaner and much better looking than others. For example, the Python code takes up one-third of the volume that we see with code in Java, and one-fifth of the volume of code in C++, just to do the same task.

The use of the common expressions in code writing, rather than going with variable declarations and space in place of ugly brackets can also help the code in Python to look better. But, in addition to having the code look more attractive, it can help take some of the tediousness that comes in when learning a new coding language. This coding language can save a lot of time and is going to tax the brain of the data scientist a lot less, making working on some of the more complex tasks, like those of Data Analysis, much easier to handle overall.

Another benefit here is that the data formats are not going to be as worrisome with Python. Python can work with any kind of data format that you would like. It is possible for us to directly import SQL tables in the code without having to convert to a specific format or worry that our chosen format is not going to work. Besides, we can work with the Comma Separated Value documents and the web sourced JSON. Python request library can make it easy to import data from a lot of websites and will build up sets of data to help

The Python Data Analysis library known as Pandas is one of the best for helping us to handle all of the parts of not only our Data Analysis but also for the whole process of data science. Pandas can grab onto many data, without having to worry about lagging and other issues in the process. This is great news for the data scientist because it helps them to filter, sort, and quickly display their data.

Next on the list is that the Python library is quickly growing in demand. While the demand for professionals in the world of IT has seen a decline recently, at least compared to what it was in the past, the demand for

programmers who can work with Python is steadily on the rise. This is good news for those who still want to work in this field and are looking for their niche or their way to stand out when getting a new job.

Since Python has so many great benefits and has been able to prove itself as a great language for many things, including programs for data analytics and machine learning algorithms, many companies who are centered on data are going to be into those with Python skills. If you already have a strong on Python, you can work to ride the market that is out there right now.

Finally, we come back to the idea of the vibrant community that is available with the Python language. There are times when you will work on a project, and things are just not working the way that you had thought they would, or the way you had planned. Getting frustrated is one option, but it is not going to help you to find the solution.

The good news with this is that you will be able to use the vibrant community, and all of the programmers who are in this community, to provide you with a helping hand when you get stuck. The community that is around Python has grown so big and it includes members who are passionate and very active in these communities. For the newer programmer, this means that there is always an ample amount of material that is flowing on various websites, and one of these may have the solution that you are looking for when training your data.

Once you are able to get the data that you want to use and all of the libraries that work with as well, you can work on this community to see some of the results that you want. Never get stuck and just give up on a project or an idea that you have with your code, when you have that community of programmers and more, often many of whom have a lot of experience with Python, who will be able to help answer your questions and get that problem solved.

As we work through this guidebook, and you do more work with Python and Data Analysis, you will find that there are many libraries that are compatible with Python that can help to get the work done. These are all going to handle different algorithms and different tasks that you want to get done during your Data Analysis, so the ones that you will want to bring out may vary. There are many great choices that you can make, including TensorFlow, Pandas, NumPy, SciPy, and Scikit-Learn to name a few.

Sometimes these libraries work all on their own, and sometimes they need to be combined with another library so that they can draw features and functionalities from each other. When we can choose the right library to work with, and we learn how to make them into the model that we need, our Data Analysis is going to become more efficient overall.

While there may be other programming languages out there that are able to handle the work of Data Analysis, and that may be able to help us create the models that we need to see accurate insights and predictions based on the data, none of them are going to work as well as the Python library. Taking the time to explore this library and seeing what it can do for your Data Analysis process can be a winner when it comes to your business and using data science to succeed.

Chapter 3 The Steps of Data Analysis

With some of the ideas of a Data Analysis defined above to show us why this is so important, it is time for us to look at some of the steps that are so important to this process. When we know a bit more about some of the steps of Data Analysis, and what we can do with it, we are going to find why we should use this method of learning from Big Data and then ensuring that your business will be able to use this information to get further ahead of the competition.

For most businesses, there isn't going to be any problem with a lack of information. These businesses are going to suffer from having too much information to handle, and they are not certain what they are supposed to do with it. This over-amount of data is going to make it harder to come up with a clear decision based on the data, and that can be a problem as well. With so much data to go and sort through, we need to get something more from the data.

This means that we need to know that the data we have is right for the questions that we want to be answered. We need to know how we can draw some accurate conclusions from the data that we are working with. In addition, we need data that is going to be able to take on and inform our decision-making process.

In all, we need to make sure that we have the best kind of Data Analysis set up and ready to go. With the right process and tools set up for our Data Analysis, something that may have seemed like too much in the beginning and like an overwhelming amount of stuff to go through will then become a simple process that is clear and easy as possible.

To help us get all of this done, we need to go through some of the basic steps that are needed to use data to make better decisions overall. There are a lot of ways that we can divide this all up and make it work better for our needs, but we are going to divide this up into five steps that we are able to use and rely on to see some of the best results overall. Some of the steps that we can use to help make our Data Analysis more productive for better decision making in the company include:

- Defining your question
- Setting up clear measurement
- Collecting the data
- Analyzing the data
- Interpreting the results

Defining Your Question

The first step that we need to undertake when it comes to working on Data Analysis is to define the main question that we would like to handle. You should not just randomly work with the data that is on hand and hope that it shows you something. Because this is just going to get you lost and confused in the process. You need to have a clear picture of where you want to go and what you would like to learn from data, and then work from there.

In your Data Analysis, you need to start with the right questions. Questions are important, but we need to make sure that they are concise, measurable, and clear. Design the questions so that they can either qualify or disqualify some of the potential solutions that you are looking for on a specific problem or opportunity.

For example, you may want to start with a problem that you are able to clearly define. Maybe you are a government contractor, and you find that your costs are rising quite a bit. Because of this, you are no longer able to submit a competitive contract for some of the work that you are doing. You will want to go through with this and figure out how to deal with the business problem.

Setting up Clear Measurements

The next thing that we need to be able to do here is make sure that we can set up some clear priorities on your measurements. This is one that we will be able to break down into two subsets to help us out. The first part is that we need to decide what we want to measure and then the second thing we need to decide is how to measure it.

So, let’s start with the first part of deciding what we would like to measure. Going with the example of being a government contractor from before, we

would have to take this time to consider what kind of data is needed to answer that question we posed in the beginning. In this case, you would first need to gather up information on the number and the cost that you have for that current staff and the amount of time that these employees are going to spend on necessary business functions.

When we are going through and answering this kind of question, there are going to be many other questions that we need to answer as well to help us figure out the options that we need to take as well. For example, we would want to see if there are any employees we could do without if the staff is under-utilized in their current jobs and what you could do to help with some of this.

Then, we are going to get to the decision that we want to measure. When doing this, we need to make sure that we are considering any of the objections, the reasonable ones, of stakeholders and others who are working with the company. They may be worried about what would happen if you reduced the staff and then there was a big surge in demand shortly afterward, and you were not able to hire more people in the right amount of time.

Once we are done with that first step, it is time for us to make some decisions on how to measure. Thinking about how we can measure the data that we have is going to be just as important here, especially before we go through the phase of collecting data because the measuring process is going to either back up our analysis or discredits it later on. There are frequently questions of different questions that you are going to ask in this stage, but some of the most important ones to consider will include:

1. What is the time frame that we are looking at?
2. What is the unit of measure that we are relying on?
3. What factors are important to consider in all of this.

Collecting the Data

After we have had some time to go through and define our big problem and then work on the measurements that we are going to use, it is time for us to move on to collecting the data. With the question defined and the measurement priorities set, it is now time for us to go through and collect

the data. As we organize and collect the data, there are going to be many important points that we must keep in mind with this will include:

1. Before you collect some new data, you need to determine what information that we need to work with. We can look through some of the existing databases and some of the existing sources that we have on hand. You need to go through and collect some data first because it is simple and easier and can save a lot of money as well. We can move out to some other sources later, as well, if we need more information.
2. During this process, we also need to determine what naming system and file storing system we would like to use. This is going to make it easier for your team members to collaborate. This process is going to save some time and will prevent members of your team from wasting time and money by collecting the same kind of information more than once.
3. If you would like to gather up data through interviews and observations, then you need to go through and develop an interview template ahead of time. This is going to ensure that we are able to save some time and that some continuity goes on in this process as well.
4. Finally, we need to be able to keep the collected data that we have as organized as possible. We can work with a log that has the collection dates and add in the notes about sources as you would like. This should also include some data normalization that you may have performed as well. This is going to be important because it will validate the conclusions that you make down the road.

As you go through this process, we need to make sure that we are taking care of some of the data that we are working with. This means that we need to get it all organized, the values handled, and the duplicates took care of before you try to do some of the analysis that we want to do.

Since you are getting the data from different sources, and you are going to work with data that may be incomplete and not perfect along the way, we

need to be careful here. It is hard to know whether the data is going to be perfect or that you can use it the way that you want. In addition, if the information is in the wrong format, or it brings with it some errors or missing values, then it is going to be hard to go through and get the algorithm to work.

The first thing that we need to do with this is to make sure that the data is in the same format. Usually, the best way to handle this is for us to go through and put all of the information in a standardized database that we can look at. We can use this in our storage service and make sure that all of the data we bring is in just put through that database and ready to go.

From there, we are tenable to focus on dealing with some of the errors found in that data. We want to make sure that the outliers, the missing values, and the duplicates are gone. For the most part, the outliers are things that you are going to need to ignore and get rid of. If there are a number of these, and they all end up in the same spot in the process, then you should take a look at this to see what is going on and if this is new information that you should pay attention to. However, for most of these, you will find that they are not worth your time and should just be ignored.

From there, we are going to look at the missing values. When you get information from the real world, there are going to be times when there is a missing value in one of the parts that you are working with. You can choose to either delete these if there are just a few, or you can go through and replace these missing values with the mean of the other values that are in this column or row. It is up to you what you would like to do with these missing values to ensure that your information is as accurate as possible.

Finally, we need to deal with some of the duplicate values that are present in the set of data. If there are many duplicate values that show up in some of the data, then we are going to find that this will skew a lot of the numbers that we have and the results that we will get. This is why we need to take care of them, so we can see the true values that are inside of it.

You can go through here and figure out how much of the duplicate you would like to keep, and how much you would like to get rid of during your time. It is best to usually limit it as much as possible. Sometimes, this is keeping duplicates down to just two, and sometimes, it means only making

sure that all of the entries are only in there once. It is up to us to figure out which method to go with.

Analyzing the Data

After you have been able to collect the data, which is something that is going to take some time, we will then need to go through and start analyzing the data. We are going to start with this by manipulating the data, and there are some methods that we can use to make this happen. For example, we may decide to plot that information out and then find out what correlations are there or do a pivot table in Excel.

The pivot table is going to be useful to help us sort out and filter the data by different variables and then make it easier for us to calculate out what the minimum, maximum, mean, and even the standard deviation of the data that we are working with. This can give us a lot of great information that makes it easier for us to get going and to get a better understanding out of the data we have.

As we work on the process of manipulating the data that we have, you may find that you already have the exact data that we need. But, of course, life doesn't always work out as nicely as we would like, and you need to go out and collect some more data to work with or revise the original question that we are working with. Either way, you will find that this initial analysis of trends, variations, outliers, and correlations will help us to focus the Data Analysis that we do to better answer our questions and any of the objections that others may have in the process as well.

During this step, we will find that software and tools that work well with Data Analysis are going to be helpful through all of this. There are a lot of good packages that work for helping us to get through all of this and will ensure that we are able to get some of the work done that we want with statistical Data Analysis. However, in many cases, you will be just fine with using something as simple as Microsoft Excel when it is time to find a tool for decision making. You need to go through all of this stuff.

Interpreting the Results

After we have gone through the process of analyzing our data, which is going to take some time to accomplish and will often require some training and testing of your data through the algorithm to make sure that it works the way that you want, it is time to go through and interpret the results that we have. As we are going through this process and interpreting the analysis that we have, keep in mind that you are never going to be able to completely prove your hypothesis true at a 100% level. However, you can go through and reject the hypothesis. This means that no matter how much data you can collect, it is still possible for it to interfere with the results that we have.

As we are going through the process of interpreting our results, there are a few questions that we need to ask ourselves about data as well. Some of these major questions are going to include:

1. Is the data able to answer the original question that we had in the beginning and how?
2. Does the data help us to defend against the objects that were raised and how?
3. Are there any limitations to the conclusion that we have, or any angles that you have not considered?

If the interpretation that you have about the data can hold up under these questions and considerations, then it is likely that you are working with a productive conclusion. The only remaining step here is to use the results of your Data Analysis to help decide which course of action is the best one for you to work with.

By following the steps that are outlined in this chapter for your Data Analysis, you will find that it will help you to make some better decisions for your business. The main reason for this is that your choices are going to be backed up by data that has been collected robustly and analyzed as well. With practice, your Data Analysis can provide us with faster and more accurate results. This means, in the long run, that you are going to be able to make better and more informed decisions that help your company to run more efficiently than before.

Chapter 4 Python Libraries

We have talked about Data Analysis, and now it is time to take some of that information and put it to good use. You are probably interested in deep learning, and maybe even in making some of your Convolutional Neural Networks, but are wondering where you should start. The best step is to pick out the library that you want to use. However, this brings up another challenge because there are just so many coding libraries out there that you can choose from, and all of them have some amazing power and features behind them.

To start with, we are going to take a look at some of the best Python libraries that can help with deep learning. Other languages can help with things like machine learning and deep learning. But for most of the tasks that you want to do, especially if you are a beginner in Data Analysis and all of the processes that we have been talking about, then Python is going to be the choice for you. Even within Python, there are several libraries that you can choose from to get your deep learning work done. So, with that in mind, let's dive right in and see some of the best Python deep learning libraries that you can use for your Data Analysis.

Caffe

It is very hard to get started with a look at deep learning libraries through Python without spending some time talking about the Caffe library. It is likely that if you have done any research on deep learning at all, then you have heard about Caffe and what it can do for some of the projects and models that you want to create.

While Caffe is technically not going to be a Python library, it is going to provide us with some bindings into the Python language. We are going to use these bindings when it is time to deploy the network in the wild, rather than just when we try to train the model. The reason that we are going to include it in this chapter is that it is used pretty much everywhere and on all of the parts of a deep learning model that you need to create.

Theano

The next kind of library that we can work with is known as Theano. This one has helped to develop and work with a lot of the other deep learning libraries that we have that work with Python. In the same way that a programmer would not be able to have some options like scikit-image, scikit-learn, and SciPy without NumPy, the same thing can be said when we talk about Theano and some of the other higher-level abstractions and libraries that come with deep learning.

When we look at the core of this, Theano is going to be one of the Python libraries that not only helps out with deep learning, but also can be used to define, optimize, and evaluate a lot of mathematical expressions that will involve multi-dimensional arrays. Theano is going to accomplish this because it is tightly integrated with the NumPy library, and it keeps its use of GPU pretty transparent overall.

While you can use the Theano library to help build up some deep learning networks, this one is often seen as the building blocks of these neural networks, just like how the NumPy library is going to serve as the building blocks when we work on scientific computing. Most of the other libraries that we will talk about as we progress through all of this are going to wrap around the Theano library, which makes it more accessible and convenient than some of the other options.

TensorFlow

Similar to what we can find with the Theano library, TensorFlow is going to be an option that is open-sourced and can work with numerical computation with the help of a data flow graph. This one was originally developed to be used with research on the Google Brain Team within Google's Machine Intelligence organization. In addition, this library, since that time, has turned into an open-sourced option so that the general public can use it for their deep learning and data science needs.

One of the biggest benefits that we are going to see with the TensorFlow library, compared to what we see with Theano, is that it is able to work with distributed computing. This is particularly true when we look at multiple-GPUs for our project, though Theano is working on improving this one as well.

Keras

Many programmers find that they love working with the Keras library when it comes to performing models and other tasks with deep learning. Keras is seen as a modular neural network library that is more minimalistic than some of the others that we talk about. This one can use either TensorFlow or Theano as the backend, so you can choose the one that works the best for any needs you have. The primary goal that comes with this library is that you should be able to experiment on your models quickly and get from the idea that you have over to the result as fast as possible.

Many programmers like this library because the networks that you architect are going to feel almost natural and easy, even as a beginner. It is going to include some of the best algorithms out there for optimizers, normalization, and even activation layers, so this is a great one to use if your process includes these.

Besides, if you want to spend some time developing your CNNs, then Keras is a great option to work with. Keras is set up to place a heavy focus on these kinds of neural networks, which can be valuable when you are working from the perspective of computer vision. Keras also allows us to construct both sequence-based networks, which means that the input is going to be able to flow linearly throughout that network and the graph-based network, which is where the inputs can skip over some of the layers if needed, only to be concatenated at a later point. This is going to make it easier for us to implement more complex network architectures.

One thing to note about this Python library is that it is not going to support some of the multi-GPU environments if you would like to train a network in parallel. If this is something that you want to do, then you may need to choose another library that you want to use. However, for some of the work that you want to do, this may not be a big issue.

If you want to get your network trained as fast as possible, working with a library like MXNet may be a better choice. But if you are looking to tune your hyperparameters, then you may want to work with the capability of Keras to set up four independent experiments and then evaluate how the results are similar or different between each of these.

Sklearn-Theano

There are going to be times when working with deep learning when you will want to train a CNN end-to-end. And then there are times when this is not needed. Instead, when this is not useful, you can treat your CNN as the feature extractor. This is going to be the most useful with some situations you may encounter where there is just not enough data to train the CNN from scratch. So, with this one, just pass your input images through a popular pre-trained architecture that can include some options like VGGNet, AlexNet, and OverFeat. You can then use these pre-trained options and extract features from the layer that you want, usually the FC layers.

Nolearn

A good library for you to work with is the Nolearn library. This is a good one to help out with some initial GPU experiments, especially with a MacBook Pro. It is also an excellent library to assist with performing some deep learning on an Amazon EC2 GPU instance.

While Keras wraps TensorFlow and Theano into a more user-friendly API, you will find that the Nolearn library will be able to do the same, but it will do this with the Lasagna library. Also, all of the code that we find with Nolearn is going to be compatible with Scikit-Learn, which is a big bonus for a lot of the projects that you want to work with.

Digits

The first thing to notice with this library is that it isn’t considered a true deep learning library. Although it is written out in Python and it stands for Deep Learning GPU Training System. The reason for this is because this library is more of a web application that can be used for training some of the models of deep learning that you create with the help of Caffe. You could work with the source code a bit to work with a backend other than Caffe, but this is a lot of extra work in the process. In addition, since the Caffe library is pretty good at what it does, and can help with a lot of the deep learning tasks that you want to accomplish, it is not worth your time.

If you have ever spent some time working with the Caffe library in the past, you can already attest to the fact that it is tedious to define your .prototxt files, generate the set of data for the image, run the network, and babysit the

network training with the terminal that you are provided. The good news here is that the DIGITS library aims to fix all of this by allowing you to complete many of these tasks, if not all of these tasks, just from your browser. So, it may not be a deep learning library per se, but it does come into use when you struggle with the Caffe library.

In addition to all of the benefits above, the interface that the user gets to interact with is seen as excellent. This is because it can provide us with some valuable statistics and graphs to help you rain your model more effectively. You can also easily visualize some of the activation layers of the network to help with various inputs as needed.

And finally, another benefit that is possible with this library is that if you come in with a specific image that you want to test, you have a few options on how to get this done. The first choice is to upload the image over to the DIGITS server. Alternatively, you can enter the URL that comes with the image, and then the model you make with Caffe will automatically be able to classify the image and display the results that you want in the browser.

Python is one of the best coding languages available for helping with tasks like deep learning, machine learning, and even with the topic of artificial intelligence, which encompasses both of the other two ideas. Other languages can handle the deep learning that we have been talking about, but none are going to be as effective, as powerful, have as many options, or be designed for a beginner in the way that Python can.

This is why we have focused our attention on the Python language and some of the best libraries that we can choose to help with a variety of deep learning tasks. Each of these libraries can come on board with your project and will provide a unique set of functions and skills to get the job done. Look through some of these libraries and see which one is going to be just right for your Data Analysis and for providing you with great insights while completing deep learning.

Chapter 5 Pandas

Pandas are built on NumPy and they are meant to be used together. This makes it extremely easy to extract arrays from the data frames. Once these arrays are extracted, they can be turned into data frames themselves. Let's take a look at an example:

In: import pandas as pd

import numpy as np

marketing_filename = 'regression-datasets-marketing.csv'

```
marketing = pd.read_csv(marketing _filename, header=None)
```

In this phase, we are uploading data to a data frame. Next, we're going to use the "values" method to extract an array that is of the same type as those contained inside the data frame.

```
In: marketing_array = marketing.values

marketing_array.dtype

Out: dtype('float64')
```

We can see that we have a float type array. You can anticipate the type of the array by first using the "dtype" method. This will establish which types are being used by the specified data frame object. Do this before extracting the array. This is how this operation would look:

In: marketing.dtypes

Out: 0float64

1int64

2float64

3int64

4float64

5float64

6float64

7float64

8int64

9int64

10int64

11float64

12float64

13float64

dtype: object

Matrix Operations

This includes matrix calculations, such as matrix to matrix multiplication. Let’s create a two-dimensional array.

This is a two dimensional array of numbers from 0 to 24. Next, we will declare a vector of coefficients and a column that will stack the vector and its reverse. Here’s what it would look like:

In: coefs = np.array([1., 0.5, 0.5, 0.5, 0.5])

coefs_matrix = np.column_stack((coefs,coefs[::-1]))

print (coefs_matrix)

Out:

[[1. 0.5]

[0.50.5]

[0.50.5]

[0.50.5]

[0.51.]]

Now we can perform the multiplication. Here’s an example of multiplying the array with the vector:

In: np.dot(M,coefs)

Out: array([5.,20.,35.,50.,65.])

Here’s an example of multiplication between the array and the coefficient vectors:

In: np.dot(M,coefs_matrix)

Out: array([[5.,7.],

[20.,22.],

[35.,37.],

[50.,52.],

[65.,67.]])

In both of these multiplication operations, we used the “np.dot” function in order to achieve them. Next up, let’s discuss slicing and indexing.

Slicing and Indexing

Indexing is great for viewing the nd-array by sending an instruction to visualize the slice of columns and rows or the index.

Let’s start by creating a 10x10 array. It will initially be two-dimensional.

In: import numpy as np

M = np.arange(100, dtype=int).reshape(10,10)

Next let’s extract the rows from 2 to 8, but only the ones that are evenly numbered.

In: M[2:9:2,:]

Out: array([[20, 21, 22, 23, 24, 25, 26, 27, 28, 29],

[40, 41, 42, 43, 44, 45, 46, 47, 48, 49],

[60, 61, 62, 63, 64, 65, 66, 67, 68, 69],

[80, 81, 82, 83, 84, 85, 86, 87, 88, 89]])

Now let’s extract the column, but only the ones from index 5.

In: M[2:9:2,5:]

Out: array([[25, 26, 27, 28, 29],

[45, 46, 47, 48, 49],

[65, 66, 67, 68, 69],

[85, 86, 87, 88, 89]])

We successfully sliced the rows and the columns. But, what happens if we try a negative index? Doing so would reverse the array. Here's how our previous array would look when using a negative index.

In: M[2:9:2,5::-1]

Out: array([[25, 24, 23, 22, 21, 20],

[45, 44, 43, 42, 41, 40],

[65, 64, 63, 62, 61, 60],

[85, 84, 83, 82, 81, 80]])

However, keep in mind that this process is only a way of viewing the data. If you want to use these views further by creating new data, you cannot make any modifications to the original arrays. If you do, it can lead to some negative side effects. In that case, you want to use the "copy" method. This will create a copy of the array, which you can modify however you wish. Here's the code line for the copy method:

In: N = M[2:9:2,5:].copy()

Chapter 6 Jupyter Notebook

Getting started with Jupyter Notebook (IPython)

The Jupyter Note pad is an open-source web application that permits you to produce and share files that contain live code, formulas, visualizations and narrative text. Utilizes consist of information cleansing and change, mathematical simulation, analytical modeling, information visualization, artificial intelligence, and far more.

Jupyter has assistance for over 40 various shows languages and Python is among them. Python is a requirement (Python 3.3 or higher, or Python 2.7) for setting up the Jupyter Notebook itself.

Setting up Jupyter utilizing Anaconda

Set up Python and Jupyter utilizing the Anaconda Distribution, which includes Python, the Jupyter Notebook, and other typically utilized bundles for clinical computing and information science. You can download Anaconda's newest Python3 variation.

Now, set up the downloaded variation of Anaconda.

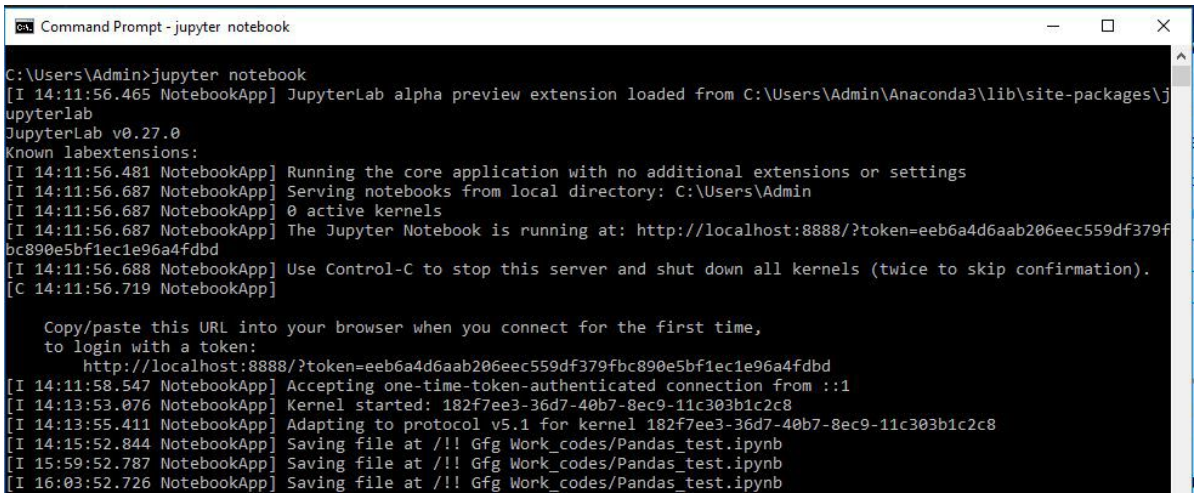
Setting up Jupyter Notebook utilizing PIP:

- python3 -m pip install --upgrade pip
- python3 -m pip install jupyter

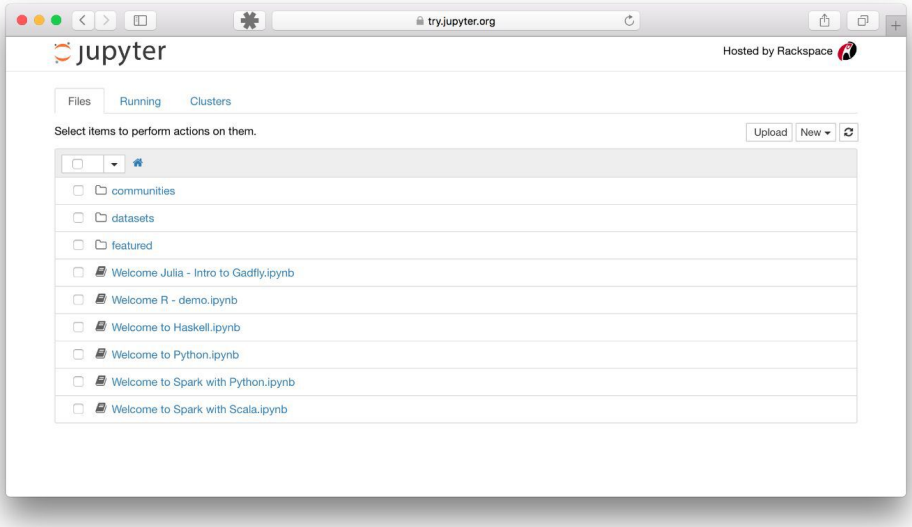
Command to run the Jupyter notebook:

jupyter notebook

This will print some details about the note pad server in your terminal, consisting of the URL of the web application (by default, http://localhost:8888) and after that, open your default Web Internet browser to this URL.

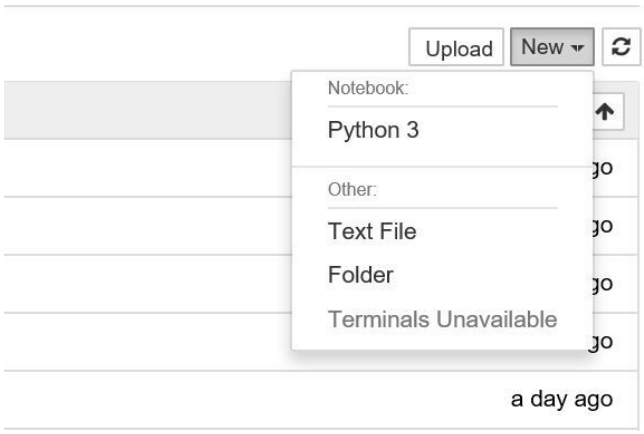


When the Notepad opens in your Internet browser, you will see the Notebook Dashboard, which will reveal a list of the notepads, files, and subdirectories in the directory site where the Notepad server was started. The majority of the time, you will want to begin a Notepad server in the greatest level directory site consisting of notepads. Typically, this will be your house directory site.



Create a new Notebook

Now on the control panel, you can see a brand-new button on the top right corner. Click it to open a drop-down list and after that, if you'll click Python3, it will open a brand-new notebook.



Few Useful Commands

Command to open a notebook in the currently running notebook server.

```
jupyter notebook notebook_name.ipynb
```

By default, the note pad server begins on port 8888. If port 8888 is not available or in usage, the Notepad server browses the next readily available port.

```
jupyter note pad-- port 9999
```

Command to begin the Notepad server without opening a Web Internet browser:

```
jupyter notebook --no-browser
```

The notebook server provides help messages for other command-line arguments using the –help flag:

```
jupyter notebook –help
```

Running your First code in Jupyter

Action #1: After effectively setting up Jupyter compose 'jupyter note pad' in the terminal/command timely. This will open a brand-new Notepad server on your Web Internet browser.

Action #2: On the leading left corner, click the brand-new button and choose python3. This will open a brand-new Notepad tab in your Web browser where you can begin to compose your very first code.

Action #3: Press Click or get in on the very first cell in your Notepad to enter into the edit mode.

Action #4: Now you are free to compose any code.

Action #5: You can run your code by pushing Shift + Enter or the run button offered at the top. An example code is provided listed below:

```
In [6]: # Define a function for addition
        def add(a , b):
            return a + b

        val = add(12, 13)

        val

Out[6]: 25
```

Jupyter Notebook Tips and Tricks

Python is a fantastic language for doing information analysis, mainly because of the great environment of data-centric python bundles. What makes information analysis in Python more efficient and effective is Jupyter Notepad or what was previously called the IPython Notepad.

In this area, we are going to go over some great functions of the Jupyter Notepad, which increases the efficiency and effectiveness of the information expert. The Jupyter Notepad extends the console-based technique to interactive computing in qualitatively brand-new instructions, offering a web-based application appropriate for recording the entire calculation procedure: establishing, recording, and carrying out code, along with interacting the outcomes. In a nutshell, it is a total plan.

Let's see some functions of the Jupyter Notepad, which becomes extremely convenient while doing information analysis.

%%timeit and %%time

It's not an unusual thing for an information researcher that while doing information analysis, they have more than one service for the provided issue. They wish to pick the very best technique, which accomplishes the job in the minimum quantity of time. Jupyter Notepad supplies an

extremely effective method to inspect the running time of a specific block of code.

We can utilize the %% time command to inspect the running time of a specific cell. Let's see the time takes to carry out the code pointed out listed below.

```
# For capturing the execution time

%%time

# Find the squares of a number in the

# range from 0 to 14

for x in range(15):

square = x**2


print(square)
```

Output:

```
0
1
4
9
16
25
36
49
64
81
100
121
144
169
196
Wall time: 999 µs
```

We can likewise utilize the command %%timeit to run the provided bit of code over some variety of times to discover the typical run time for that piece of code.

Commenting/Uncommenting a block of code:

While dealing with codes, we typically include brand-new lines of code and comment out the old pieces of code for enhancing the efficiency or to debug it. Jupyter Notepad supplies an extremely effective method to accomplish the same – to comment out a block of code.

We require to choose all those lines which we want to comment out, as shown in the following picture:

```
:
1  df = pd.DataFrame({'Date':['10/2/2011','11/2/2011','12/2/2011','13/2/2011'],
2                     'Event':['Music','Poetry','Theatre','Comedy'],
3                     'Cost':[10000,5000,15000,2000]})
4
5  df.index = ['A', 'B', 'A', 'D']
6
7  print(df)
8
9  df2 = pd.DataFrame({'Cost1':[10025,5700,2415,1800],
10                     'Cost2':[1000,500,150,20],
11                     'Cost3':[10000,5000,15000,2000]})
12
13 print(df.columns)
```

Next, on a Windows computer system, we need to push the ctrl +/ crucial mix to comment out the highlighted part of the code.

```
:
1  df = pd.DataFrame({'Date':['10/2/2011','11/2/2011','12/2/2011','13/2/2011'],
2                     'Event':['Music','Poetry','Theatre','Comedy'],
3                     'Cost':[10000,5000,15000,2000]})
4
5  df.index = ['A', 'B', 'A', 'D']
6
7  # print(df)
8
9  # df2 = pd.DataFrame({'Cost1':[10025,5700,2415,1800],
10 #                     'Cost2':[1000,500,150,20],
11 #                     'Cost3':[10000,5000,15000,2000]})
12
13 print(df.columns)
```

This does conserve a great deal of time for the information expert.

Next, on a Windows computer system, we need to push the ctrl +/ crucial mix to comment out the highlighted part of the code.

```
df = pd.DataFrame({'Date':['10/2/2011','11/2/2011','12/2/2011','13/2/2011'],
                   'Event':['Music','Poetry','Theatre','Comedy'],
                   'Cost':[10000,5000,15000,2000]})

df.index = ['A', 'B', 'A', 'D']

print(df)

df2 = pd.DataFrame({'Cost1':[10025,5700,2415,1800],
                   'Cost2':[1000,500,150,20],
                   'Cost3':[10000,5000,15000,2000]})

print(df.columns)
```

Chapter 7 The PyTorch Library

The next library that we need to look at is known as PyTorch. This is going to be a Python-based package that works for scientific computing that is going to rely on the power that it can receive from graphics processing units. This library is also going to be one of the most common and the preferred deep learning platforms for research to provide us with maximum flexibility and a lot of speed in the process.

There are plenty of benefits that come with this kind of library. And it is known for providing two of the most high-level features out of all the other deep learning libraries. These will include tensor computation with the support of a strong GPU acceleration, along with being able to build up the deep neural networks on an autograd-system that is tape-based.

There are many different libraries through Python that can help us work with a lot of artificial intelligence and deep learning projects that we want to work with. In addition, the PyTorch library is one of these. One of the key reasons that this library is so successful is because it is completely Pythonic and one that can take some of the models that you want to build with a neural network almost effortlessly. This is a newer deep learning library to work with, but there is a lot of momentum in this field as well.

The Beginnings of PyTorch

As we mentioned above, PyTorch is one of the newest libraries out there that works with Python and can help with deep learning. Even though it was just released in January 2016, it has become one of the go-to libraries that data scientists like to work with, mainly because it can make it easy to build up complex neural networks. This is perfect for countless beginners who haven’t been able to work with these neural networks at all in the past. They can work with PyTorch and make their network in no time at all, even with a limited amount of coding experience.

The creators of this Python library envisioned that this library would be imperative when they wanted to run plentiful numerical computations as quickly as possible. This is one of the ideal methodologies that also fits in perfectly with the programming style that we see with Python. This library,

along with the Python library, as allowed debuggers of neural networks, machine learning developers, and deep learning scientists to not only run but also to test parts of their code in real-time. This is great news because it means that these professionals no longer have to wait for the entire code to complete and execute before they can check out whether this code works or if they need to fix certain parts.

In addition to some of the functionality that comes with the PyTorch library, remember that you can extend out some of the functionalities of this library by adding in other Python packages. Python packages like Cython, SciPy, and NumPy all work well with PyTorch as well.

Even with these benefits, we still may have some questions about why the PyTorch library is so special, and why we may want to use this when it is time to build up the needed models for deep learning. The answer with this is simple, mainly that PyTorch is going to be seen as a dynamic library. This means that the library is flexible and you can use it with any requirements and changes that you would like. It is so good at doing this job that it is being used by developers in artificial intelligence, students and researchers in many industries. In fact, in a Kaggle competition, this library was used by almost all of the individuals who finished in the top ten.

While there are multiplied benefits that can come with the PyTorch library, we need to start with some of the highlights of why professionals of all sorts love this language so much. Some of these include:

1. The interface is simple to use. The PyTorch interface is going to offer us an API that is easy to use. This means that we will find it simple to operate and run as we do with Python.
2. It is Pythonic in nature. This library, since it is considered Pythonic, will smoothly integrate to work with the Python data science stack. Those who do not want to work with other coding languages along the way and want to just stick with the basics and some of the power of Python, will be able to do so with this library. You will get the leverage of using all of the functionalities and services that are offered through the environment of Python.

3. Computational graphs. Another highlight that comes with the PyTorch library is that it is going to provide us with the platform with some dynamic computational graphs. This means that you can change these graphs up even during runtime. This is going to be useful any time that you need to work on some graphs and you are not sure how much memory you need to use while creating this model for a neural network.

The Community for PyTorch

The next thing that we need to look at here is some of the community that comes with the PyTorch library. Because of all the benefits that come with PyTorch, we can see that the community of developers and other professionals is growing daily. In just a few years, this library has shown many developments and has even led this library to be cited in many research papers and groups. And when it comes to artificial intelligence and models of deep learning, PyTorch is starting to become one of the main libraries to work with.

One of the interesting things that come with PyTorch is that it is still in the early-release beta. But because so many programmers are adopting the framework for deep learning already, and at such a brisk pace, it already shows the power and the potential that comes with it and how the community is likely to continue growing. For example, even though we are still on the beta release with PyTorch, there are currently 741 contributors just on the GitHub repository right now. This means that more than 700 people are working to enhance and add some improvements to the functionalities of PyTorch that are already there.

Think of how amazing this is! PyTorch is not technically released yet and is still in the early stages. However, there has been so much buzz around this deep learning library, and so many programmers have been using it for deep learning and artificial intelligence, that there are already a ton of contributors who are working on adding some more functionality and improvements to this library for others to work with.

PyTorch is not going to limit the specific applications that we are working with because of the modular design and the flexibility that comes with it. It

has seen a heavy amount of use by some of the leading tech giants, and you may even recognize some of the names. Those who have already started to work with PyTorch to improve their deep learning models will include Uber, NVIDIA, Twitter, and Facebook. This library has also been used in a lot of domains for research, including neural networks, image recognition, translation, and NLP, among other key areas.

Why Use PyTorch with the Data Analysis

Anyone who is working with the field of data science, Data Analysis, artificial intelligence, or deep learning has likely spent some time working with the TensorFlow library that we also talked about in this guidebook. TensorFlow may be the most popular library from Google, but because of the PyTorch framework for deep learning, we can find that this library is able to solve a few new problems when it comes to research work that these professionals want to fix.

It is often believed that PyTorch is now the biggest competitor out there to TensorFlow when it comes to handling data, and it is going to be one of the best and most favorite artificial intelligence and deep learning library when it comes to the community of research. There are many reasons for this happening, and we will talk about some of these below:

First, we will notice that the dynamic computational graphs are going to be popular among researchers. This library is going to avoid some of the static graphs that can be used in other frameworks from TensorFlow. This allows researchers and developers to change up how the network is going to behave at the last minute. Some of those who are adopting this library will like it because these graphs are more intuitive to learn when we compare it to what TensorFlow can do.

The second benefit is that this one comes with a different kind of backend support. PyTorch is going to use a different backend based on what you are doing. The GPU, CPU, and other functional features will all come with a different backend rather than focusing on just one backend to handle all of these. For example, we are going to see the THC for our GPU, and TH for CPU. Being able to use separate backends can make it easier for us to deploy this library through a variety of constrained systems.

The imperative style is another benefit of working with this kind of library. This means that when we work with this library, it is easy to use and very intuitive. When you execute a line of code, it is going to get executed just as you want, and you are able to work with some tracking that is in real-time. This allows the programmer to keep track of how the models for neural networks are doing. Because of the excellent architecture that comes with this, and the lean and fast approach, it has been able to increase some of the overall adoptions that we are seeing with this library throughout the communities of programmers.

Another benefit that we are going to enjoy when it comes to working with PyTorch is that it is easy to extend. This library, in particular, is integrated to work well with the code for C++ and it is going to share a bit of the backend with this language when we work on our framework for deep learning. This means that a programmer is going to be able to not just use Python for the CPU and GPU, but it can also add in the extension of the API using the C or the C++ languages. This means that we can extend out the usage of PyTorch for some of the new and experimental cases, which can make it even better when we want to do some research with it.

Finally, the last benefit that we are going to focus on here is that PyTorch is going to be seen as a Python approach library. This is because the library is a native Python package just by the way that it is designed. The functionalities that come with this are built as classes in Python, which means that all of the code that you write here can integrate seamlessly with the modules and packages of Python.

Similar to what we see with NumPy, this Python-based library is going to enable us to work on a tensor that is GPU-accelerated, with computations that provide us with some rich options for APIs while applying a neural network. PyTorch is going to provide us with the research framework that we need from one end to another, which will come with most of the different building blocks that we need to carry out the research of deep learning on a day to day basis. And we also notice that this library is going to provide us with high-level neural network modules because it can work with an API that is similar to the Keras library as well.

PyTorch 1.0 – How To Go From Research to Production

During this chapter, we have spent some time discussing a lot of the strengths that we can see with the PyTorch library, and how these will help to make lots of researchers and data scientists run to it as their go-to library. However, there are a few downsides that come with this library and one of these includes that it has been lacking when it comes to supporting production. However, due to some of the improvements and changes that will happen with PyTorch, it is expected that this is something that will change soon.

The next release of PyTorch, which is known as PyTorch 10, is already expected to be a big release that is meant to overcome some of the biggest challenges that researchers, programmers, and developers face in production. This is the newest iteration of the whole framework, and it is expected to combine with the Python-based Caffe2, allowing for deep learning researchers and machine learning developers to move from research to production. And the point of doing this is to allow the process to happen in a manner that is hassle-free and without the programmer needing to deal with challenges that show up in migration.

The new version 1.0 is meant to help to unify the research and production capabilities in one framework, rather than doing it in two parts, making things easier and avoiding some of the missed values and complications that happen when you try to merge two parts. This allows us with more performance optimization and the flexibility that we need to complete both research and production.

This newer version is going to promise us a lot of help with handling the tasks that come up. Many of these tasks are going to make it more efficient to run your models of deep learning on a much larger scale. Along with the support of production, remember that PyTorch 10 will have countless of improvements with optimization and usability.

With the help of the PyTorch 1.0 library, we are going to be able to take the existing code and continue to work on it as-is. The existing API is not going to change, so that makes it easier for those who have already been able to create some coding and programs with the old API. To help make sense of the progress that is going to happen with the PyTorch library, you can look at the PyTorch website to help out.

As we can see with all of the information that we explored in this chapter, PyTorch is already seen as a compelling player when it comes to the various processes that we can do with artificial intelligence and deep learning. Being able to exploit all of the unique parts that come with this, and seeing that it is going to work as a research first library can be an important part of our Data Analysis overall.

The PyTorch library is able to overcome some of the challenges that it has and can provide us with all of the power and the performance that is necessary to get the job done. If you are a student, a researcher, or a mathematician who is inclined to work with some of the models of deep learning, then the PyTorch library is going to be a great option as a framework for deep learning to help you get started.

Applications to Data Science

Table of Contents

Chapter 1 Data Visualization and Matplotlib

At some point in your journey to working with all of that data and trying to analyze it, you are going to decide that it is time actually to see the data and visualize it. Sure, we could put all of that information into a table or write it out in long and boring paragraphs, but this is going to make it hard to focus and see the comparisons and more that come with that information. One of the best ways to do this is to take that information, and after it has been analyzed accurately, we can use data visualization to get this to work for our needs.

The good news is that the Matplotlib library, which is an extension of NumPy and SciPy, can help us to create all of them. Whether you want to look at your data as a line graph, a pie chart, a histogram or in some other form, the Matplotlib library is there to help.

The first thing that we are going to try and explore when it comes to this topic is data visualization. We need to have a better idea of what data visualization is all about, how it works, why we would want to use it, and so on. When we can put all of these parts together, it becomes much easier to take all of that data we have been collecting, and then actually see it in a way that is easy to look over and make smart business decisions with.

Data visualization is going to be the presentation of quantitative information in a form that is more graphical. In other words, you can use data visualization to take a set of data, whether it is small or large, and turn it into visuals that are much easier for the brain to understand and to process. It is a pretty common occurrence in your daily life, but you may recognize them the most from graphs and charts that help you to look over the information a bit better. A combination of more than one of these visuals and even some added information in the mix are going to be known as infographics.

You will find that there are a lot of benefits that come with using this kind of data visualization. It can be used to help out with trends and facts that may not be known, and which can be hidden way down deep in the information that you have, especially in some of the larger sets of data. You

can see these visualizations inline charts that help to show us the change in something over time, column and bar charts that can help you to make comparisons and see the relationship between two or more things, and even pie charts to show how much of a whole something takes up.

These are just a few of the examples of the data visualization that you may encounter as you do your work.

The next thing that we need to take a look at is what is going to make some good data visualization? These are going to be created when we can add together design, data science, and communication all in one. Data visualization, when it is done right, can offer us some key insights into the sets of data that are complicated, and this is done in a manner that makes them more meaningful for the ones who are using them, and more intuitive overall.

There are many companies that are going to be able to benefit from this kind of data visualization, and it is a tool that you do not want to overlook. Any time that you are trying to analyze your data, and you want to make sure that it matches up well and that you are fully comprehending all of the information that is being shared, it is a good idea to look at some of the plots later on and decide which ones can talk about your data the best, and which one you should use.

You may be able to get all of the information that you need out of your data, and you may be able to complete your Data Analysis, without needing to have these charts. However, when you have a lot of data, and you don’t want to miss anything, and you don’t have time to go through it all, these visuals will help. The best kinds of data visualizations are going to include plenty of complex ideas that can be communicated in a manner that has efficiency, precision, and clarity, to name a few.

To help you to craft a good kind of data visualization, you need to make sure that your data is clean, well-sourced, and complete. This is a process that we did earlier in this guidebook, so your data should be prepared by this time. Once you are ready and the data is prepared, it is time to look over some of the charts and plots that are available to use. This is sometimes hard and a bit tricky based on the kind of information that you are working with at the time. You have to go with the chart type that seems to work best for the data that you have available.

After you have done some research and figured out which of the types of charts is the best, it is time to go through and design, as well as customize, that chosen visual to work the best for you. Always keep the graphs and charts as simple as possible because these often make the best types of graphs. You do not want to take the time adding in a bunch of elements that are not needed and will simply distract us from the data.

At this point, the visualization should be complete. You have picked out the one that you want to use, after sorting through and cleaning your chosen data of course, and then picked out the right chart to use, bringing in the code that goes with it to get the best results. Now that this part is done, it is time to take that visualization, publish and share it with others as well.

Why is it important to use data visualization

With the information above in mind, it is time to look at some of the benefits of using data visualization, and why so many people like to work with this. While it is possible to do the analysis and more on your own, without having any of the graphs and other visuals to go along, this is often a poor way to make decisions and does not ensure that you know what is going on with the data in front of you, or that you will see the full amount of information and trends that are presented. In addition to this, some of the other reasons that data analysts like to work with these kinds of visualizations include:

It helps them to make better decisions. Today more than any other time in history, companies have decided to look at a variety of data tools, including data visualizations, to ask the right questions and make the best decisions for them. Emerging computer technologies and some user-friendly software programs have made it a bit easier to learn more about your company and to ensure that you are making the best decisions for your business, driven with sound data behind it.

The strong emphasis on things like KPIs, data dashboards, and performance metrics is already showing us the importance of taking all of the data the company has collected and then measuring and monitoring it. Some of the best quantitative information that a business may already be measuring right now, and that they could put to good use after the analysis is going to include the amount of market share the company has, the expenses of each

department, the revenue that is received by quarter, and even the units or product that are sold by the company.

The next benefit of working with this kind of data visualization is that it can help to tell us a story with a lot of meaning behind it. Data visualizations, as well as other informational graphics, have become an essential tool when we are looking at some of the work the mainstream media is doing. Data journalism is a field that is on the rise, and many journalists are consistently relying on quality visualization tools to make sure they can tell the stories about what is happening in the world around you.

This is something that is taking on steam in recent years. Many of the most well-respected institutions out there have been able to fully embrace the idea of news that is driven by data, including places like The Washington Post, The New York Times, CNN, and The Economist.

Marketers are also able to come onto the scene and benefit from the combination of emotional storytelling and quality data that they have available to them all of the time. You will find that a good marketer is able to make decisions that are driven by data daily, but sharing this information with their customers is going to need an approach that is a bit different. This approach needs to be one that can touch on both the emotional and the intelligent all at the same time. You will find that the visuals from the data can make sure that the marketers are able to get their message out there, using heart and statistics.

Data literacy is another thing that can push us towards working with this kind of data visualization. Being able to read and then understand the data visualization has become something that is a requirement in our modern world. Because a lot of the resources and tools that go with these visuals are readily available in our modern world, it is expected that professionals, even if they are not in the technical world, can look at these visuals and gain the right data from them.

This is why increasing the amount of data literacy that is found around the world is going to be one of the biggest pillars that we are going to see when it comes to a lot of data visualization companies and more. To help people in business to make better decisions, it is important to have the right information and the right tools and the data visualization graphs are going to be the key to getting that done.

The history that comes with data visualization

Also, we can turn to Florence Nightingale to help here as well. She was most known for being a nurse during the Crimean War, but along with that, she was a data journalist who was known for her rose or coxcomb diagrams. These were a type of revolutionary chart that helped here to get better conditions in the hospitals, which, in the end, helped to save many lives of the soldiers who were there

In addition, one of the most well-known data visualizations that we can find is going to come from Charles Joseph Minard. Minard is known as a civil engineer from France who was known for his representation of numerical data on the maps that he used. In particular, he is the one who is famous for the work he did on the map that showed Napoleon's Russian campaign of 1812, displaying the dramatic loss of his army when they were trying to make an advance in Moscow, along with some of the retreat that followed.

Why we should use data visualization

Before exploring further what the matplotlib library can do (so that we have more ways to represent our data), we need to end this section with a look at data visualization and why we should use it in the first place. Some of the different reasons why you would want to work with data visualization include:

1. *It can take the data that you have and makes it easier for you to remember and understand rather than just reading through the information and hoping it makes sense.*
2. *It is going to give you the ability to discover facts that are not known to you, trends, and even outliers in the information that could be useful as well.*
3. *It makes your life a bit easier because it helps you to visualize relationships and patterns quickly and effectively.*
4. *It ensures that you can ask questions in a better way and make the best decisions for your business.*

As you can see, there are many different benefits when it comes to working with data visualization and all of the different things that you can do with it. It is worth your time to learn a few of the different parts of data

visualization and what all it can do for you. If you have plenty of data or just a little bit, these visuals are going to be the perfect tools to make sure that you can get it done.

What is matplotlib?

With the rest of the information in mind from above, we know now how important it is to work with data visualization and to have a method in place that helps us to understand what is going on in all of the data that we collected earlier. This ensures that we are going to be set to go and that the whole analysis works the way that we would like.

This brings up another question that we have to consider, though. We need to know what methods we can use that are going to help us to create some of these charts and graphs and the other visuals that we choose to use. Your business likely has a ton of information in place, and you want to make sure that you are not only choosing the right kind of visualization but that you are also able to put it all together and make the right visual, and this is where matplotlib is going to come into play.

The idea that comes with matplotlib is that a picture is going to be worth a thousand words. Luckily, this kind of library is not going to take up a thousand words of code to make the graphics that you want, but it is there to make the visuals and graphics that are needed to go along with any information that you have.

Even though you can rest assured that this library is not going to take a thousand words to use, you will find that it is a massive library to look through, and getting the plot to behave the way that you want, and even choosing the right kind of plot, is going to be something that, at some points, you need to achieve with the help of trial and error. Using one-liners to generate some of the basic plots that come in this kind of coding does not have to be difficult, a lot of the rest of the library can seem a bit daunting at times. And this is why we are going to take a look at what matplotlib is all about, and why it is something that you should consider learning so you can add some of these graphs and visuals in with your data.

First, we need to explore why matplotlib is sometimes seen as confusing. Learning how to work with this kind of library is sometimes going to be frustrating when you first get started, this isn’t to say that the matplotlib

documentation is going to be lacking because there is an extensive amount of documentation that is present. However, there are a few challenges that can show up for programmers, and some of these are going to include:

1. *The library that you can use is going to be pretty large. In fact, it is going to contain about 70,000 lines of code, and it is always expanding, so it is likely that this is going to increase over time.*
2. *Matplotlib is going to be home for more than one type of interface, or ways to construct a figure and it is able to interact with a lot of backends as well. The backend is going to deal with the whole process that happens when the chart is rendered, not just structured. This can lead to some problems along the way.*
3. *While this library is going to be very comprehensive, it is going to be a part of NumPy and SciPy, so you need to make sure that you know how to use these languages ahead of time to make things easier.*

To help us see how this one can work, we need to be able to start with a bit of history on this topic. John D. Hunter, known as a neurobiologist, began to develop this particular library in 2003. He was originally inspired to emulate the commands that were found in the MATLAB software that came with Mathworks.

Hunter passed on in 2012, and now the matplotlib library is a community effort that is designed and even maintained by a host of others.

One of the relevant features that come with the MATLAB is that it has a global style. The Python concept of importing is important at times, but it is not going to be used all that heavily when we look at MATLAB, and most of the functions that we use with this are going to be available to the user when they are on the top level.

Knowing that the matplotlib has its roots in the MATLAB process can help to explain why pylab is going to exist in the first place. Pylab is going to be a module that happens inside of our library of matplotlib that was built to mimic some of that global style that we can find in the MATLAB. It is only in existence to help bring in several classes and even functions from

matplotlib and NumPy together to make a namespace. This is going to be useful for those who want to transition from the former MATLAB without having to import the statements.

The biggest issue that did show up here is something that could be apparent to some of those who had used Python in the past. Using the form pylab import in a session or a script could be done, but it was generally seen as something that was in bad practice. Matplotlib is not going to advice against doing this at all in some of the tutorials it has released. Internally, there are lots of potentially conflicting imports that are being used and masked in the short-come source and because of this, the matplotlib library has abandoned some of the convenience that goes with this model and recommends to all users that they should not work with pylab, bringing them more in line with some of the key parts of the Python language, so that these can work better together.

Another part of this that we need to look at is the matplotlib object hierarchy. If you have gone through any kind of beginners' tutorial on this library, you have likely called up something like plt.plot([1, 2, 3]) or something similar. This is going to be a good one to use on occasion, but we have to remember that this one-liner is going to hide the fact that the plot is going to be a hierarchy that has objects of Python that are nested and hidden inside. In this case, the hierarchy is going to mean that in the objects, there is going to be a structure that is similar to a trial that is hiding with each of the plots that we have.

A figure object is going to be the container that is on the outside for this kind of graphic. Instead, we are going to see that there are multiple Axes objects that we are then able to use. One source of confusion that we can run into is the name of Axes. This is something that can translate into what we think of as an individual graph or plot, rather than the axis that we are used to seeing on a chart or graph.

You can think of the function of Figure that we are using as a box-like container that will hold onto at least one, but often more than one, Axes or the actual plots. Below the Axes that we see, there is going to be a hierarchy of smaller objects that could include text boxes, legends, individual lines, and tick marks to name a few, and almost all of the elements that show up in this kind of chart can be manipulated by a Python object on its own. This

is going to include even the smallest of the elements, the labels, and the ticks.

The next thing that we need to look at is the stateless and the stateful approaches to matplotlib. For now, we are going to visualize at the stateful interfaces that include the state machine and the state-based, along with the stateless, which would be the object-oriented interfaces.

Almost all of the functions that you can work with from the pyplot, including plt.plot(), are going to either refer to the existing current Figure as well as to the current Axes that you worked with, or it will help you to create a brand new one if you don't have any that exists.

Those who have spent a lot of time using MATLAB before switching over may choose to word the differences here by saying something more along the lines of plt.plot() is a state machine interface that is going to track the tracks of the current figures implicitly. This may not make a lot of sense unless you have spent a ton of time working on programming, but what this means in some more common terms includes:

1. The standard interface that you can use is going to be called up with plt.plot) and other functions that are considered top-level in pyplot. Remember here that there is only going to be one Figure or Axes that you are trying to manipulate during a given time, and you don't need to go through and refer to it explicitly to get things done.
2. Modifying the underlying objects directly is going to be considered an object-oriented approach. It is common to do this by using the calling methods with an object that is Axes, which is the object that can represent the plot on its own.

To help us get a bit more information about how this plt.pot() function is going to work since we have already spent some time talking about it in this chapter, we can boil it all down and see how it works in just a few lines of code. These codes are going to be below:

```
def plot(*args, **kwargs):
  """An abridged version of plt.plot()."""
  ax = plt.gca()
```

```
  return ax.plot(*args, **kwargs)
def gca(**kwargs):
  """Get the current Axes of the current Figure."""
  return plt.gcf().gca(**kwargs)
```

Chapter 2

Neural Networks

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are one of the main categories of deep neural networks that have proven to be very effective in numerous computer science areas like object recognition, object classification, and computer vision. ConvNets have been used for many years for distinguishing faces apart, identifying objects, powering vision in self-driving cars, and robots.

A ConvNet can easily recognize countless image scenes as well as suggest relevant captions. ConvNets are also able to identify everyday objects, animals or humans, as well. Lately, Convolutional Neural Networks have also been used effectively in natural language processing problems like sentence classification.

Therefore, Convolutional Neural Networks are one of the most important tools when it comes to machine learning and deep learning tasks. LeNet was the very first Convolutional Neural Network introduced that helped significantly propel the overall field of deep learning. This very first Convolutional Neural Network was proposed by Yann LeCun back in 1988. It was primarily used for character recognition problems such as reading digits and codes.

Convolutional Neural Networks that are regularly used today for innumerable computer science tasks are very similar to this first Convolutional Neural Network proposed back in 1988.

Just like today’s Convolutional Neural Networks, LeNet was used for many character recognition tasks. Just like in LeNet, the standard Convolutional Neural Networks we use today come with four main operations, including convolution, ReLU non-linearity activation functions, sub-sampling or pooling and classification of their fully-connected layers.

These operations are the fundamental steps of building every Convolutional Neural Network. To move onto dealing with Convolutional Neural Networks in Python, we must get deeper into these four basic functions for a better understanding of the intuition lying behind Convolutional Neural Networks.

As you know, every image can be easily represented as a matrix containing multiple values. We are going to use a conventional term channel where we are referring to a specific component of images. An image derived from a standards camera commonly has three channels, including blue, red and green. You can imagine these images as three-2D matrices that are stacked over each other. Each of these matrices also comes with certain pixel values in the specific range of 0 to 255.

On the other hand, if you have a grayscale image, you only get one channel as there are no colors present, just black and white. In our case here, we are going to consider grayscale images, so the example we are studying is just a single-2D matrix that represents a grayscale image. The value of each pixel contained in the matrix must range from 0 to 255. In this case, 0 indicates a color of black, while 255 indicates a color of white.

How Convolutional Neural Networks Work?

A Convolutional Neural Network structure is normally used for various deep learning problems. As already mentioned, Convolutional Neural Networks are used for object recognition, object segmentation, detection and computer vision due to their structure. CNNs learn directly from image data, so there is no need to perform manual feature extraction, which is commonly required in regular deep neural networks.

The use of CNNs has become popular due to three main factors. The first of them is the structure of CNNs, which eliminates the need for performing manual data extraction as all data features are learned directly by the Convolutional Neural Networks. The second reason for the increasing popularity of CNNs is that they produce amazing, state-of-art object recognition results. The third reason is that CNNs can be easily retained for many new object recognition tasks to help build other deep neural networks.

A CNN can contain hundreds of layers, which each learns automatically to detect many different features of an image data. In addition, filters are commonly applied to every training image at different resolutions, so the output of every convolved image is used as the input to the following convolutional layer.

The filters can also start with very simple image features like edges and brightness, so they commonly can increase the complexity of those image features, which define the object as the convolutional layers progress.

Therefore, filters are commonly applied to every training image at different resolutions as the output of every convolved image acts as the input to the following convolutional layer.

Convolutional Neural Networks can be trained on hundreds, thousands and millions of images.

When you are working with large amounts of image data and with some very complex network structures, you should use GPUs that can significantly boost the processing time required for training a neural network model.

Once you train your Convolutional Neural Network model, you can use it in real-time applications like object recognition, pedestrian detection in ADAS or Advanced Driver Assistance Systems and many others.

The last fully-connected layer in regular deep neural networks is called the output layer and in every classification setting, this output layer represents the overall class score.

Due to these properties, regular deep neural nets are not capable of scaling to full images. For instance, in CIFAR-10, all images are sized as 32x32x3. This means that all CIFAR-10 images gave 3 color channels and that they are 32 wide and 32 inches high. This means that a single fully-connected neural network in a first regular neural net would have 32x32x3 or 3071 weights. This is an amount that is not as manageable as those fully-connected structures are not capable of scaling to larger images.

Besides, you would want to have more similar neurons to quickly add-up more parameters. However, in this case of computer vision and other similar problems, using fully-connected neurons is wasteful as your parameters would lead to over-fitting of your model very quickly. Therefore, Convolutional Neural Networks take advantage of the fact that their inputs consist of images for solving these kinds of deep learning problems.

Due to their structure, Convolutional Neural Networks constrain the architecture of images in a much more sensible way. Unlike a regular deep

neural network, the layers contained in the Convolutional Neural Network are comprised of neurons that are arranged in three dimensions including, depth, height and width. For instance, the CIFAR-10 input images are part of the input volume of all layers contained in a deep neural network and the volume comes with the dimensions of 32x32x3.

The neurons in these kinds of layers can be connected to only a small area of the layer before it, instead of all the layers being fully-connected like in regular deep neural networks. In addition, the output of the final layers for CIFAR-10 would come with dimensions of 1x1x10 as the end of Convolutional Neural Networks architecture would have reduced the full image into a vector of class score arranging it just along the depth dimension.

To summarize, unlike the regular-three-layer deep neural networks, a ConvNet composes all its neurons in just three dimensions. Besides, each layer contained in Convolutional Neural Network transforms the 3D input volume into a 3D output volume containing various neuron activations.

A Convolutional Neural Network contains layers that all have a simple API resulting in 3D output volume that comes with a differentiable function that may or may not contain neural network parameters.

A Convolutional Neural Network is composed of several subsamples and convolutional layers that are times followed by fully-connected or dense layers. As you already know, the input of a Convolutional Neural Network is an n x n x r image where n represents the height and width of an input image while the r is a total number of channels present. The Convolutional Neural Networks may also contain k filters known as kernels. When kernels are present, they are determined as q, which can be the same as the number of channels.

Each Convolutional Neural Network map is subsampled with max or mean pooling over p x p of a contiguous area in which p commonly ranges between 2 for small images and more than 5 for larger images. Either after or before the subsampling layer, a sigmoidal non-linearity and additive bias is applied to every feature map. After these convolutional neural layers, there may be several fully-connected layers and the structure of these fully-connected layers is the same as the structure of standard multilayer neural networks.

Stride and Padding

Secondly, after specifying the depth, you also must specify the stride that you slide over the filter. When you have a stride that is one, you must move one pixel at a time. When you have a stride that is two, you can move two pixels at a time, but this produces smaller volumes of output spatially. By default, stride value is one. However, you can have bigger strides in the case when you want to come across less overlap between your receptive fields, but as already mentioned, this will result in having smaller feature maps as you are skipping over image locations.

In the case when you use bigger strides, but you want to maintain the same dimensionality, you must use padding that surrounds your input with zeros. You can either pad with the values on the edge or with zeros. Once you get the dimensionality of your feature map that matches your input, you can move onto adding pooling layers that padding is commonly used in Convolutional Neural Networks when you want to preserve the size of your feature maps.

If you do not use padding, your feature maps will shrink at every layer. Adding zero-padding is times very convenient when you want to pad your input volume just with zeros all around the border.

This is called as zero-padding, which is a hyperparameter. By using zero-padding, you can control the size of your output volumes.

You can easily compute the spatial size of your output volume as a simple function of your input volume size, the convolution layers receptive field size, the stride you applied and the amount of zero-padding you used in your Convolutional Neural Network border.

For instance, if you have a 7x7 input and, if you use a 3x3 filter with stride 1 and pad 0, you will get a 5x5 output following the formula. If you have stride two, you will get a 3x3 output volume and so on using the formula as following in which W represents the size of your input volume, F represents the receptive field size of your convolutional neural layers, S represents the stride applied and P represents the amount of zero-padding you used.

(W-F +2P)/S+1

Using this formula, you can easily calculate how many neurons can fit in your Convolutional Neural Network. Consider using zero-padding

whenever you can. For instance, if you have an equal input and output dimensions, which are five, you can use zero-padding of one to get three receptive fields.

If you do not use zero-padding in cases like this, you will get your output volume with a spatial dimension of 3, as 3 are several neurons that can fit into your original input.

Spatial arrangement hypermeters commonly have mutual constraints. For instance, if you have an input size of 10 with no zero-padding used and with a filter size of three, it is impossible to apply stride. Therefore, you will get the set of your hyperparameter to be invalid and your Convolutional Neural Networks library will throw an exception or zero pad completely to the rest to make it fit.

Fortunately, sizing the convolutional layers properly, so that all dimensions included work using zero-padding, can make any job easier.

Parameter Sharing

You can use a parameter sharing schemes in your convolutional layers to entirely control the number of used parameters. If you denoted a single two-dimensional slice of depth as your depth slice, you can constrain the neurons contained in every depth slice to use the same bias and weights. Using parameters sharing techniques, you will get a unique collection of weights, one of every depth slice, and you will get a unique collection of weights. Therefore, you can significantly reduce the number of parameters contained in the first layer of your ConvNet. Doing this step, all neurons in every depth slice of your ConvNet will use the same parameters.

In other words, during backpropagation, every neuron contained in the volume will automatically compute the gradient for all its weights.

However, these computed gradients will add up over every depth slice, so you get to update just a single collection of weights per depth slice. In this way, all neurons contained in one depth slice will use the same weight vector. Therefore, when you forward the pass of the convolutional layers in every depth slice, it is computed as a convolution of all neurons' weights alongside the input volume. This is the reason why we refer to the collection of weights we get as a kernel or a filter, which is convolved with your input.

However, there are a few cases in which this parameter sharing assumption does not make any sense. This is commonly the case with many input images to a convolutional layer that comes with a certain centered structure, where you must learn different features depending on your image location.

For instance, when you have an input of several faces, which have been centered in your image, you probably expect to get different hair-specific or eye-specific features that could be easily learned at many spatial locations. When this is the case, it is very common to just relax this parameter sharing scheme and simply use a locally- connected layer.

Matrix Multiplication

The convolution operation commonly performs those dot products between the local regions of the input and between the filters. In these cases, a common implementation technique of the convolutional layers is to take full advantage of this fact and to formulate the specific forward pass of the main convolutional layer representing it as one large matrix multiply.

Implementation of matrix multiplication is when the local areas of an input image are completely stretched out into different columns during an operation known as im2col. For instance, if you have an input of size 227x227x3 and you convolve it with a filter of size 11x11x3 at a stride of 4, you must take blocks of pixels in size 11x11x3 in the input and stretch every block into a column vector of size 363.

However, when you iterate this process in your input stride of 4, you get 55 locations along with both weight and height that lead to an output matrix of x column in which every column is a maximally stretched out receptive fields and where you have 3025 fields in total.

Each number in your input volume can be duplicated in multiple distinct columns. Also, remember that the weights of the convolutional layers are very similarly stretched out into certain rows as well. For instance, if you have 95 filters in size of 11x11x3, you will get a matrix of w row of size 96x363.

When it comes to matrix multiplications, the result you get from your convolution will be equal to performing one huge matrix multiply that evaluates the dot products between every receptive field and between every filter resulting in the output of your dot production of every filter at every

location. Once you get your result, you must reshape it back to its right output dimension, which in this case is 55x55x96.

This is a great approach, but it has a downside. The main downside is that it uses a lot of memory as the values contained in your input volume will be replicated several times. However, the main benefit of matrix multiplications is that many implementations can improve your model. In addition, this im2col can be re-used many times when you are performing pooling operation.

Chapter 3 Decision Trees

Decision trees are built similarly to support vector machines, meaning they are a category of supervised machine learning algorithms that are capable of solving both regression and classification problems. They are powerful and used when working with a great deal of data.

You need to learn beyond the barebones basics so that you can process large and complex datasets. Furthermore, decision trees are used in creating random forests, which is arguably the most powerful learning algorithm. In this chapter, we are going to exclusively focus on decision trees explicitly because of their popular use and efficiency.

An Overview on Decision Trees

Decision trees are essentially a tool that supports a decision that will influence all the other decisions that will be made. This means that everything from the predicted outcomes to consequences and resource usage will be influenced in some way. Take note that decision trees are usually represented in a graph, which can be described as some kind of chart where the training tests appear as a node. For instance, the node can be the toss of a coin which can have two different results. Furthermore, branches sprout to individually represent the results and they also have leaves which are the class labels. Now you see why this algorithm is called a decision tree. The structure resembles an actual tree. As you probably guessed, random forests are exactly what they sound like. They are collections of decision trees, but enough about them.

Decision trees are one of the most powerful supervised learning methods you can use, especially as a beginner. Unlike other more complex algorithms, they are fairly easy to implement and they have a lot to offer. A decision tree can perform any common data science task and the results you obtain at the end of the training process are highly accurate. With that in mind, let’s analyze a few other advantages, as well as disadvantages, to gain a better understanding of their use and implementation.

Let’s begin with the positives:

1. Decision trees are simple in design and therefore easy to implement even if you are a beginner without a formal education in data science or machine learning. The concept behind this algorithm can be summarized with a sort of a formula that follows a common type of programming statement: If this, then that, else that. Furthermore, the results you will obtain are very easy to interpret, especially due to the graphic representation.
2. The second advantage is that a decision tree is one of the most efficient methods in exploring and determining the most important variables, as well as discovering the connection between then. Also, you can build new features easily to gain better measurements and predictions. Don’t forget that data exploration is one of the most important stages in working with data, especially when there is a large number of variables involved. You need to be able to detect the most valuable ones in order to avoid a time-consuming process, and decision trees excel at this.
3. Another benefit of implementing decision trees is the fact that they are excellent at clearing up some of the outliers in your data. Don’t forget that outliers are noise that reduces the accuracy of your predictions. In addition, decision trees aren’t that strongly affected by noise. In many cases, outliers have such a small impact on this algorithm that you can even choose to ignore them if you don’t need to maximize the accuracy scores.

Finally, there’s the fact that decision trees can work with both numerical as well as categorical variables. Remember that some of the algorithms we already discussed can only be used with one data type or the other. Decision trees, on the other hand, are proven to be versatile and handle a much more varied set of tasks.

As you can see, decision trees are powerful, versatile, and easy to implement, so why should we ever bother using anything else? As usual, nothing is perfect, so let’s discuss the negative side of working with this type of algorithm:

1. One of the biggest issues encountered during a decision tree implementation is overfitting. Take note that this algorithm tends to sometimes create very complicated decision trees that will have issues generalizing data due to their complexity. This is known as overfitting and it is encountered when implementing other learning algorithms as well, however, not to the same degree. Fortunately, this doesn't mean you should stay away from using decision trees. All you need to do is invest some time to implement certain parameter limitations to reduce the impact of overfitting.

2. Decision trees can have issues with continuous variables. When continuous numerical variables are involved, the decision trees lose a certain amount of information. This problem occurs when the variables are categorized. If you aren't familiar with these variables, a continuous variable can be a value that is set to be within a range of numbers. For example, if people between ages 18 and 26 are considered of student age, then this numerical range becomes a continuous variable because it can hold any value between the declared minimum and maximum.

While some disadvantages can add to additional work in the implementation of decision trees, the advantages still outweigh them by far.

Classification and Regression Trees

We discussed earlier that decision trees are used for both regression tasks as well as classification tasks. However, this doesn't mean you implement the exact same decision trees in both cases. Decision trees need to be divided into classification and regression trees. They handle different problems; however, they are similar in some ways since they are both types of decision trees.

Take note that classification decision trees are implemented when there's a categorical dependent variable. On the other side, a regression tree is only implemented in the case of a continuous dependent variable. Furthermore, in the case of a classification tree, the result from the training data is the

mode of the total relevant observations. This means that any observations that we cannot define will be predicted based on this value, which represents the observation which we identify most frequently.

Regression trees, on the other hand, work slightly differently. The value that results from the training stage is not the mode value, but the mean of the total observations. This way, the unidentified observations are declared with the mean value which results from the known observations.

Both types of decision trees undergo a binary split; however, going from the top to bottom. This means that the observations in one area will spawn two branches that are then divided inside the predictor space. This is also known as a greedy approach because the learning algorithm is seeking the most relevant variable in the split while ignoring the future splits that could lead to the development of an even more powerful and accurate decision tree.

As you can see, there are some differences as well as similarities between the two. However, what you should note from all of this is that the splitting is what has the most effect on the accuracy scores of the decision tree implementation. Decision tree nodes are divided into subnodes, no matter the type of tree. This tree split is performed to lead to a more uniform set of nodes.

Now that you understand the fundamentals behind decision trees, let’s dig a bit deeper into the problem of overfitting.

The Overfitting Problem

You learned earlier that overfitting is one of the main problems when working with decision trees and sometimes it can have a severe impact on the results. Decision trees can lead to a 100% accuracy score for the training set if we do not impose any limits. However, the major downside here is that overfitting creeps in when the algorithm seeks to eliminate the training errors, but by doing so it actually increases the testing errors. This imbalance, despite the score, leads to terrible prediction accuracy in the result. Why does this happen? In this case, the decision trees grow many branches and that’s the cause of overfitting. To solve this use, you need to impose limitations on how much the decision tree can develop and how many branches it can spawn. Furthermore, you can also prune the tree to

keep it under control, much like how you would do with a real tree in order to make sure it produces plenty of fruit.

To limit the size of the decision tree, you need to determine new parameters during the definition of the tree. Let’s analyze these parameters:

1. min_samples_split: The first thing you can do is change this parameter to specify how many observations a node will require to be able to perform the splitting. You can declare anything with a range of one sample to maximum samples. Just keep in mind that to limit the training model from determining the connections that are very common to a particular decision tree, you need to increase the value. In other words, you can limit the decision tree with higher values.
2. min_samples_leaf: This is the parameter you need to tweak to determine how many observations are required by a node, or in other words, a leaf. The overfitting control mechanism works the same way as for the samples split parameter.
3. max_features: Adjust this parameter in order to control the features that are selected randomly. These features are the ones that are used to perform the best split. To determine the most efficient value, you should calculate the square root of the total features. Just keep in mind that in this case, the higher value tends to lead to the overfitting problem we are trying to fix. Therefore, you should experiment with the value you set. Furthermore, not all cases are the same. Sometimes a higher value will work without resulting in overfitting.
4. max_depth: Finally, we have the depth parameter, which consists of the depth value of the decision tree. To limit the overfitting problem, however, we are only interested in the maximum depth value. Take note that a high value translates to a high number of splits, therefore a high amount of information. By tweaking this value you will have control over how the training model learns the connections in a sample.

Modifying these parameters is only one aspect of gaining control of our decision trees in order to reduce overfitting and boost performance and accuracy. The next step after applying these limits is to prune the trees.

Pruning

This technique might sound too silly to be real; however, it is a legitimate machine learning concept that is used to improve your decision tree by nearly eliminating the overfitting issue. As with real trees, what pruning does is reduce the size of the trees in order to focus the resources on providing highly accurate results. However, you should keep in mind that the segments that are pruned are not entirely randomly selected, which is a good thing. The sections that are eliminated are those that don't help with the classification process and don't lead to any performance boosts. Less complex decision trees lead to a better-optimized model.

In order to better understand the difference between an unmodified decision tree and one that was pruned and optimized, you should visualize the following scenario. Let's say that there's a highway that has a lane for vehicles that travel at 80 mph and a second lane for the slower vehicles that travel at 50 mph. Now let's assume you are on this highway in a red car and you are facing a decision. You have the option to move on the fast lane to pass a slow-moving car; however, this means that you will have a truck in front of you that can't achieve the high speed he should have in the left lane and therefore you will be stuck on that lane. In this case, the cars that are in the other lane are slowly starting to overpass you because the truck can't keep up. The other option is staying in your lane without attempting to make a pass. The most optimal choice here is the one that allows you to travel a longer distance during a certain amount of time. Therefore if you choose to stay in the slow lane until you gradually pass the truck that is blocking the fast lane, you will eventually be able to switch to that lane and pass all the other vehicles. As you can see, the second option might look slow at the time of consideration; however, in the long run, it ends up being the most efficient one. Decision trees are the same. If you apply limits to your trees, they won't get greedy by switching you to the left lane where you will be stuck behind a truck. However, if you prune the decision tree, it will allow you to examine your surroundings in more detail and allow you

predict a higher number of options you have to be able to make a better choice.

As you can see, performing the pruning process does yield some benefits that cannot be ignored. However, the implementation of this technique requires a number of steps and conditions. For instance, for a decision tree to be suitable for pruning, it needs to have a high depth value. Furthermore, the process needs to start at the bottom to avoid any negative returns. This issue needs to be avoided because if we have a negative node split at the bottom and another one occurs at the top, we will end up with a decision tree that will stop when the first division occurs. If the tree is pruned, it will not stop there and you will have higher gains.

Visualizing decision trees can sometimes be difficult when all you have is the theory, so let's start with a step by step implementation to see them in action.

Decision Tree Implementation

Creating a decision tree starts from the root node. The first step is to select one of the data attributes and set up a logical test based on it. Once you have a set of results you can branch out and create another set of tests, which you will use to create the subnode. Once we have at least a subnode, we can apply a recursive splitting process on it in order to determine that we have clean decision tree leaves. Keep in mind that the level of purity is determined based on the number of cases that sprout from a single class. At this stage, you can start pruning the tree to eliminate anything that doesn't improve the accuracy of the classification stage. Furthermore, you will also have to evaluate every single split that is performed based on each attribute. This step needs to be performed in order to determine which is the most optimal attribute, as well as split.

But enough theory for now. All you should focus on at this point is the fundamental idea behind decision trees and how to make them efficient. Once you think you grasped the basics, you need to start the implementation.

In the following example, we will once again rely on the Iris dataset for the data and the Scikit-learn library which contains it.

K-means Clustering

As mentioned, unsupervised learning methods are ideal for working with unlabeled data. However, to be more specific, one of the best techniques, if not the best, is to use a type of clustering algorithm. The main idea behind this approach is the cluster analysis which involves reducing data observations to clusters, or subdivisions of data, where each cluster contains information that is similar to that of a predefined attribute. Clustering involves several techniques that all achieve the same goal because they are all about forming a variety of theories regarding the data structure.

One of the most popular unsupervised learning algorithms and clustering techniques is known as k-means clustering. This concept revolves around building data clusters based on the similarity of the values. The first step is to determine the value of k, which is represented by the total number of clusters we define. These clusters are built as k-many points, which hold the average value that represents the entire cluster. Furthermore, the values are assigned based on the value, which is the closest average. Keep in mind that clusters have a core that is defined as an average value which pushes the other averages aside, changing them. After enough iterations, the core value will shift itself to a point where the performance metric is lower. When this stage is reached, we have the solutions because there aren't any observations available to be designated.

If you're confused by now by all this theory, that's ok. You will see that this technique is a lot easier than it sounds. Let's take a look at the practical implementation. In this example we will use the UCI handwritten digits dataset. It is freely available and you don't need to download if you are using Scikit-learn along with the book. With that being said, here's the code:

```
from time import time

import numpy as np

import matplotlib.pyplot as plt

from sklearn import datasets

np.random.seed()
```

```
digits = datasets.load_digits()
data = scale(digits.data)
n_samples, n_features = data.shape
n_digits = len(np.unique(digits.target))
labels = digits.target
sample_size = 300
print("n_digits: %d, \t n_samples %d, \t n_features %d"
% (n_digits, n_samples, n_features))
print(79 * '_')
print('% 9s' % 'init"    time  inertia  homo  compl  v-meas
ARI AMI  silhouette')
def bench_k_means(estimator, name, data):
t0 = time()
estimator.fit(data)
print('% 9s %.2fs %i %.3f %.3f %.3f %.3f %.3f %.3f'
% (name, (time() - t0), estimator.inertia_,
metrics.homogeneity_score(labels, estimator.labels_),
metrics.completeness_score(labels, estimator.labels_),
metrics.v_measure_score(labels, estimator.labels_),
metrics.adjusted_rand_score(labels, estimator.labels_),
metrics.silhouette_score(data, estimator.labels_,
metric='euclidean',
sample_size=sample_size)))
```

(Source: K Means clustering – Implementing k Means. Adapted from https://techwithtim.net/tutorials/machine-learning-Python/k-means-2/ accessed in October 2019)

If you analyze the code line by line you will notice that the implementation is fairly simple, logical, and easy to understand. In fact, it is somewhat similar in parts to other techniques we used so far. However, there is one major difference that is important to mention, namely the performance measurements we are using in order to accurately interpret the data.

First, we have a homogeneity score. This metric can have a value between zero and one. It mainly seeks the clusters that make room only for one class system. The idea is that if we have a score that is close to the value of one, then the cluster is mostly built from the samples that belong to a single class. On the other hand, if the score is close to zero, then we have achieved a low homogeneity.

Next, we have the completeness score. This metric compliments the homogeneity measure. Its purpose is to give us information on how the measurements became part of a specific class. The two scores allow us to form the conclusion that we either managed to perform perfect clustering, or we simply failed.

The third metric is known as the V-metric, or sometimes the V-measure. This score is calculated as the harmonic mean of the previous two scores. The V-metric essentially checks on the homogeneity and the completeness score by assigning a zero to one value that verifies the validity.

Next, we have the adjusted Rand index metric. This is a score that is used to verify the similarity of the labeling. Using a value between zero and one, the Rand index simply determines the relation between the distribution sets.

Finally, we have the silhouette metric, which is used to verify whether the performance of the clustering is sufficient without having labeled data. The measurement goes from a negative value to a positive one and it determines whether the clusters are well-structured or not. If the value is negative, then we are dealing with bad clusters. To make sure we have dense clusters, we need to achieve a score close to a positive one. Keep in mind that in this case, we can also have a score that is close to zero. In this case, the silhouette measurement tells us that we have clusters which are overlapping each other.

Now that you understand the measurement system, we have to apply one more step to this implementation and make sure that the results are accurate. To verify the clustering scores, we can use the bench_k_means function like so:

```
bench_k_means(KMeans(init='k-means++', n_clusters=n_digits, n_
init=10),

name="k-means++", data=data) print(79 * '_')
```

Now let’s see what conclusion we can draw from the scores. Here’s how your results should look:

```
n_digits: 10,        n_samples 1797,        n_features 64

init                 time                 inertia        homo
compl

k-means++ 0.25s      69517                0.596          0.643

init                 v-meas      ARI                AMI
silhouette

k-means++            0.619       0.465        0.592         0.123
```

As you can see, with a basic k-means implementation, we have fairly decent scores; however, there is a lot we could improve. Clustering is sufficient, but we could perfect the scores by implementing other supervised or unsupervised learning techniques. For instance, in this case, you might consider using the principal component analysis algorithm as well. Another option would be applying various dimensionality reduction methods. However, to learn how to implement the K-means clustering algorithm, these results will suffice. However, you should keep in mind that in the real world of data science, you will often implement several algorithms and techniques together. You will rarely be able to get useful results with just one algorithm, especially when working with raw datasets instead of the practice ones.

In this chapter, we have focused on learning about unsupervised learning algorithms, namely K-means clustering. The purpose of this section was to show you a technique that can be used on more complex datasets. Clustering algorithms are the staple of data science and frequently used,

especially in combination with other algorithms and learning techniques. Furthermore, as you will learn later, clustering techniques, especially K-means clustering, are highly efficient in dealing with Big Data.

Chapter 4 Applications of Big Data Analysis

The applications of Big Data and Big Data Analytics are benefitting both small and big companies across various industrial domains. In this chapter, we are going to explore more in detail such applications.

eCommerce

Over 2.6 billion and counting active social media users include customers and potential customers for every company out there. The race is on to create more effective marketing and social media strategies, powered by machine learning, aimed at providing enhanced customer experience to turn prospective customers into raving fans. The process of sifting through and analyzing a massive amount of data has not only become feasible, but it's easy now. The ability to bridge the gap between execution and big Data Analysis has been supplemented by artificial intelligence marketing solutions.
Artificial Intelligence (AI) marketing can be defined as a method of you using artificial intelligence consonants like machine learning on available customer data to anticipate customer's needs and expectations while significantly improving the customer's journey. Marketers can boost their campaign performance and return on investment read a little to no extra effort in the light of big data insights provided by artificial intelligence marketing solutions. The key elements that make AI marketing as powerful are:

- Big data - A marketing company's ability to aggregate and segment a huge dump of data with minimal manual work is referred to as Big Data. The marketer can then leverage the desired medium to ensure the appropriate message is being delivered to the target audience at the right time.
- Machine learning - Machine learning platforms enable marketers to identify trends or common occurrences and gather effective insights and responses, thereby deciphering the root cause and probability of recurring events.

- Intuitive platform – Super fast and easy to operate applications are integral to AI marketing. Artificial intelligence technology is capable of interpreting emotions and communicating like a human, allowing AI-based platforms to understand open form content like email responses and social media.

Predictive Analysis

All artificial intelligence technology-based solutions are capable of extracting information from data assets to predict future trends. AI technology has made it possible to model trends that could previously be determined only retroactively. These predictive analysis models can be reliably used in decision-making and to analyze customers' purchase behavior. The model can successfully determine when the consumer is more likely to purchase something new or reorder an old purchase. The marketing companies are now able to reverse engineer customer's experiences and actions to create more lucrative marketing strategies. For example, FedEx and Sprint are using predictive analytics to identify customers who are at potential risk of deflecting to the competitor.

Smart searches

Only a decade ago, if you type in "women's flip flops" on Nike.com, the probability of you finding what you were looking for would be next to zero. However, today's search engines are not only accurate but also much faster. This upgrade has largely been brought on by innovations like "semantic search" and "natural language processing" that enable search engines to identify links between products and provide relevant search results, recommend similar items, and auto-correct typing errors. The artificial intelligence technology and big data solutions can rapidly analyze user search patterns and identify key areas that the marketing companies should focus on.

In 2015, Google introduced the first Artificial Intelligence-based search algorithm called “RankBrain.” Following Google’s lead, other major e-commerce websites, including Amazon has incorporated big Data Analysis and artificial intelligence into their search engines to offer smart search experience for their customers, who can find desired products even when they don’t know exactly what they’re looking for. Even small e-commerce

stores have access to Smart search technologies like “Elasticsearch.” The data-as-a-service companies like “Indix" allow companies to learn from other larger data sources to train their product search models.

Recommendation Engines

Recommendation engines have quickly evolved into fan favorites and are loved by the customers just as much as the marketing companies. "Apple Music" already knows your taste in music better than your partner, and Amazon always presents you with a list of products that you might be interested in buying. This kind of discovery aide that can sift through millions of available options and hone in on an individual's needs are proving indispensable for large companies with huge physical and digital inventories.

In 1998, Swedish computational linguist, Jussi Karlgren, explored the practice of clustering customer behaviors to predict future behaviors in his report titled “Digital bookshelves.” The same here, Amazon implemented collaborative filtering to generate recommendations for their customers. The gathering and analysis of consumer data paired with individual profile information and demographics, by the predictive analysis based systems allow the system to continually learn and adapt based on consumer activities such as likes and dislikes on the products in real-time. For example, the company "Sky" has implemented a predictive analysis based model that is capable of recommending content according to the viewer's mode. The smart customer is looking for such an enhanced experience not only from their Music and on-demand entertainment suppliers but also from all other e-commerce websites.

Product Categorization and Pricing

E-commerce businesses and marketing companies have increasingly adopted artificial intelligence in their process of categorization and tagging of the inventory. The Marketing companies are required to deal with awful data just as much, if not more than amazingly organized, clean data. This bag of positive and negative examples serves as training resources for predictive analysis based classification tools. For example, different detailers can have different descriptions for the same product, such as

sneakers, basketball shoes, trainers, or Jordan's, but the AI algorithm can identify that these are all the same products and tag them accordingly. Alternatively, if the data set is missing the primary keyword like skirts or shirts, the artificial intelligence algorithm can identify and classify the item or product as skirts or shirts based solely on the surrounding context.

We are familiar with the seasonal rate changes of the hotel rooms, but with the advent of artificial intelligence, product prices can be optimized to meet the demand with a whole new level of precision. The machine learning algorithms are being used for dynamic pricing by analyzing customer's data patterns and making near accurate predictions of what they are willing to pay for that particular product as well as their receptiveness to special offers. This empowers businesses to target their consumers with high precision and calculated whether or not a discount is needed to confirm the sale. Dynamic pricing also allows businesses to compare their product pricing with the market leaders and competitors and adjust their prices accordingly to pull in the sale. For example, “Airbnb” has developed its dynamic pricing system, which provides ‘Price Tips' to the property owners to help them determine the best possible listing price for their property. The system takes into account a variety of influencing factors such as geographical location, local events, property pictures, property reviews, listing features and most importantly, the booking timings and the market demand. The final decision of the property owner to follow or ignore the provided ‘price tips' and the success of the listing are also monitored by the system, which will then process the results and adjust its algorithm accordingly.

Customer Targeting and Segmentation

For the marketing companies to be able to reach their customers with a high level of personalization, they are required to target increasingly granular segments. The artificial intelligence technology can draw on the existing customer data and train Machine learning algorithms against "gold standard" training sets to identify common properties and significant variables. The data segments could be as simple as location, gender, and age, or as complex as the buyer's persona and past behavior. With AI Dynamics Segmentation is feasible which accounts for the fact that

customers' behaviors are ever-changing, and people can take on different people in different situations.

Sales and Marketing Forecast

One of the most straightforward artificial intelligence applications in marketing is in the development of sales and marketing forecasting models. The high volume of quantifiable data such as clicks, purchases, email responses, and time spent on webpages serve as training resources for the machine learning algorithms. Some of the leading business intelligence and production companies in the market are Sisense, Rapidminer, and Birst. Marketing companies are continuously upgrading their marketing efforts, and with the help of AI and machine learning, they can predict the success of their marketing initiatives or email campaigns. Artificial intelligence technology can analyze past sales data, economic trends, as well as industrywide comparisons to predict short and long-term sales performance and forecast sales outcomes. The sales forecasts model aid in the estimation of product demand and to help companies manage their production to optimize sales.

Programmatic Advertisement Targeting

With the introduction of artificial intelligence technology, bidding on and targeting program based advertisement has become significantly more efficient. Programmatic advertising can be defined as “the automated process of buying and selling ad inventory to an exchange which connects advertisers to publishers.” To allow real-time bidding for inventory across social media channels and mobile devices as well as television, artificial intelligence technology is used. This also goes back to predictive analysis and the ability to model data that could previously only be determined retroactively. Artificial intelligence is able to assist the best time of the day to serve a particular ad, the probability of an ad turning into sales, the receptiveness of the user, and the likelihood of engagement with the ad.

Programmatic companies can gather and analyze visiting customers’ data and behaviors to optimize real-time campaigns and to target the audience more precisely. Programmatic media buying includes the use of "demand-side platforms" (to facilitate the process of buying ad inventory on the open

market) and "data management platforms" (to provide the marketing company an ability to reach their target audience). In order to empower the marketing rep to make informed decisions regarding their prospective customers, the data management platforms are designed to collect and analyze the big volume of website "cookie data." For example, search engine marketing (SEM) advertising practiced by channels like Facebook, Twitter, and Google. To efficiently manage huge inventory of the website and application viewers, programmatic ads provide a significant edge over competitors. Google and Facebook serve as the gold standard for efficient and effective advertising and are geared to words providing a user-friendly platform that will allow non-technical marketing companies to start, run and measure their initiatives and campaigns online.

Visual Search and Image Recognition

Leaps and bounds of the advancements in artificial intelligence-based image recognition and analysis technology have resulted in uncanny visual search functionalities. With the introduction of technology like Google Lens and platforms like Pinterest, people can now find results that are visually similar to one another using the visual search functionality. The visual search works in the same way as traditional text-based searches that display results on a similar topic. Major retailers and marketing companies are increasingly using the visual search to offer an enhanced and more engaging customer experience. Visual search can be used to improve merchandising and provide product recommendations based on the style of the product instead of the consumer's past behavior or purchases.

Major investments have been made by Target and Asos in the visual search technology development for their e-commerce website. In 2017, Target announced a partnership with interest that allows integration of Pinterest's visual search application called "Pinterest lens" into Target's mobile application. As a result, shoppers can take a picture of products that they would like to purchase while they are out and about and find similar items on Target's e-commerce site. Similarly, the visual search application launched by Asos called “Asos’ Style Match” allows shoppers to snap a photo or upload an image on the Asos website or application and search their product catalog for similar items. These tools attract shoppers to retailers for items that they might come across in a magazine or while out

and about by helping them to shop for the ideal product even if they do not know what the product is.

Image recognition has tremendously helped marketing companies to gain an edge on social media by allowing them to find a variety of uses of their brand logos and products in keeping up with the visual trends. This phenomenon is also called "visual social listening" and allows companies to identify and understand where and how customers are interacting with their brand, logo, and product even when the company is not referred directly by its name.

Healthcare Industry

With the increasing availability of healthcare data, big Data Analysis has brought on a paradigm shift to healthcare. The primary focus of big data analytics in the healthcare industry is the analysis of relationships between patient outcomes and the treatment or prevention technique used. Big Data Analysis driven Artificial Intelligence programs have successfully been developed for patient diagnostics, treatment protocol generation, drug development, as well as patient monitoring and care. The powerful AI techniques can sift through a massive amount of clinical data and help unlock clinically relevant information to assist in decision making.

Some medical specialties with increasing big Data Analysis based AI research and applications are:

- Radiology – The ability of AI to interpret imaging results supplements the clinician's ability to detect changes in an image that can easily be missed by the human eye. An AI algorithm recently created at Stanford University can detect specific sites in the lungs of the pneumonia patients.
- Electronic Health Records – The need for digital health records to optimize the information spread and access requires fast and accurate logging of all health-related data in the systems. A human is prone to errors and may be affected by cognitive overload and burnout. This process has been successfully automated by AI. The use of Predictive models on the electronic health records data allowed the prediction of individualized treatment response with 70-72% accuracy at baseline.

- Imaging – Ongoing AI research is helping doctors in evaluating the outcome of corrective jaw surgery as well as in assessing the cleft palate therapy to predict facial attractiveness.

Entertainment Industry

Big Data Analysis, in coordination with Artificial intelligence, is increasingly running in the background of entertainment sources from video games to movies and serving us a richer, engaging, and more realistic experience. Entertainment providers such as Netflix and Hulu are using big Data Analysis to provide users personalized recommendations derived from individual user's historical activity and behavior. Computer graphics and digital media content producers have been leveraging big Data Analysis based tools to enhance the pace and efficiency of their production processes. Movie companies are increasingly using machine learning algorithms in the development of film trailers and advertisements as well as pre-and post-production processes. For example, big Data Analysis and an artificial intelligence-powered tool called "RivetAI" allows producers to automate and excellently read the processes of movie script breakdown, storyboard as well as budgeting, scheduling, and generation of shot-list. Certain time-consuming tasks carried out during the post-production of the movies such as synchronization and assembly of the movie clips can be easily automated using artificial intelligence.

Marketing and Advertising

A machine learning algorithm developed as a result of big Data Analysis can be easily trained with texts, stills, and video segments as data sources. It can then extract objects and concepts from these sources and recommend efficient marketing and advertising solutions. For example, a tool called "Luban" was developed by Alibaba that can create banners at lightning speed in comparison to a human designer. In 2016, for the Chinese online shopping extravaganza called "Singles Day," Luban generated a hundred and 17 million banner designs at a speed of 8000 banner designs per second.

The "20th Century Fox" collaborated with IBM to use their AI system "Watson" for the creation of the trailer of their horror movie "Morgan." To

learn the appropriate "moments" or clips that should appear in a standard horror movie trailer, Watson was trained to classify and analyze input "moments" from audio-visual and other composition elements from over a hundred horror movies. This training resulted in the creation of a six-minute movie trailer by Watson in a mere 24 hours, which would have taken human professional weeks to produce.

With the use of Machine learning, computer vision technology, natural language processing, and predictive analytics, the marketing process can be accelerated exponentially through an AI marketing platform. For example, the artificial intelligence-based marketing platform developed by Albert Intelligence Marketing can generate autonomous campaign management strategies, create custom solutions and perform audience targeting. The company reported a 183% improvement in customer transaction rate and over 600% higher conversation efficiency credited to the use of their AI-based platform.

In March 2016, the artificial intelligence-based creative director called “AI-CD ß” was launched by McCann Erickson Japan as the first robotic creative director ever developed. “AI-CD ß" was given training on select elements of various TV shows and the winners from the past 10 years of All Japan Radio and Television CM festival. With the use of data mining capabilities, "AI-CD ß” can extract ideas and themes fulfilling every client’s individual campaign needs.

Personalization of User Experience

The expectations of the on-demand entertainment users for rich and engaging personal user experience are ever-growing. One of the leading on-demand entertainment platforms, Netflix, rolled out artificial intelligence-based workflow management and scheduling application called "Meson," comprised of various "machine learning pipelines" that are capable of creating, training, and validating personalization algorithms to provide personalized recommendations to users. Netflix collaborated with the University of Southern California to develop new Machine learning algorithms that can compress video for high-quality streaming without degrading image quality called "Dynamic Optimizer." This artificial intelligence technology will address streaming problems in developing nations and mobile device users by optimizing video fluency and definition.

IBM Watson recently collaborated with IRIS.TV to offer a business-to-business service to media companies such as CBS, The Hollywood Reporter, and Hearst Digital Media by tracking and improving the introduction of their customers with their web content. IBM Watson is boosting IRIS.TV company's Machine learning algorithms that can 'learn' from users' search history and recommend similar content. Reportedly, a 50% increase in view or retention or a small PDF three months was achieved by the Hollywood reporter with the use of IRIS.TV application.

Search Optimization and Classification

The ability to transform text, audio, and video content into digital copies has led to an explosion of media availability on the Internet, making it difficult for people to find exactly what they're looking for. To optimize the accuracy of search results, advancements are being made in machine learning technology. For example, Google is using artificial intelligence to augment its platform for accurate image searching. People can now simply upload a sample picture to Google Image instead of typing in keywords for their search. The image recognition technology used by Google image will automatically identify and manage features of the uploaded user image and provide search results with similar pictures. Google is also using artificial intelligence technology in advertisement positioning across the platform. For example, a pet food ad will only appear on the pet-related website, but a chicken wings advertisement will not appear on a site targeted to vegetarians.

The company Vintage Cloud has partnered with an artificial intelligence-based startup called "ClarifAI" to develop a film digitalization platform. With the use of computer vision API provided by ClarifAI, Vintage Cloud succeeded in burgeoning the speed of movie content classification and categorization.

A visual assets management platform integrated with machine learning algorithms has been developed by a company called “Zorroa." This platform enables users to search for specific content within large databases called an "Analysis Pipeline." The database contains processors that can tag each visual asset uniquely and Machine learning algorithms that have been ‘trained' to identify specific components of the visual data. This visual

content is then organized and cataloged to deliver high-quality search results.

Conclusion

Almost everyone will agree with the statement that big data has arrived in a big way and has taken the business world by storm. However, what is the future of Data Analysis and will it grow? What are the technologies that will grow around it? What is the future of big data? Will it grow more? Or is the big data going to become a museum article soon? What is cognitive technology? What is the future of fast data? Let's look at the answers to these questions. We'll take a look at some predictions from the experts in the field of Data Analysis and big data to get a clearer picture.

The data volume will keep on growing. There is practically no question in the minds of people that we'll keep on developing a larger and larger quantity of data, especially after taking into consideration the number of internet-connected devices and handheld devices is going to grow exponentially. The ways we undertake Data Analysis will show marked improvement in the upcoming years. Although SQL will remain the standard tool, we'll see other tools such as Spark emerging as a complementary method for the Data Analysis and their number will keep on growing as per reports.

More and more tools will become available for Data Analysis and some of them will not need the analyst. Microsoft and Salesforce have announced some combined features, which will allow the non-coders to create apps for viewing the business data. The prescriptive analytics will get built into the business analytics software and IDC predicts that 50 percent of all software related to business analysis will become available with all the business intelligence it needs by the year 2020.

In addition to these features, real-time streaming insight into the big data will turn into a hallmark for the data winners moving forward. The users will be looking to use data for making informed decisions within real-time by using programs such as Spark and Kafka. The topmost strategic trend that will emerge is machine learning. Machine learning will become a mandatory element for big data preparation and predictive analysis in businesses going forward.

You can expect big data to face huge challenges as well, especially in the field of privacy of user details. The new private regulations enforced by the

European Union intend to protect the personal information of the users. Various companies will have to address privacy controls and processes. It is predicted that most of the business ethics violations will be related to data in the upcoming years.

Soon you can pretty much expect all companies to have a chief data officer in place. Forrester says that this officer will rise in significance within a short period, but certain kinds of businesses and generation gaps might decrease their significance in the upcoming future. Autonomous agents will continue to play a significant role and they will keep on being a huge trend as per Gartner. These agents include autonomous vehicles, smart advisers, virtual personal assistants, and robots.

The staffing required for the Data Analysis will keep on expanding and people from scientists to analysts to architects to the experts in the field of data management will be needed. However, a crunch in the availability of big data talent might see the large companies develop new tactics. Some large institutes predict that various organizations will use internal training to get their issues resolved. A business model having big data in the form of service can be seen on the horizon.