

# 50 Python Projects For All

**#Pythonista**

Practical Projects with source  
codes

BY:  
**Edcorner Learning**



# 50 Python Projects For All

[OceanofPDF.com](https://oceanofpdf.com)

# 50 Python Projects For All

Edcorner Learning

[OceanofPDF.com](https://oceanofpdf.com)

# Table of Contents

## **Introduction**

### **Describe Python.**

[Why Python, you ask?](#)

[Motive for Learning Python](#)

[Standard Python Syntax](#)

[Describe syntax.](#)

[Python indentation](#)

[Syntax Identifiers in Python](#)

[Python Keywords](#)

[Variables in Python](#)

[Python comments:](#)

[Blank lines in python:](#)

[Waiting for user:](#)

[Command Line Arguments:](#)

[Sys module:](#)

[What is Indentation in Python](#)

[Python Variables:](#)

[Object References in python](#)

[Object Identity in python](#)

[What are Python Keywords](#)

[Python Identifiers:](#)

[Literals in python:](#)

[Literal Collections in python](#)

[Operators in python:](#)

[Python Input and Output](#)

[Implicit Type Conversion in Python](#)

[Explicit Type Conversion in Python](#)

[Control Flow Statements in Python](#)

[if statement](#)

[while loop in Python:](#)

[for loop in Python:](#)

[For Else and While Else in Python](#)

[Break, Pass and Continue Statement in Python](#)

[pass statement](#)

[Continue Statement in Python](#)

[Strings in Python](#)

[String Slicing in Python](#)

[String Methods Python](#)

[String Formatting in Python](#)

[Functions in Python](#)

[Scope of Variable in Python](#)

[Types of Function Arguments in Python](#)

[\\*Args and \\*\\*Kwargs in Python](#)

[Packing and Unpacking Arguments in Python](#)

[Lambda and Anonymous Function in Python](#)

[Recursion in Python](#)

[list\(\) in Python](#)

[Dictionary in Python](#)

[Sets in Python](#)

[Numbers in Python](#)

[Tuples in Python](#)

[Exception Handling in Python](#)

[Assert Keyword in Python](#)

[Opening & Closing Files in Python](#)

[How to Read a File in Python](#)

[How to Delete File in Python](#)

[With Statement in Python](#)

[Python Modules](#)

[Packages in Python and Import Statement](#)

[Python Collection Module](#)

[Regular Expression in Python](#)

**[Module 1 Project 1 -10](#)**

[1. Age Calculator GUI](#)

[2. Attachment Downloader](#)

[3. Auto Birthday Wisher](#)

[4. Automatic Backup](#)

[5. Python Script to auto fill Google Forms](#)

[6. Automate Facebook bot](#)

[7. Automatic-Certificate-Generator](#)

[8. Automate FB login](#)

[9. Brick Breaker Game](#)

[10. Bubble shooter game](#)

### **Module 2 Project 11-20**

[11. Calculate Your Age!](#)

[12. Calculator GUI](#)

[13. Calendar GUI](#)

[14. Capture Screenshot](#)

[15. Capture Video Frames](#)

[16. Check Website Connectivity](#)

[17. Chrome Automation](#)

[18. Cli Proxy Tester](#)

[19. Simple CLI Todo App](#)

[20. PDF to TXT converter](#)

### **Module 3 Projects 21-30**

[21. Convert an Image to PDF](#)

[22. Dictionary to Python Object](#)

[23. Convert Images](#)

[24. CONVERT JPEG to PNG](#)

[25. Convert a json file into a csv](#)

[26. Convert Numbers To Words](#)

[27. CONVERT PNG to ICON](#)

[28. Convert XML to JSON](#)

[29. REAL TIME COVID-19 OUTBREAK NOTIFICATION](#)

[30. Covid-19-Update-Bot](#)

### **Module 4 Projects 31-40**

[31. Create Audio Book in Python](#)

[32. create script to encrypt files and folder](#)

[33. Create a stopwatch](#)

[34. Create Calculator App.](#)

[35. Currency Converter](#)

[36. Currency Exchange Rate](#)

[37. Decimal to Binary Converter](#)

[38. Desktop Notification Application](#)

[39. Dictionary GUI](#)

[40. Digital Clock GUI:](#)

**[Module 5 Projects 41-50](#)**

[41. Digital Clock using Python and Tkinter](#)

[42. Discord Bot](#)

[43. DNS Record](#)

[44. Document-Summary-Creator](#)

[45. Document Word Detection](#)

[46. Finding Dominant Color](#)

[47. DOMINANT COLOR DETECTION FROM IMAGE](#)

[48. Download website Articles as pdf](#)

[49. Scrap images from URL](#)

[50. Duplicate Files Remover](#)

**[How to download this project:](#)**

[\*OceanofPDF.com\*](#)

# Introduction

Python is a general-purpose interpreted, interactive, object-oriented, and a powerful programming language with dynamic semantics. It is an easy language to learn and become expert. Python is one among those rare languages that would claim to be both easy and powerful. Python's elegant syntax and dynamic typing alongside its interpreted nature makes it an ideal language for scripting and robust application development in many areas on giant platforms.

Python helps with the modules and packages, which inspires program modularity and code reuse. The Python interpreter and thus the extensive standard library are all available in source or binary form for free of charge for all critical platforms and can be freely distributed. Learning Python doesn't require any pre-requisites. However, one should have the elemental understanding of programming languages.

**This Book consist of 50 Python Projects for All Developers/Students and all concepts of python practically to practice different projects and scenarios. Use these learnings in professional tasks or daily learning projects.**

**At the end of this book, you can download all this projects by using our link.**

All 50 projects are divided into different modules, every project is special in its own way of performing daily task by a developer. Every project has its source codes which learners can copy and practice/use on their own systems. If there is special requirement for any projects, its already mentioned in the book.

Happy learning!!

[OceanofPDF.com](http://OceanofPDF.com)



## Describe Python.

Guido van Rossum created Python, a high-level, object-oriented programming language. This indicates that Python is based on data and is simple for people to understand.

Prior to its discovery as a general-purpose language, Python was solely thought to be useful for automating tedious tasks. We came to understand that Python was different from purpose-specific languages like HTML/CSS, Ruby, or PHP.

Python's debut as a general-purpose language transformed the field of programming. Python was no longer utilised in the same way as other languages, i.e., merely to address certain issues. can be applied to a variety of fields, including machine learning, data science, web development, and app development.

In addition to Python's flexibility, its straightforward syntax makes it the ideal choice for any beginning coder.

## Why Python, you ask?

Python is currently employed in the industry for a variety of reasons, depending on how you want to use it.

Among the causes are:

### 1. For programmers, Python:

Python's high level of abstraction is advantageous to all programmers. It is everyone's favourite because it is extremely interactive and has a straightforward syntax.

### 2. Python for Machine Learning and AI

Python is widely used in machine learning and artificial intelligence due to its consistency, adaptability, and simplicity. Because of the readily available

AI and ML toolkits, it is one of the preferred languages for Data Scientists and Machine Learning practitioners.

### 3. Data science and analytics

Python is widely used in the fields of data analytics and data science, much as AI and ML. Nowadays, with the amount of data we produce increasing daily, efficient platforms like Python are in high demand for data analysis, manipulation, and management.

### 4. Website and video game creation

The Python development fields of web development and game development are not far behind. Python is quite effective at creating quick game prototypes, giving most developers an advantage. When it comes to duties involving online development, it can be difficult to say no to Python programming because of web frameworks like Django and Flask.

### 5. very versatile across all main platforms

Every popular operating system is supported by Python. The same code can run on many platforms without needing to be recompiled.

### 6. Sysadmins should use Python

Those who have experience with shell scripting are really grateful for Python. Python makes sure that automating tedious processes is neither tedious nor overly complicated.

### 7. Community for Python

Python has a sizable user base. This suggests that with the help of the Python Community, you may discover a solution to any issue you may encounter.

**Python: Is it open source?**

All current versions of Python are certified as open source by the Open Source Initiative and released under a GPL-compliant licence. In fact, Python is totally free—even for commercial use.

## **How Does Python Work?**

By using Python as a programming language, we can accomplish a lot. Here is a rundown of what Python can do:

- On a server, create web apps.
- Develop workflows using software.
- Modifying and reading database systems.
- Handle vast data and solve challenging challenges.
- Develop software that is production-ready quickly.

[OceanofPDF.com](https://oceanofpdf.com)

## Motive for Learning Python

Let's examine a few crucial factors:

### 1. Learning Python is simple.

You read that right, you did. Python is one of the easiest programming languages to learn thanks to its straightforward syntax, which also makes it more readable.

When we are just starting out, most people find programming intimidating. When you first begin learning, Python becomes a friend and gives you English-like syntax!

### 2. Python includes a variety of libraries, frameworks, and packages.

Consider a programmable solution, research pertinent Python libraries, import them, and then implement it by writing Python code! It really is as easy as it seems, believe it or not.

Numerous well-known and effective libraries are available in Python, including NumPy and Pandas for data science, Matplotlib and Seaborn for data visualisation, Keras, PyTorch, and many others for machine learning.

### 3. Applications of Python.

You can locate a tonne of Python Success Stories by going to [python.org](https://python.org). These success tales come from organisations that operate in industries including hiring, healthcare, finance, marketing, and education, to mention a few. So we can say that Python has a plethora of applications, and now would be an excellent time to join the Python train!

### 4. Python is an extendable, dynamically typed language.

Contrary to conventional programming languages like Java and C, variables are not strictly declared before having values assigned to them. Since the variables might have a different data type at various points throughout the execution, Python is able to be versatile. Python is now dynamically typed

as a result. Python is an expandable language since we can use it to change or add syntax and code in other languages.

[OceanofPDF.com](http://OceanofPDF.com)

## Standard Python Syntax

Python's fundamental syntax is incredibly user-friendly and straightforward. Although there are some similarities to the grammar of Java, JavaScript, and Perl, it is almost like English.

Python's syntax is unique from all other programming languages in three important ways. These are

### 1. Indentation in Python

Indentation is a feature implemented to most computer languages to improve code readability. However, indentation is highly important in Python.

The white spaces that are inserted before each line of code are what are known as indentation. Instead of using brackets, Python employs indentation to signify the start of a block of code. Python programmers must therefore always take care to correctly indent their code.

### 2. Declaration in Python

Python's dynamic typing system makes it considerably distinct from other programming languages in terms of how variables are declared. In Python, the data type does not have to be specified at declaration time; it can be changed as the programme is running.

### 3. There are three types of comments in Python:

- Single-Line Comments - These are added by first adding the "#" sign and then the appropriate comment.
- Multi-Line Comments: These can be inserted by either using the delimiter or the "#" sign on each line (""").
- Docstring: The doc property can be used to access this built-in Python functionality.

## A Python programme example

Let's develop a basic Python programme now that we know what Python is and what it can do in order to put our knowledge to use!

Check out this Python application sample. –

**Statement of the Problem** Create a Python software that will ask the user for two numbers and then print the sum.

Code:

```
# To find the sum of 2 numbers
num1 = int(input("Enter the 1st number:"))
num2 = int(input("Enter the 2nd number:"))
my_sum = num1 + num2
print("The sum of the 2 numbers is", my_sum)
```

- In the above code snippet, we took two numbers from the user using the “input” function of Python and computed the sum.
- The sum was then displayed using the “print” function.
- We also added the relevant comment for the same

**Although Python has numerous capabilities, those that make it so well-liked and application-oriented in many disciplines are covered here.**

- Simple to Learn

One of the easiest programming languages to learn is Python. In a few days, anyone can pick up the fundamentals of Python, become comfortable with its syntax, and develop simple applications. The more complex ideas and mastering Python, however, could take some

time. Python is the simplest language to learn and master when compared to other languages like C, C++, Java, etc.

- Simple to Code

Python's syntax is fairly simple. Typically, it contains English-language words. Writing code often feels like delivering directions to a young child. In addition, unlike other programming languages (such as C, C++, Java, etc.), the scope can be defined without having to worry about opening or closing any brackets. For the scope in Python, we utilise indentation (spaces or Tabs), which gives the code a neat and professional appearance.

- Language that has been translated

Python code isn't instantly compiled, transformed into an.exe file, and then run. Because Python is an interpreted language, its code is run line by line rather than all at once as is the case with other programming languages. It is also simple to debug the code thanks to this line-by-line execution.

- Open and Free Source

Python is a free and open-source programming language, thus using it for free on any operating system and without worrying about copyright issues is possible. Anyone can get Python, along with its libraries and documentation, from its official website. Although you can download, you can also create and share your own modules and libraries.

- Object-Oriented Language

The ideas of classes and objects are the foundation of the programming paradigm known as object-oriented programming. Classes act as a blueprint for objects, containing the information and methods that manipulate it. The goal of object-oriented programming



is to create code that is reusable and has a high level of abstraction. Object-Oriented programming is one of the key features of Python. The object-oriented programming structures of classes, data encapsulation, inheritance, polymorphism, and others are supported by Python. In Python, classes, objects, and OOP techniques are simple to build, use, and implement. This method enables the development of effective and potent Python applications.

- Cross-Platform Language,

Python is a platform-independent language. You may have frequently seen a list of software versions compatible with various operating systems while downloading software from a website. In Python, this is not the case; once a python programme is created on one computer or operating system, it may be executed anywhere. For instance, if we create a Python application on a Mac, we can run the exact same code on Linux, Windows, or any other operating system without making any modifications. This is due to the fact that before being executed, Python code must first be translated into an intermediary format known as Bytecode.

- A significant feature

Python may be expanded upon and is a more flexible programming language. Python shows itself to be a flexible language since, thanks to its adaptation to varied capabilities, it covers a wide range in software development applications. We can use the code that has been compiled in languages like C and C++ in our portable Python code, which can be executed everywhere. Code developed in other programming languages can be run thanks to it. By incorporating the code from other programming languages, this gives Python extra features and functionality.

- High-Level Language

Python is a high-level programming language, so users can create, comprehend, or interpret code with ease. With the aid of high-level programming languages, programmers can create codes that are less dependent on a particular machine type. Python is a programming language that has a very high abstraction from the system or machine's low-level building blocks. The architecture, memory management, or underlying machine type need not be considered by the developer while developing code.

- Database support,

A database is now a must for practically every application we create, and the Python Database API (DB-API) provides an interface to almost all of the main commercial databases. Standard Python supports a number of databases, including MySQL, PostgreSQL, Microsoft SQL, Oracle, Informix, etc. To use a specific database, we must import the interface for that database. Both relational and non-relational databases can be handled with Python.

- Support for GUI Programming

How do we engage with our computers or cellphones when we use them? What do we see? There are obviously many different application icons on the screen, and when we open some of them, we get to see a lovely visual depiction that makes it simple for us to engage with and use that particular application. This is the graphic user interface, or GUI. One of the most important features of the Python programming language is it. Tkinter, PyQt5, PSide2, and other GUI libraries are all available in Python. Python GUI packages have the capabilities that make the development of graphics-intensive software simple and quick. GUI enables a user to interact with the application and the system more simply.

- Large Standard Library

Python has a large number of cross-platform libraries and offers a wide range of modules and functions. These libraries work with many different operating systems, including UNIX, Mac, Windows, and others. We can import and use the necessary functionality instead of writing code for every little thing because there are so many libraries available. For instance, you don't have to create the request, response, and other functions from scratch if you need to access some websites and wish to scrape data from them. You can utilise any of the libraries that are available for this purpose. If we require more capability, we can also instal other packages that are not a part of the standard library.

- Dynamic Typed Language

A dynamically typed language is Python. Dynamically-typed refers to the fact that, in contrast to other programming languages, Python does not require us to explicitly declare the data type (such as int, float, double, char, etc.) of a variable. The variable's data type is chosen at runtime. In addition to this, one variable can be used to hold many sorts of data at various points in the programme. The fact that Python allows us to implicitly specify the datatype saves us a tonne of time and keeps us from falling into traps that would have happened otherwise.

## Describe syntax.

Let's learn how to create and run a simple Python programme before moving on to its syntax.

Two methods exist for doing this:

1. Interactive mode - You create and run the software in this mode.
2. Script mode - In this mode, a file (a.py file) is first created, saved, and then it is executed.

After installing Python, type the following command in your terminal to launch interactive mode.

```
$ Python
```

The interactive mode is now active. Any Python statement can now be executed interactively in the terminal. However, if you're using an IDE or script editor, you don't need to follow the same steps.

We can now proceed to comprehending the syntax. The print statement is the first to be studied when learning the syntax of Python.

[OceanofPDF.com](http://OceanofPDF.com)

```
// Edcorner: Learning Python Concepts

Print Statement:

Let us run the python program in the terminal now
>>> print("Syntax of Print")
The output for the print statement is:
Syntax of Print

Now we discussed the interactive mode above, what about the script mode? To run code in
the script mode, you must first create a .py file, write your code and save it. You can
write the same code as above (the print statement). Let's say you saved it as
syntax.py. Once you've done that, you can either use the 'run' button in your IDE or if
you would like to run it on your terminal :

$ python syntax.py

Syntax of Print

Once you execute the command, you see that it prints - Syntax of print on the terminal.
Congratulations! You've learnt your first python syntax.
```

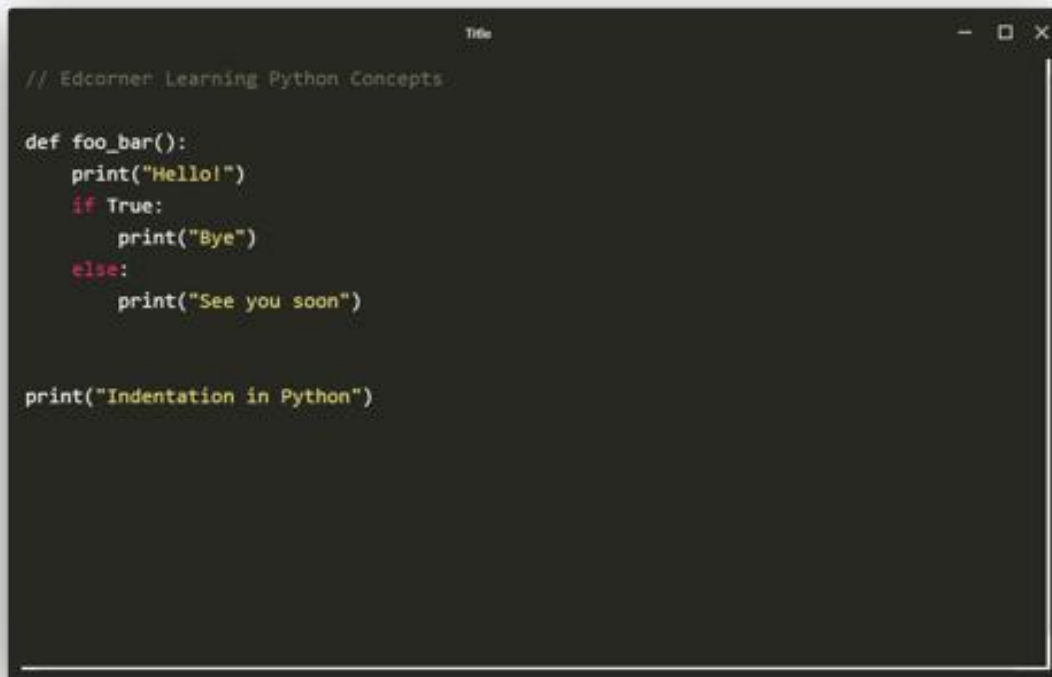
## Python indentation

Every time we write a Python programme, such as a function or a loop, we provide blocks of code for that programme. How can we tell which block is used for what? Statements in that block are "indented" to accomplish the task.

We make use of a number of leading whitespaces, which could be tabs or spaces, to establish the line's indentation level. Use one tab (or four spaces) for a single indent level is the general rule.

Let's look at a simple illustration to better grasp indentation levels.

[OceanofPDF.com](http://OceanofPDF.com)



```
// Edcorner Learning Python Concepts

def foo_bar():
    print("Hello!")
    if True:
        print("Bye")
    else:
        print("See you soon")

print("Indentation in Python")
```

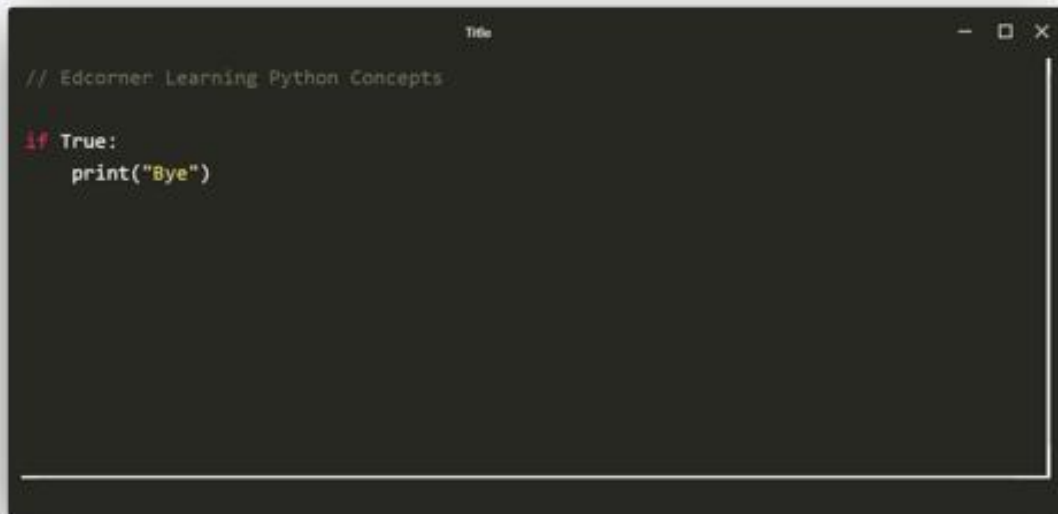
Instead of discussing the code and what it does, let's concentrate on the indentation levels.

"def foo\_bar():"

Every line of code that is a part of the function must be indented at least one level because it serves as the function's starting point. Keep in mind that the print and if statements both have at least one level of indentation. What about the final print statement, though? It is not a component of the function because, as can be seen, it is indented by 0 levels.

Let's now move into the function block, which consists of all the statements indented at least one level below the function starting.

Because the first print statement appears solely under the function and not in any other loops or conditions, it is indented by one level. The second print statement, in contrast, follows the if statement.



```
// Edcorner Learning Python Concepts

if True:
    print("Bye")
```



The if statement is only one level indented, but each block of code that follows it should be written one level deeper than the if statement. The else statement is similar.

Now, there are precise guidelines we must adhere to when indenting code in Python. As follows:

The backslash ("\") character cannot be used to divide an indentation into numerous lines.

If you attempt to indent the first line of code in Python, an IndentationError will be raised. The initial line of code cannot be indented.

[OceanofPDF.com](http://OceanofPDF.com)

It is advised to maintain the same preferred spacing option throughout your code if you have indented it with a tab or space. To avoid using the incorrect indentation, do not mix tabs and whitespaces when doing this.

For the first level of indentation, it is advisable to use 1 tab (or 4 whitespaces), and for subsequent levels, add 4 more whitespaces (or 1 more tab).

Python code that has been indented seems more aesthetically pleasing and organised. It provides you with a thorough comprehension of the code at just one glance. Additionally, the indentation guidelines are straightforward, and most integrated development environments (IDEs) will do the indenting for you as you write code.

One drawback of indentation is that it can be time-consuming to correct even a single line of indentation errors if your code is lengthy and uses high indentation levels.

To make sure you completely understand, let's look at a few indentation errors.

```
Title
-- □ ×

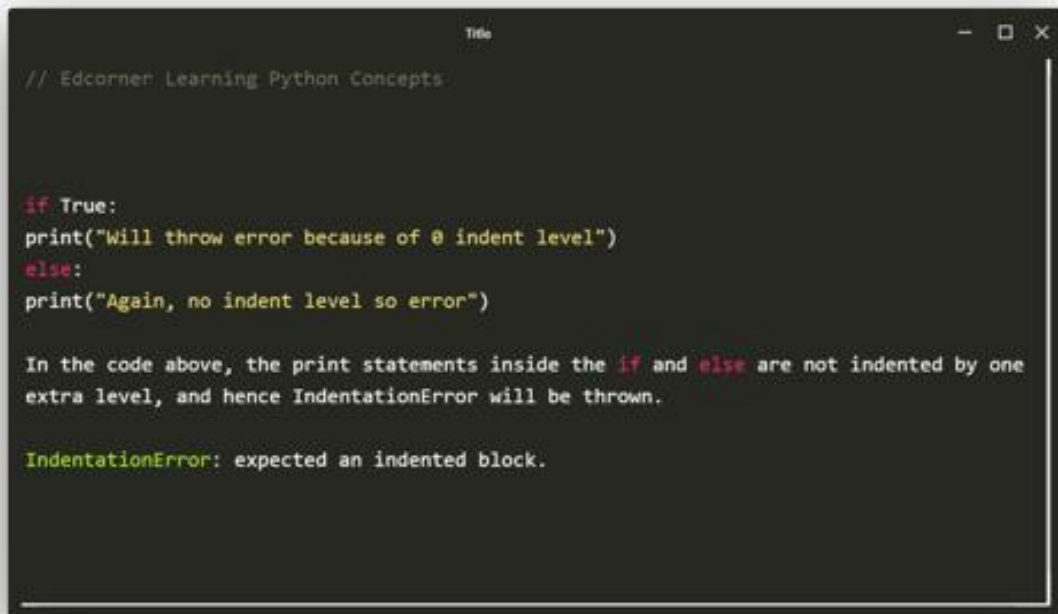
// Edcorner: Learning Python Concepts

>>>     y = 9
File "<stdin>", line 1
    y = 9
IndentationError: unexpected indent
>>>

As discussed above, since we cannot indent the first line, and Indentation Error is
thrown.

if True:
    print("Inside True")
    print("Will throw error because of extra whitespace")
else:
    print("False")

It will throw an IndentationError because the second print statement is written with
one level indent, but it has extra whitespace, which is not allowed in Python.
```



```
// Edcorner Learning Python Concepts

if True:
print("Will throw error because of 0 indent level")
else:
print("Again, no indent level so error")
```

In the code above, the print statements inside the `if` and `else` are not indented by one extra level, and hence `IndentationError` will be thrown.

`IndentationError: expected an indented block.`

## Syntax Identifiers in Python

We use "identifiers" to refer to variables, functions, classes, and modules when we discuss them.

Describe identifiers.

It serves as a name for anything to set it apart from other code entities.

We now have some name rules, which are the same as those for indentation.

1. The identification will include both lowercase (a-z) and capital (A-Z) letters, integers (0–9), underscore (\_), or both.
2. A digit cannot begin the Identifier.
3. Any reserved terms or keywords
4. Special characters or symbols cannot be used in the identification.
5. There is no restriction on the identifier's length, which is flexible.
6. Keep in mind that since Python is case-sensitive, var and VAR are different identifiers.

Correct naming examples include: var, Robo, python 001, etc.

Inappropriate names include 95learning, hel!ov, etc.

[OceanofPDF.com](http://OceanofPDF.com)

Here are some instances of variable identifiers:

var = 210

My var = 30

My var344 = 80

STRING = "EDCORNER" , fLoat = 3.144

**Examples of functions include:**




```
// Edcorner Learning Python Concepts

def my_function():
    print("Valid")

def My_function():
    print("Valid")

def Myfunction3():
    print("Valid")

def FUNCTION():
    print("Valid")
```



```
// Edcorner Learning Python Concepts

To check if your name is a valid identifier or not, you can use the inbuilt function
"isidentifier()".

It can be used as :

print("var".isidentifier())
print("@var".isidentifier())

True
False
```

The first one will display True because it is a legitimate identifier, whereas the second one will print False since it is an invalid one.

But there's a catch to this function. No keyword may be used to determine whether or not an identifier is legitimate. Even if you are aware that keywords cannot be used as identifiers, this function will display True for any term. As a result, this function carries a warning.

## Python Keywords

They are special-purpose terms that should only be employed in certain circumstances and not as identifiers.



```
// Edcorner Learning Python Concepts

To view all the keywords your version of python supports, you can open your script
editor or IDE or write the following two commands on your terminal to list all the
keywords.

import keyword
print(keyword.kwlist)

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

## Variables in Python

We utilise containers called variables to hold any data values. A variable can be declared in Python without using a specific command. Once a value is assigned to a variable, it is considered to have existed.

For instance:



```
y = 3
```

```
x = "Edcorner"
```

```
print(x)
```

```
print(y)
```

It will give us the output –

**Edcorner**

**3**

The majority of programming languages require you to declare variables with a specific type, meaning the sort of value they store. It may be an int, a char, etc. But Python does not need this. By assigning an integer value, you can declare a variable as an integer and later convert it to any other type as a string (by assigning a string value).

[OceanofPDF.com](http://OceanofPDF.com)

## Python comments:

Let's first examine the requirement for comments and the significance of programme comments. Comments are an essential component of any programme. As docstrings or single-line comments, we can create multi-line comments.

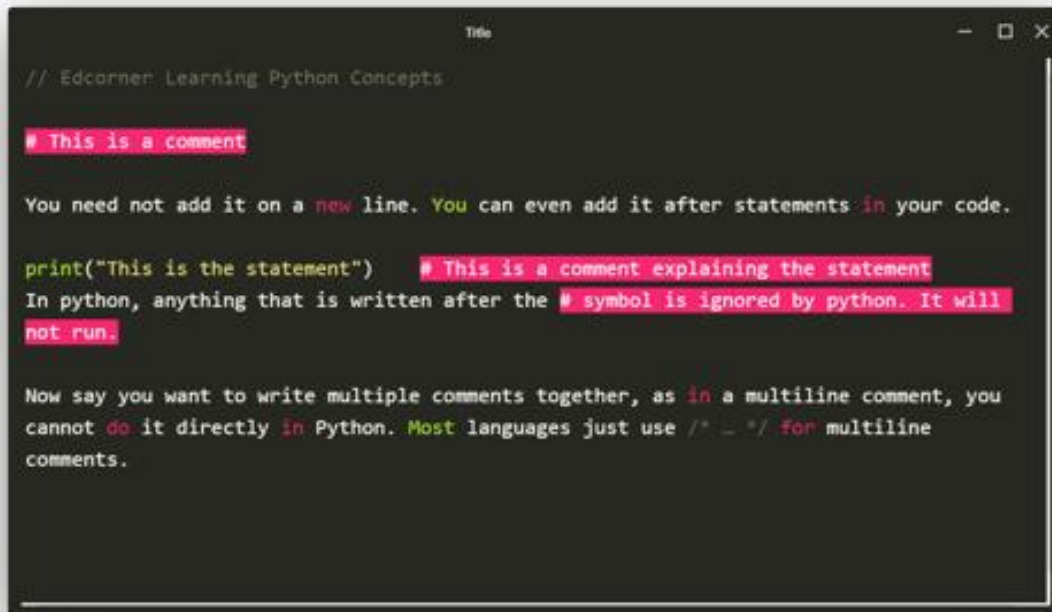
Reading your code when writing comments is crucial. It facilitates easier comprehension of the code. You could even add specific comments in between to review and update particular code blocks. You may need to modify your code even later, say, after six months. Your code would be easier for you to understand if you made comments throughout.

Imagine you are a member of a team. You don't need comments since you understand the code you've written. However, if you don't describe it in a few brief lines as comments, it will be difficult for others to understand your code.

Let's examine how we can incorporate comments into our Python scripts now that we are aware of their importance.

[OceanofPDF.com](http://OceanofPDF.com)

You must put the # symbol before your comment if it is to be a straightforward one-line comment. similar to this:



```
// Edcorner Learning Python Concepts

# This is a comment

You need not add it on a new line. You can even add it after statements in your code.

print("This is the statement") # This is a comment explaining the statement
In python, anything that is written after the # symbol is ignored by python. It will not run.

Now say you want to write multiple comments together, as in a multiline comment, you cannot do it directly in Python. Most languages just use /* */ for multiline comments.
```

```
Title
// Edcorner: Learning Python Concepts

## incorrect way to write multi line comments in Python
/*
so this is
just not
possible in Python
*/
However, you can do this :

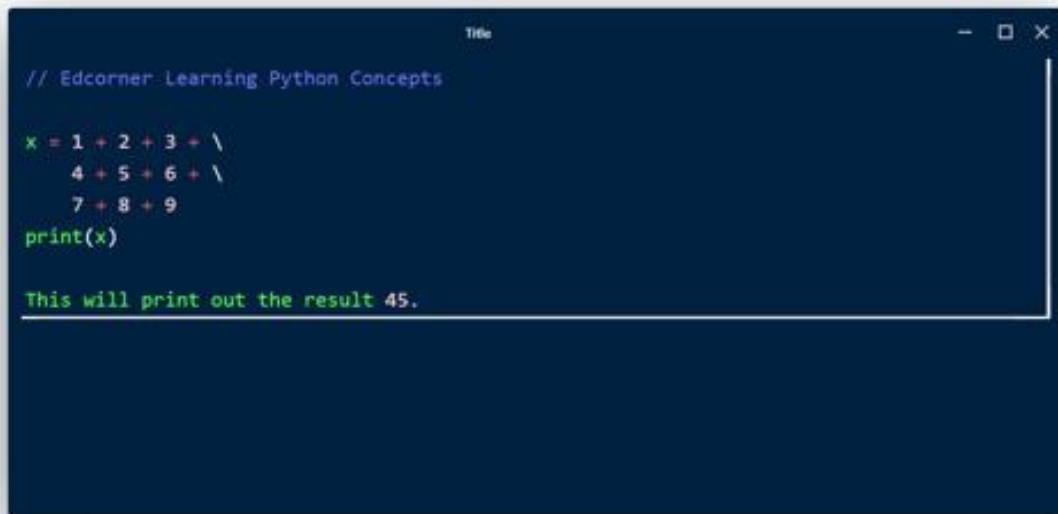
# this is
# one way to
# write multiline comments in python

If you don't want to constantly hit enter and add the # symbol at the start of every
line, you can make it a docstring by adding three double quotation marks.

"""
It makes multiline
comments in Python
much easier
"""
```

## Multiline statements:

In Python, you can use the line continuation character () to split a single statement into multiple lines. For example:



```
// Edcorner Learning Python Concepts

x = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
print(x)

This will print out the result 45.
```

You can also use parentheses to split a statement into multiple lines. For example:

[OceanofPDF.com](https://oceanofpdf.com)

A screenshot of a code editor window titled "Title". The code inside is: 

```
// Edcorner Learning Python Concepts

x = (1 + 2 + 3 +
    4 + 5 + 6 +
    7 + 8 + 9)
print(x)

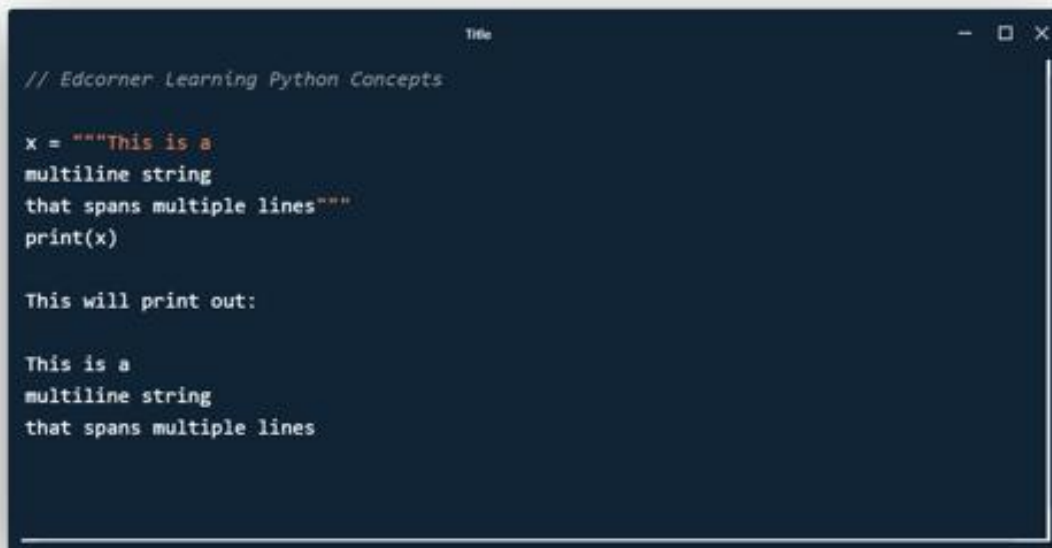
This will print out the result 45.
```

Note that when using the line continuation character or parentheses, you should not indent the continuation lines. Indenting the continuation lines can cause a syntax error.

**# This is incorrect and will cause a syntax error**

```
x = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
```

You can also use the triple quotes (either single or double) to split a statement into multiple lines. This is commonly used for writing multiline strings. For example:



```
// Edcorner Learning Python Concepts

x = """This is a
multiline string
that spans multiple lines"""
print(x)

This will print out:

This is a
multiline string
that spans multiple lines
```

Triple quotes can also be used for writing multiline comments. For example:

```
"""
```

**This is a multiline comment.**

**It can span multiple lines.**

```
"""
```

Note that when using triple quotes, you can indent the continuation lines if you want to, as it will not cause a syntax error.

## Blank lines in python:

In Python, a blank line is a line that contains only whitespace characters (spaces, tabs, etc.) and is not considered to be a statement. You can use

blank lines to improve the readability of your code by separating blocks of code into logical sections.

For example:



```
// Edcorner Learning Python Concepts

def greet(name):
    print("Hello,", name)

print("Welcome to my program!")

greet("Edcorner")
greet("Learning")
```



In this example, the blank line between the greet function definition and the first print statement helps to visually separate the function definition from the rest of the code. Similarly, the blank line between the two calls to the greet function helps to visually separate the two calls.

Blank lines are generally not required in Python, but they can be used to improve the readability of your code. You can use as many blank lines as you like, as long as they are not inside a block of code that requires a specific indentation level (such as a function definition or a for loop).

[OceanofPDF.com](http://OceanofPDF.com)

## Waiting for user:

There are a few ways you can pause your Python program and wait for user input.

One way to do this is to use the input function. The input function reads input from the user and returns it as a string. You can use it to pause the program and wait for the user to enter some text. For example:



```
// Edcorner Learning Python Concepts

name = input("Enter your name: ")
print("Hello, " + name)

This will print a prompt asking the user to enter their name, and then it will print
"Hello, [name]" once the user has entered their name and pressed the Enter key.
```

Another way to pause the program and wait for user input is to use the raw\_input function (in Python 2) or the input function with the eval function (in Python 3). For example:



```
// Edcorner Learning Python Concepts

# Python 2
age = int(raw_input("Enter your age: "))

# Python 3
age = int(input("Enter your age: "))
```

This will print a prompt asking the user to enter their age, and then it will store the age as an integer once the user has entered it and pressed the Enter key.

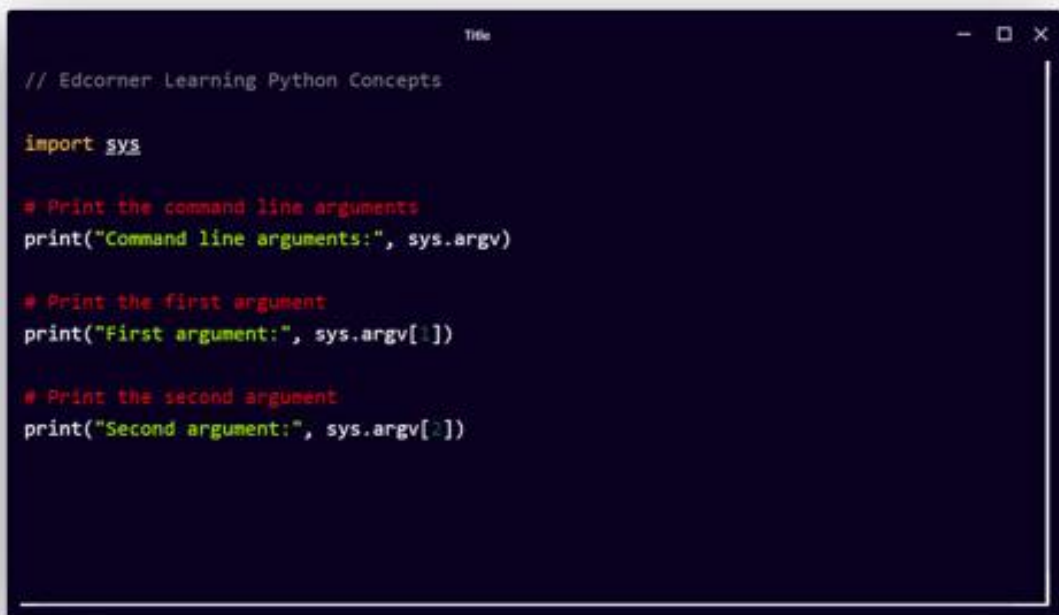
[OceanofPDF.com](https://oceanofpdf.com)

## Command Line Arguments:

In Python, you can use command line arguments to pass information to your program when you run it from the command line.

To access the command line arguments in your Python program, you can use the `sys` module and the `argv` attribute. The `argv` attribute is a list of strings that contains the command line arguments passed to the program. The first element of the list (`argv[0]`) is the name of the program itself, and the following elements (`argv[1]`, `argv[2]`, etc.) are the arguments passed to the program.

Here's an example of how you can use command line arguments in a Python program



```
// Edcorner Learning Python Concepts

import sys

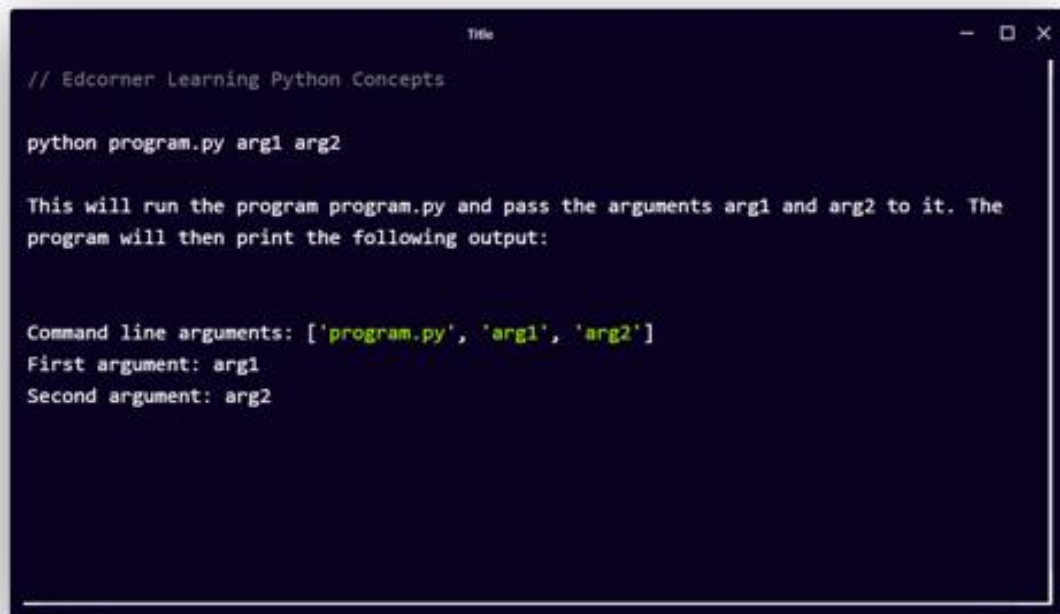
# Print the command line arguments
print("Command line arguments:", sys.argv)

# Print the first argument
print("First argument:", sys.argv[1])

# Print the second argument
print("Second argument:", sys.argv[2])
```

To run this program and pass some command line arguments, you can use the following command:

[OceanofPDF.com](http://OceanofPDF.com)



```
// Edcorner Learning Python Concepts

python program.py arg1 arg2

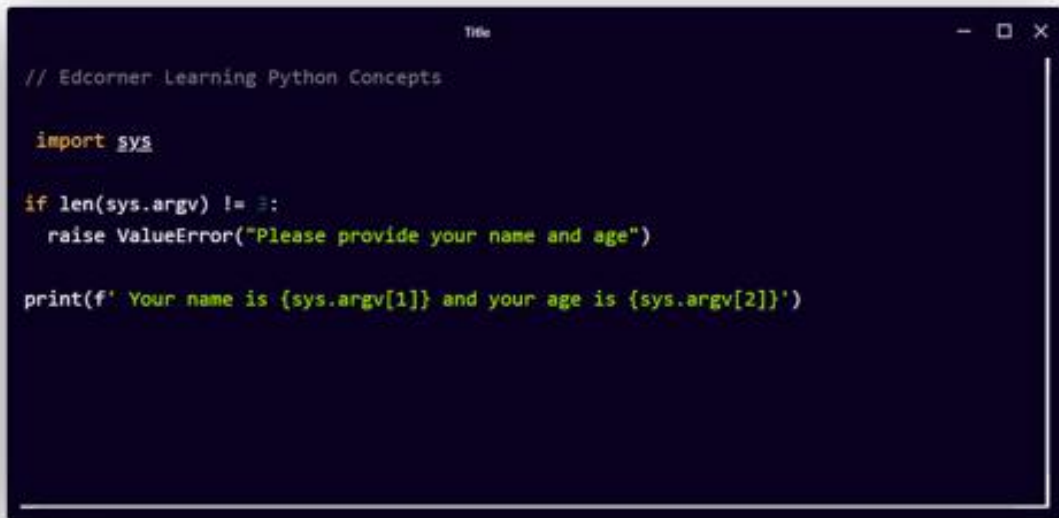
This will run the program program.py and pass the arguments arg1 and arg2 to it. The
program will then print the following output:

Command line arguments: ['program.py', 'arg1', 'arg2']
First argument: arg1
Second argument: arg2
```

## Sys module:

The sys module in Python provides access to various functions and variables that are used to interact with the interpreter. It is a built-in module, so you don't have to install it separately.

Here are some examples of how you can use the sys module in your Python programs:



```
// Edcorner Learning Python Concepts

import sys

if len(sys.argv) != 3:
    raise ValueError("Please provide your name and age")

print(f' Your name is {sys.argv[1]} and your age is {sys.argv[2]}')
```

In this code, the import sys statement imports the sys module, which provides access to various functions and variables that are used to interact with the Python interpreter.

The sys.argv attribute is a list of strings that contains the command line arguments passed to the program. The first element of the list (sys.argv[0]) is the name of the program itself, and the following elements (sys.argv[1], sys.argv[2], etc.) are the arguments passed to the program.

The if statement checks whether the number of command line arguments (`len(sys.argv)`) is not equal to 3. If this is the case, it raises a `ValueError` with the message "Please provide your name and age".

[OceanofPDF.com](http://OceanofPDF.com)



If the number of command line arguments is equal to 3, the print statement prints a message with the name and age provided as arguments. The f string syntax (f'...') allows you to embed expressions inside a string, and the {} placeholders are replaced with the values of the expressions.

For example, if you run the program with the following command:

**python program.py Edcorner 32**

**The program will print the following output:**

**Your name is Edcorner and your age is 32**

[OceanofPDF.com](https://oceanofpdf.com)

# What is Indentation in Python

In Python, indentation is used to define blocks of code. A block of code is a group of statements that are executed together as a unit. Indentation is used to indicate which statements belong to the same block of code.

For example, consider the following code:

## Code

```
x = 10
```

```
if x > 5:
```

```
    print("x is greater than 5")
```

```
    print("This is inside the if block")
```

```
print("This is outside the if block")
```

In this code, the if statement and the two print statements that are indented below it form a block of code. The print statement that is not indented below the if statement is outside of the block.

The indentation level in Python is determined by the number of spaces or tabs that are used at the beginning of a line. The indentation level must be consistent within a block of code.

For example, the following code is correct:

## Code

```
x = 10
```

```
if x > 5:
```

```
    print("x is greater than 5")
```

```
    print("This is inside the if block")
```

```
print("This is outside the if block")
```

But the following code is incorrect because the indentation level is not consistent within the block of code:

### **Code**

```
x = 10
```

```
if x > 5:
```

```
    print("x is greater than 5")
```

```
    print("This is inside the if block") # Incorrect indentation
```

```
print("This is outside the if block")
```

[OceanofPDF.com](http://OceanofPDF.com)

Indentation is an important aspect of Python's syntax, and it is used to define blocks of code such as function definitions, for loops, and if statements. It is important to pay attention to the indentation level in your code to avoid syntax errors.

[OceanofPDF.com](http://OceanofPDF.com)

## Python Variables:

In Python, there are several different types of variables that you can use to store different types of data. Here are some common types of variables in Python:

### **Integer**

An integer is a whole number that can be positive, negative, or zero. In Python, you can create an integer variable by assigning an integer value to it. For example:

Code

```
x = 10
```

```
y = -5
```

```
z = 0
```

In this example, x is an integer variable that stores the value 10, y is an integer variable that stores the value -5, and z is an integer variable that stores the value 0.

### **Float**

A float is a number with a decimal point. In Python, you can create a float variable by assigning a float value to it. For example:

Code

```
x = 3.14
```

```
y = -2.71828
```

```
z = 0.0
```

In this example, x is a float variable that stores the value 3.14, y is a float variable that stores the value -2.71828, and z is a float variable that stores the value 0.0.

[OceanofPDF.com](http://OceanofPDF.com)

## String

In Python, a string is a sequence of characters. You can create a string variable by assigning a string value to it. You can use single quotes (') or double quotes (") to define a string.

Here's an example of how you can create and use string variables in Python:

code

```
# Assign a string value to a variable
```

```
x = "hello"
```

```
# Print the value of the variable
```

```
print(x) # Output: "hello"
```

```
# Concatenate two strings and assign the result to a new variable
```

```
y = x + " edcorner"
```

```
# Print the value of the new variable
```

```
print(y) # Output: "hello edcorner"
```

```
# Access individual characters in the string using indexing
```

```
print(x[0]) # Output: "h"
```

```
print(x[1]) # Output: "e"
```

```
# Find the length of the string using the len() function
```

```
print(len(x)) # Output: 5
```

```
# Use slicing to extract a substring
```

```
print(x[1:4]) # Output: "ell"
```

# Use string methods to manipulate the string

```
print(x.upper()) # Output: "HELLO"
```

```
print(x.lower()) # Output: "hello"
```

```
print(x.capitalize()) # Output: "Hello"
```

Strings are one of the most commonly used types of variables in Python, and they are often used to store and manipulate text data.

## **Local python variable**

In Python, a local variable is a variable that is defined within a function or a block of code, and it is only accessible within that function or block of code. Local variables are created when the function or block of code is executed, and they are destroyed when the function or block of code ends.

Here's an example of how you can create and use local variables in Python:

code

```
def greet(name):
```

```
    # This is a local variable
```

```
    greeting = "Hello " + name + "!"
```

```
    # Print the value of the local variable
```

```
    print(greeting)
```

```
# Call the function and pass a string argument
```



```
greet("Edcorner ") # Output: "Hello Edcorner!"
```

```
# Try to access the local variable outside of the function
```

```
print(greeting) # This will cause a NameError
```

In this example, the greet function defines a local variable greeting and assigns a string value to it. The greeting variable is only accessible within the function, and if you try to access it outside of the function, you will get a NameError because the variable does not exist in the global scope.

Local variables are useful for storing temporary data that is only needed within a specific function or block of code. They are also useful for avoiding name conflicts with global variables that have the same name.

[OceanofPDF.com](http://OceanofPDF.com)

## **global python variable**

In Python, a global variable is a variable that is defined outside of any function or block of code, and it is accessible from anywhere in the program. Global variables are created when the program starts, and they are destroyed when the program ends.

Here's an example of how you can create and use global variables in Python:

code

```
# This is a global variable
```

```
x = 10
```

```
def print_x():
```

```
    # Print the value of the global variable
```

```
    print(x)
```

```
# Call the function
```

```
print_x() # Output: 10
```

```
# Modify the value of the global variable
```

```
x = 20
```

```
# Call the function again
```

```
print_x() # Output: 20
```

In this example, the `x` variable is defined outside of any function or block of code, so it is a global variable. The `print_x` function accesses and prints the value of the `x` variable, and the value of the `x` variable can be modified outside of the function.

Global variables are useful for storing data that needs to be accessed and shared by multiple functions or blocks of code in your program. However, it is generally considered good programming practice to avoid using global variables whenever possible, as they can make your code harder to understand and maintain.

[OceanofPDF.com](http://OceanofPDF.com)

# Object References in python

In Python, an object reference is a way to access an object in memory. When you create an object in Python, the interpreter assigns a unique identity to the object and stores it in memory. You can access the object using a reference to its identity.

For example, consider the following code:

Code

```
x = 10
```

```
y = x
```

In this code, x is a variable that stores the integer value 10. When you assign the value of x to y, Python creates a new reference to the object in memory. Both x and y are now references to the same object, and they both point to the same identity in memory.

You can use the is operator to compare the identities of two objects. If the objects have the same identity, the is operator will return True, and if the objects have different identities, the is operator will return False.

**For example:**

Code

```
x = 10
```

```
y = x
```

```
# x and y are references to the same object
```

```
print(x is y) # Output: True
```

**# Create a new object with the same value as x**

**z = int(10)**

**# x, y, and z are all references to different objects**

**print(x is z) # Output: False**

**print(y is z) # Output: False**

## Object Identity in python

In Python, the `id()` function returns the identity of an object. The identity of an object is a unique integer that is assigned to the object when it is created, and it remains the same for the lifetime of the object.

You can use the `id()` function to determine whether two variables refer to the same object or not. For example:

code

`x = 10`

`y = 10`

**# Print the identities of x and y**

**print(id(x)) # Output: 140723586857648**

**print(id(y)) # Output: 140723586857648**

**# x and y have the same value, but they are different objects**

```
print(x is y) # Output: False
```

```
# Create a new object with the same value as x
```

```
z = int(10)
```

```
# Print the identity of z
```

```
print(id(z)) # Output: 140723586857648
```

```
# z has the same value as x and y, and it is the same object as x and y
```

```
print(x is z) # Output: True
```

```
print(y is z) # Output: True
```

[OceanofPDF.com](http://OceanofPDF.com)

In this example, x and y are both variables that store the integer value 10. However, they are different objects, as shown by the different identities returned by the `id()` function. On the other hand, z is a variable that stores the same integer value as x and y, and it is the same object as x and y, as shown by the fact that the `id()` function returns the same identity for all three variables.

You can use the `is` operator to compare the identities of two objects. If the objects have the same identity, the `is` operator will return `True`, and if the objects have different identities, the `is` operator will return `False`.

[OceanofPDF.com](http://OceanofPDF.com)

## What are Python Keywords

In Python, keywords are reserved words that have a specific meaning in the language and cannot be used as variable names or identifiers.

Here is a list of all the keywords in Python:

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

You should avoid using these keywords as variable names or identifiers in your code, as doing so will cause a syntax error.

[OceanofPDF.com](https://oceanofpdf.com)



## **Value Keywords in python**

In Python, value keywords are reserved words that represent fixed, unchangeable values. There are two value keywords in Python:

### **True**

The True keyword represents the boolean value True. It is equivalent to the integer value 1 and the float value 1.0.

### **False**

The False keyword represents the boolean value False. It is equivalent to the integer value 0 and the float value 0.0.

Here's an example of how you can use the True and False keywords in Python:

code

```
# Assign the value True to a variable
```

```
x = True
```

```
# Print the value of the variable
```

```
print(x) # Output: True
```

```
# Check the type of the variable
```

```
print(type(x)) # Output: <class 'bool'>
```

# Use the True keyword in a condition

if x:

print("x is True") # Output: "x is True"

# Assign the value False to a variable

y = False

# Print the value of the variable

print(y) # Output: False

# Check the type of the variable

print(type(y)) # Output: <class 'bool'>

# Use the False keyword in a condition

if not y:

print("y is False") # Output: "y is False"

[OceanofPDF.com](http://OceanofPDF.com)

## Operator keyword in python:

In Python, operator keywords are reserved words that represent certain operations or computations.

and    AND, Intersection( $\wedge$ )

or     OR, Union( $\vee$ )

not    NOT, Negation ( $\neg$ )

in     CONTAINS, ( $\in$ )

is     Checking Identity

some examples:

### **and**

The and keyword represents the logical operation of conjunction. It is used to evaluate the truth value of a compound expression.

The and operator returns True if both operands are True, and it returns False if either operand is False.

Here's an example of how you can use the and keyword in Python:

code

```
# Evaluate the truth value of a compound expression
```

```
x = True
```

```
y = False
```

```
# The expression is True only if both operands are True
```

```
z = x and y
```

```
# Print the value of the expression
```

```
print(z) # Output: False
```

[OceanofPDF.com](http://OceanofPDF.com)

## **or**

The or keyword represents the logical operation of disjunction. It is used to evaluate the truth value of a compound expression.

The or operator returns True if either operand is True, and it returns False if both operands are False.

Here's an example of how you can use the or keyword in Python:

code

```
# Evaluate the truth value of a compound expression
```

```
x = True
```

```
y = False
```

```
# The expression is True if either operand is True
```

```
z = x or y
```

```
# Print the value of the expression
```

```
print(z) # Output: True
```

## Control Flow Keywords in python

In Python, control flow keywords are reserved words that are used to control the flow of execution in a program. Here is a list of control flow keywords in Python:

### **if**

The if keyword is used to create a conditional statement. A conditional statement is a block of code that is executed only if a certain condition is True.

Here's an example of how you can use the if keyword in Python:

Code

```
x = 10
```

```
# Check if x is greater than 5
```

```
if x > 5:
```

```
    # Execute this block if the condition is True
```

```
    print("x is greater than 5")
```

```
# Execute this block regardless of the condition
```

```
print("This line is always executed")
```

[OceanofPDF.com](http://OceanofPDF.com)

## **else**

The else keyword is used in conjunction with the if keyword to create an else block. An else block is a block of code that is executed if the condition in the if block is False.

Here's an example of how you can use the else keyword in Python:

Code

```
x = 10
```

```
# Check if x is greater than 5
```

```
if x > 5:
```

```
    # Execute this block if the condition is True
```

```
    print("x is greater than 5")
```

```
else:
```

```
    # Execute this block if the condition is False
```

```
    print("x is not greater than 5")
```

```
# Execute this block regardless of the condition
```

```
print("This line is always executed")
```

[OceanofPDF.com](http://OceanofPDF.com)

## **elif**

The elif keyword is used in conjunction with the if keyword to create an elif block. An elif block is a block of code that is executed if a certain condition is True, and it is only considered if the condition in the preceding if block is False.

Here's an example of how you can use the elif keyword in Python:

Code

```
x = 10

# Check if x is greater than 5
if x > 5:
    # Execute this block if the condition is True
    print("x is greater than 5")
# Check if x is equal to 5
elif x == 5:
    # Execute this block if the condition is True
    print("x is equal to 5")
else:
    # Execute this block if all the conditions are False
    print("x is not greater than 5 and x is not equal to 5")

# Execute this block regardless of the conditions
print("This line is always executed")
```



[OceanofPDF.com](http://OceanofPDF.com)

## Iteration Keywords

In Python, iteration keywords are reserved words that are used to control the flow of execution in a loop. There are two iteration keywords in Python:

### **break**

The break keyword is used to exit a loop prematurely. When the break statement is encountered inside a loop, the loop is immediately terminated, and the control of the program is transferred to the next line of code after the loop.

Here's an example of how you can use the break keyword in Python:

Code

```
# Print the numbers from 1 to 10
for i in range(1, 11):
    # Exit the loop if i is equal to 5
    if i == 5:
        break
    # Print the value of i
    print(i)

# Execute this line after the loop has finished
print("The loop has finished")
```

In this example, the loop will iterate over the range 1 to 10, and it will print the value of i at each iteration. However, when i becomes equal to 5, the break statement will be encountered, and the loop will be terminated immediately.

[OceanofPDF.com](http://OceanofPDF.com)

## **continue**

The continue keyword is used to skip the remainder of the current iteration of a loop and move on to the next iteration. When the continue statement is encountered inside a loop, the control of the program is transferred to the beginning of the next iteration, skipping the rest of the code in the current iteration.

Here's an example of how you can use the continue keyword in Python:

Code

```
# Print the numbers from 1 to 10
for i in range(1, 11):
    # Skip the remainder of the current iteration if i is even
    if i % 2 == 0:
        continue
    # Print the value of i
    print(i)
# Execute this line after the loop has finished
print("The loop has finished")
```

In this example, the loop will iterate over the range 1 to 10, and it will print the value of i at each iteration. However, if i is even, the continue statement will be encountered, and the control of the program will be transferred to the beginning of the next iteration, skipping the rest of the code in the current iteration.

In Python, for and while are looping constructs that are used to execute a block of code repeatedly until a certain condition is False. Here's a brief overview of how for and while loops work in Python:

## **for loop**

A for loop is used to iterate over a sequence of elements, such as a list or a string. The syntax for a for loop in Python is as follows:

Code

for variable in sequence:

    # Code block to be executed

Here's an example of how you can use a for loop in Python:

Code

# Print the numbers from 1 to 10

for i in range(1, 11):

    # Print the value of i

    print(i)

In this example, the for loop will iterate over the range 1 to 10, and it will print the value of i at each iteration.

[OceanofPDF.com](http://OceanofPDF.com)

## **while loop**

A while loop is used to execute a block of code repeatedly as long as a certain condition is True. The syntax for a while loop in Python is as follows:

Code

```
while condition:
```

```
    # Code block to be executed
```

Here's an example of how you can use a while loop in Python:

Code

```
# Initialize a counter
```

```
i = 0
```

```
# Execute the loop as long as i is less than 5
```

```
while i < 5:
```

```
    # Increment i by 1
```

```
    i += 1
```

```
    # Print the value of i
```

```
    print(i)
```

In this example, the while loop will execute the code block inside the loop as long as the value of *i* is less than 5. The value of *i* is incremented by 1 at the end of each iteration, so the loop will eventually terminate when *i* becomes greater than or equal to 5.

[OceanofPDF.com](http://OceanofPDF.com)

## Structural Keywords

def	def <name of function> (<parameters>):<body of code>	Defines the beginning of a function in python
class	class <name of class>(<extends>): <body of code>	Defines the beginning of a class in python
with	with <context manager> as <variable>: <body of code>	We use with to run codes within the context manager functions. Like: reading and writing of files
as	import <module> as <alias>	It is used to provide an alias to modules, functions, etc.
pass	def function()	Used with functions classes, if-else statements, etc. to indicate 'no – operation' taking place
lambda	lambda <arguments>:<function>	It is used to create a nameless function. It is an inline function that does not contain a return statement.

## Return Keywords



return	def <function>() <statement>	Returns the statement when this function is called
yield	<pre>&gt;&gt;&gt; def &lt;function&gt;(): ... yield &lt;statement1&gt; ... yield &lt;statement2&gt;&gt;&gt;&gt; test = &lt;function&gt;() &gt;&gt;&gt; next(test)&lt;statement1&gt; &gt;&gt;&gt; next(test) &lt;statement2&gt;</pre>	Through yield we can have numerous return statements which will perform sequentially using the in built next() function in python

## Import Keywords

import	import <module>	Imports all functions of the specified modules to be used in the python file's execution.
from	from <module> import <function>	The from keyword is used together with import to import specific functions from a module.
as	import <module> as <alias>	Import module with a alias

## Exception Handling Keywords

try	try: <statements>	Initiates try block where the statement is executed if there is no exception
except	try:<statement1>except <exception>: <statement2>	Specifies the type of exception that should occur and what subsequent statements should execute
raise	raise <exception>	This statement can be used to raise a particular exception to any block of code
finally	try: <statements>finally: <statements>	Finally defines a piece of code that should run no matter what
assert	assert <expression>	An <a href="#">assert statement</a> will result in performing no operation if the expression is executed as truthful, and it will raise an AssertionError if the expression is executed as False.

## Asynchronous Programming Keywords

async	async def <function>(<params>); <statements>	Indicates that the following function needs to run asynchronously
await	await <some async function call> OR <var> = await <some async function call>	It is used in asynchronous functions to specify a point in the function where control is given back to the event loop for other functions to run

## Variable Handling Keywords

del	del <variable>	The keyword is used to reset the variable to an unused one
global	global <variable>	The keyword specifies that the variable is pulled from global scope
nonlocal	nonlocal <variable>	Similar to global, this keyword specifies that the variable has been pulled from parent scope

## Python Identifiers:

In Python, an identifier is a name used to identify a variable, function, class, module, or other object. An identifier must follow certain rules in order to be considered valid in Python.

### Rules for defining Python identifiers

- An identifier must start with a letter (uppercase or lowercase) or an underscore (\_).
- An identifier cannot start with a number.
- An identifier can only contain letters, digits, and underscores (\_).
- An identifier is case-sensitive, so myVariable and myvariable are considered different identifiers.
- An identifier cannot be a reserved word.

### Examples of valid Python identifiers

my\_variable

myVariable1

\_private\_variable

### **Examples of invalid Python identifiers**

1st\_variable (starts with a number)

my-variable (contains a hyphen)

while (is a reserved word)

It's important to choose descriptive and meaningful names for your identifiers, as it will make your code easier to understand and maintain.

[OceanofPDF.com](http://OceanofPDF.com)

## Literals in python:

In Python, a literal is a value that represents itself and can be used directly in the code. There are several types of literals in Python, including:

### **Numeric literals**

A numeric literal is a value that represents a number. There are several types of numeric literals in Python, including:

int: An integer literal is a whole number, such as 10, -5, or 0.

float: A float literal is a number with a decimal point, such as 3.14, -2.718, or 0.0.

complex: A complex literal is a number with a real and an imaginary part, such as 3+4j, -1.5+2j, or 0j.

Here are some examples of numeric literals in Python:

Code

```
x = 10 # An integer literal
```

```
y = 3.14 # A float literal
```

```
z = 3+4j # A complex literal
```

[OceanofPDF.com](http://OceanofPDF.com)

## String literals

A string literal is a value that represents a sequence of characters, such as a word, a sentence, or a paragraph. In Python, you can define a string literal using either single quotes (') or double quotes (").

Here are some examples of string literals in Python:

Code

```
x = 'hello' # A string literal defined with single quotes
```

```
y = "edcorner" # A string literal defined with double quotes
```

```
z = "Hello,
```

```
edcorner!" # A multi-line string literal
```

[OceanofPDF.com](http://OceanofPDF.com)



## **Boolean literals**

A boolean literal is a value that represents a boolean (True or False) value. In Python, you can use the keywords True and False to define boolean literals.

Here are some examples of boolean literals in Python:

Code

```
x = True # A boolean literal
```

```
y = False # Another boolean literal
```

## Literal Collections in python

In Python, a literal collection is a data type that represents a group of values, such as a list, a tuple, or a dictionary. Here's a brief overview of the most commonly used literal collections in Python:

### List literals

A list literal is a data type that represents an ordered collection of values, which can be of any data type. In Python, you can define a list literal using square brackets ([]) and separating the elements with a comma (,).

Here's an example of a list literal in Python:

Code

```
x = [1, 2, 3, 4, 5] # A list of integers
y = ['Edcorner', 'Learning', 'cherry'] # A list of strings
z = [1, 2.5, 'three', [4, 5]] # A list of mixed data types
```

You can access the elements of a list using their indices, which start at 0 for the first element and increase by 1 for each subsequent element. You can also use negative indices to access the elements from the end of the list, with -1 being the index of the last element.

You can modify the elements of a list by assigning new values to their indices, and you can add new elements to a list using the `append()` method or by using the slicing notation with an assignment.

## Tuple literals

A tuple literal is a data type that represents an immutable (unchangeable) ordered collection of values, which can be of any data type. In Python, you can define a tuple literal using parentheses (()) and separating the elements with a comma (,).

Here's an example of a tuple literal in Python:

### Code

```
x = (1, 2, 3, 4, 5) # A tuple of integers  
y = ('Edcorner', 'Learning', 'cherry') # A tuple of strings  
z = (1, 2.5, 'three', (4, 5)) # A tuple of mixed data types
```

You can access the elements of a tuple using their indices, which work in the same way as in a list. However, you cannot modify the elements of a tuple, as they are immutable.

[OceanofPDF.com](http://OceanofPDF.com)

## Dictionary literals

A dictionary literal is a data type that represents an unordered collection of key-value pairs, where the keys are unique and the values can be of any data type. In Python, you can define a dictionary literal using curly braces ({} ) and separating the key-value pairs with a colon (:).

Here's an example of a dictionary literal in Python:

### Code

```
x = {'Edcorner': 1, 'Learning': 2, 'cherry': 3} # A dictionary of fruit and their quantities
```

```
y = {'x': 10, 'y': 20, 'z': 30} # A dictionary of coordinates
```

```
z = {1: 'one', 2: 'two', 3: 'three'} # A dictionary of numbers and their string representations
```

You can access the values of a dictionary using the keys, by using the indexing notation with the key in square brackets ([]). You can also modify the values of a dictionary by assigning new values to the keys, and you can add new key-value pairs to a dictionary using the assignment notation with the key in square brackets ([]).

## Set Literals

A set literal is a data type that represents an unordered collection of unique values, which can be of any immutable data type. In Python, you can define a set literal using curly braces (`{}`) and separating the elements with a comma (`,`).

Here's an example of a set literal in Python:

### Code

```
x = {1, 2, 3, 4, 5} # A set of integers
y = {'Edcorner', 'Learning', 'cherry'} # A set of strings
z = {1, 2.5, 'three', (4, 5)} # A set of mixed data types
```

You can access the elements of a set using the `in` operator, which returns `True` if the element is in the set and `False` otherwise. You can also modify a set by adding or removing elements using the `add()` and `remove()` methods, respectively.

A set is an unordered collection, so the elements of a set are not accessed using indices like in a list or a tuple. Instead, you can use the `union()`, `intersection()`, and `difference()` methods to perform set operations, such as finding the elements that are in both sets, the elements that are in one set but not the other, and so on.

## Comments in Python and its types

In Python, a comment is a piece of text that is ignored by the interpreter and is used to annotate the code for the benefit of other developers. There are two types of comments in Python:

### Single-line comments

A single-line comment is a line of text that starts with a pound sign (#) and continues until the end of the line. Single-line comments are used to provide short explanations or to disable a line of code temporarily.

Here's an example of a single-line comment in Python:

Code

```
# This is a single-line comment
```

```
x = 10 # This is also a single-line comment
```

[OceanofPDF.com](http://OceanofPDF.com)

## Multi-line comments

A multi-line comment is a block of text that starts with a triple quotation mark (""" or """) and ends with the same quotation mark. Multi-line comments are used to provide longer explanations or to disable multiple lines of code temporarily.

Here's an example of a multi-line comment in Python:

Code

```
"""
```

```
This is a multi-line comment.
```

```
It can span multiple lines and can be used to provide longer explanations.
```

```
"""
```

```
x = 10 # This is not a part of the multi-line comment
```

It's a good practice to include comments in your code to make it easier for other developers to understand what you are trying to accomplish.

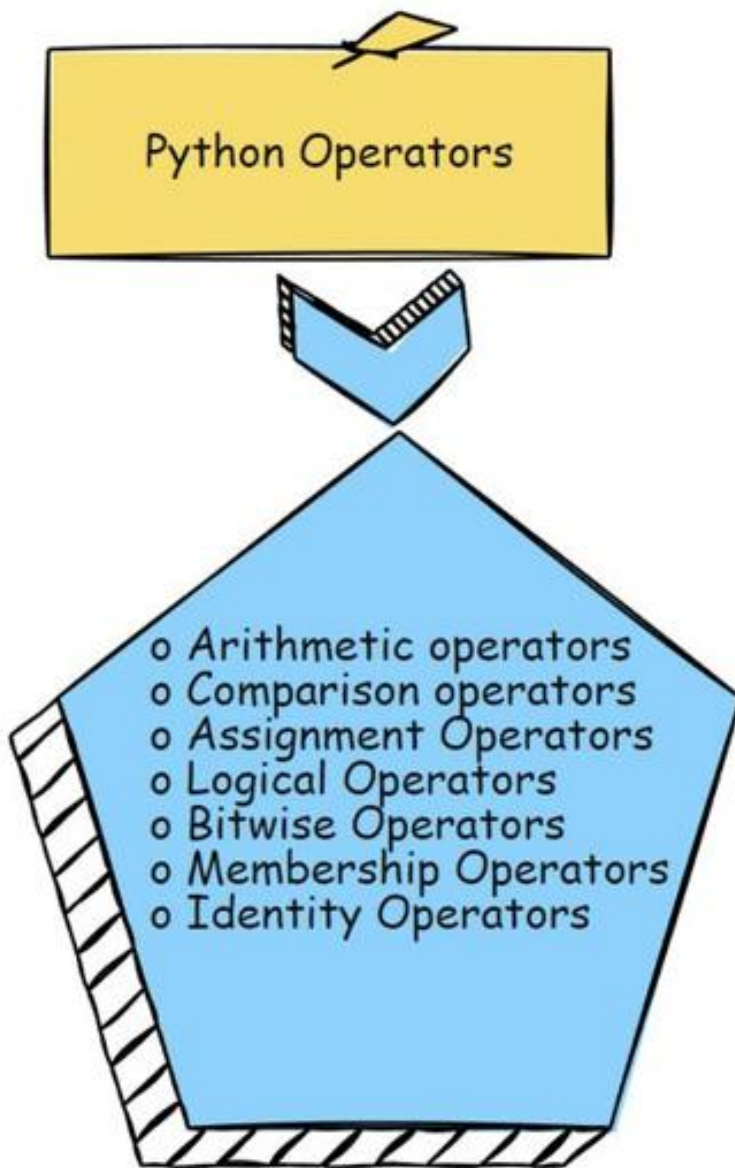
However, you should avoid adding too many comments, as it can make your code less readable.

[OceanofPDF.com](http://OceanofPDF.com)

# Operators in python:

## Python operator

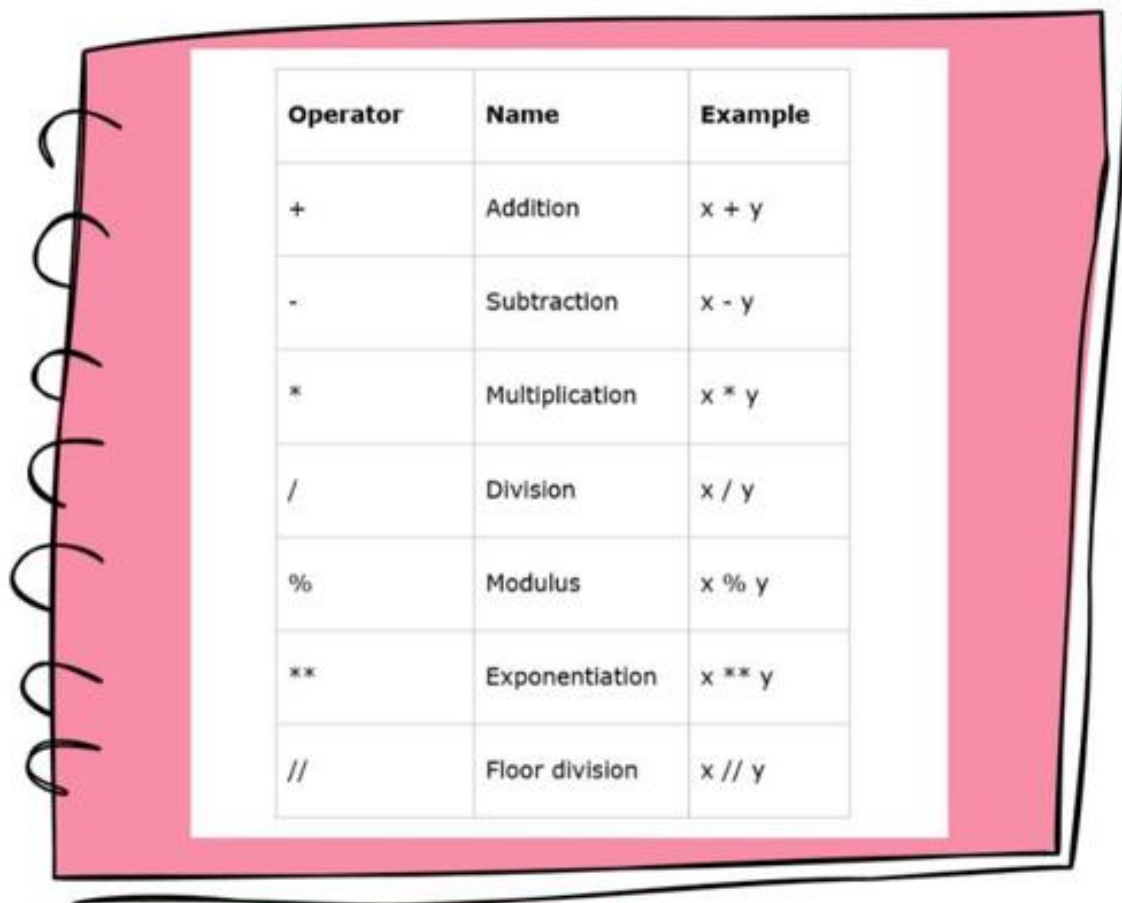
The operator is a symbol that performs a certain operation between two operands, according to one definition. The following list of operators is provided by Python and is described in detail:





## Arithmetic operators


Arithmetic operators are used with numeric values to perform common mathematical operations



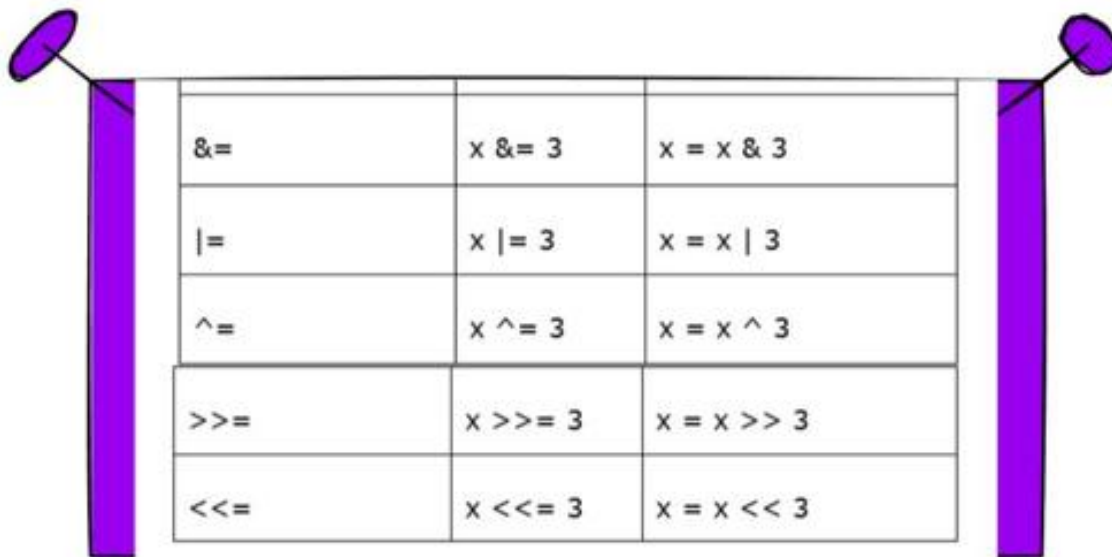
Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	$x / y$
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$

## Python Assignment Operators:

Assignment operators are used to assign values to variables:



Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3



&=	x &= 3	x = x & 3
=	x  = 3	x = x   3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

## **Python Comparison Operators**

Comparison operators are used to compare two values:

## Python Comparison Operators



Operator	Name	Example
<code>==</code>	Equal	<code>x == y</code>
<code>!=</code>	Not equal	<code>x != y</code>
<code>&gt;</code>	Greater than	<code>x &gt; y</code>
<code>&lt;</code>	Less than	<code>x &lt; y</code>
<code>&gt;=</code>	Greater than or equal to	<code>x &gt;= y</code>
<code>&lt;=</code>	Less than or equal to	<code>x &lt;= y</code>

## Python Logical Operators

Logical operators are used to combine conditional statements:

Python Logical Operators		
Operator	Description	Example
<b>and</b>	Returns True if both statements are true	<code>x &lt; 5 and x &lt; 10</code>
<b>or</b>	Returns True if one of the statements is true	<code>x &lt; 5 or x &lt; 4</code>
<b>not</b>	Reverse the result, returns False if the result is true	<code>not(x &lt; 5 and x &lt; 10)</code>

## Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

## Python Identity Operators



Operator	Description	Example
<b>is</b>	Returns True if both variables are the same object	x is y
<b>is not</b>	Returns True if both variables are not the same object	x is not y



## Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Python Membership Operators		
Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

## Python Bitwise Operators

Python Bitwise Operators		
Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

## Python Input and Output

In Python, you can use the `input()` function to read input from the user, and you can use the `print()` function to output data to the console. Here's a brief overview of how you can use these functions in Python:

### Reading input from the user

You can use the `input()` function to read input from the user and store it in a variable. The `input()` function reads a line of text from the standard input (usually the keyboard) and returns it as a string.

Here's an example of how you can use the `input()` function in Python:

Code

```
name = input('Enter your name: ') # Read a string from the user and store it
in the 'name' variable
```

```
age = int(input('Enter your age: ')) # Read a string from the user, convert it
to an integer, and store it in the 'age' variable
```

### **Outputting data to the console**

You can use the `print()` function to output data to the console. The `print()` function takes one or more arguments and prints them to the standard output (usually the console) separated by spaces.

Here's an example of how you can use the `print()` function in Python:

Code

```
name = 'Edcorner'
```

```
age = 25
```

```
print('Hello,', name) # Prints 'Hello, Edcorner'
```

```
print('You are', age, 'years old') # Prints 'You are 25 years old'
```

```
print(f'Hello, {name}. You are {age} years old') # Prints 'Hello, Edcorner.
You are 25 years old' (using f-strings)
```

You can also use the `sep` and `end` keyword arguments of the `print()` function to customize the separator between the arguments and the string that is printed after the arguments.

# Implicit Type Conversion in Python

In Python, implicit type conversion (also called coercion) is the automatic conversion of a value of one data type to another data type when it is used in an operation or function that requires a different data type.

Here are some examples of implicit type conversion in Python:

Code

```
# Implicit conversion of integers to floats
```

```
x = 10
```

```
y = 5.0
```

```
z = x + y # z will be 15.0 (a float)
```

```
# Implicit conversion of strings to integers
```

```
a = '10'
```

```
b = '5'
```

```
c = a + b # c will be '105' (a string)
```

```
d = int(a) + int(b) # d will be 15 (an integer)
```

```
# Implicit conversion of lists to tuples
```

```
e = [1, 2, 3]
```

```
f = tuple(e) # f will be (1, 2, 3) (a tuple)
```

In Python, implicit type conversion only occurs when the data types are compatible, and the conversion is lossless (i.e., the original value can be

recovered). For example, you can convert an integer to a float or a string, but you cannot convert a float to an integer (because the decimal part would be lost).

[OceanofPDF.com](http://OceanofPDF.com)

# Explicit Type Conversion in Python

In Python, explicit type conversion (also called type casting) is the process of converting a value of one data type to another data type by explicitly specifying the target data type.

You can use the built-in type conversion functions in Python to perform explicit type conversion, such as `int()`, `float()`, `str()`, `list()`, and `tuple()`. Here are some examples of explicit type conversion in Python:

Code

```
# Explicit conversion of integers to floats
```

```
x = 10
```

```
y = float(x) # y will be 10.0 (a float)
```

```
# Explicit conversion of floats to integers
```

```
a = 10.5
```

```
b = int(a) # b will be 10 (an integer)
```

```
# Explicit conversion of strings to integers
```

```
c = '10'
```

```
d = int(c) # d will be 10 (an integer)
```

```
# Explicit conversion of integers to strings
```

```
e = 10
```

```
f = str(e) # f will be '10' (a string)
```

# Explicit conversion of lists to tuples

```
g = [1, 2, 3]
```

```
h = tuple(g) # h will be (1, 2, 3) (a tuple)
```

It's important to note that explicit type conversion can lead to loss of data if the target data type is not compatible with the original value. For example, if you try to convert a float to an integer and there is a decimal part, the decimal part will be truncated.

## **Boolean operators**

Boolean operators are used to perform logical operations on boolean operands (values of True or False). In Python, there are three boolean operators: and, or, and not.

Here's how you can use boolean operators in Python:

### **and operator**

The and operator returns True if both operands are True, and False otherwise.

Code

```
x = True
```

```
y = False
```

```
z = x and y # z will be False
```

## **or operator**

The or operator returns True if at least one of the operands is True, and False otherwise.

Code

```
x = True
```

```
y = False
```

```
z = x or y # z will be True
```

[OceanofPDF.com](http://OceanofPDF.com)



## **not operator**

The not operator returns True if the operand is False, and False if the operand is True.

Code

```
x = True
```

```
y = False
```

```
z = not x # z will be False
```

```
w = not y # w will be True
```

You can use boolean operators in combination with comparison operators to create more complex expressions. For example:

Code

```
x = 10
```

```
y = 5
```

```
if x > 5 and y < 10: # x is greater than 5 and y is less than 10, so this condition will be True
```

```
    print('x is greater than 5 and y is less than 10')
```

```
if x > 5 or y < 10: # x is greater than 5, so this condition will be True
```

```
    print('x is greater than 5 or y is less than 10')
```

```
if not x > 5: # x is not greater than 5, so this condition will be False
    print('x is not greater than 5')
```

[OceanofPDF.com](http://OceanofPDF.com)

## Control Flow Statements in Python

In Python, control flow statements are used to control the flow of execution of a program based on certain conditions. There are several types of control flow statements in Python, including:

### if statement

The if statement is used to execute a block of code if a condition is True. You can also use the elif clause to test multiple conditions, and the else clause to specify a block of code to be executed if none of the conditions are True.

Here's an example of how you can use the if statement in Python:

Code

```
x = 10
```

```
if x > 5: # x is greater than 5, so this condition is True
```

```
    print('x is greater than 5')
```

```
elif x < 5: # x is not less than 5, so this condition is False
```

```
    print('x is less than 5')
```

```
else: # None of the conditions are True, so this block is executed
```

```
    print('x is equal to 5')
```

**for loop**

The for loop is used to iterate over a sequence (such as a list, tuple, or string) and execute a block of code for each element in the sequence.

Here's an example of how you can use the for loop in Python:

Code

```
fruits = ['Edcorner', 'Learning', 'cherry']
```

```
for fruit in fruits: # Iterate over the elements of the 'fruits' list
```

```
    print(fruit) # Print the current element
```

[OceanofPDF.com](http://OceanofPDF.com)

### **while loop**

The while loop is used to execute a block of code repeatedly as long as a condition is True.

Here's an example of how you can use the while loop in Python:

Code

```
x = 10
```

```
while x > 0: # x is greater than 0, so this condition is True
```

```
    print(x) # Print x
```

```
    x -= 1 # Decrement x by 1
```

### **if...else statement in Python:**

Code

```
# Example 1: Testing multiple conditions
```

```
x = 10
```

```
y = 5
```

```
if x > y:
```

```
    print('x is greater than y')
```

```
elif x < y:
```

```
    print('x is less than y')
```

else:

```
print('x is equal to y')
```

# Output: 'x is greater than y'

[OceanofPDF.com](http://OceanofPDF.com)

# Example 2: Using nested if...else statements

x = 10

y = 5

if x > y:

    if x > 0:

        print('x is positive and greater than y')

    else:

        print('x is negative and greater than y')

else:

    if x > 0:

        print('x is positive and less than or equal to y')

    else:

        print('x is negative and less than or equal to y')

# Output: 'x is positive and greater than y'

# Example 3: Using the ternary operator (a shorthand for if...else statements)

x = 10

y = 5

```
result = 'x is greater than y' if x > y else 'x is less than or equal to y'  
print(result)
```

# Output: 'x is greater than y'

[OceanofPDF.com](http://OceanofPDF.com)



## while loop in Python:

Code

# Example 1: Simple loop

```
x = 10
```

```
while x > 0: # x is greater than 0, so this condition is True
```

```
    print(x) # Print x
```

```
    x -= 1 # Decrement x by 1
```

# Output: 10 9 8 7 6 5 4 3 2 1

# Example 2: Using the break statement to exit the loop

```
x = 10
```

```
while True: # This loop will run indefinitely
```

```
    print(x) # Print x
```

```
    x -= 1 # Decrement x by 1
```

```
    if x == 0: # If x is equal to 0, exit the loop
```

```
        break
```

# Output: 10 9 8 7 6 5 4 3 2 1

[OceanofPDF.com](http://OceanofPDF.com)

# Example 3: Using the continue statement to skip the rest of the current iteration

```
x = 10
```

```
while x > 0: # x is greater than 0, so this condition is True
```

```
    x -= 1 # Decrement x by 1
```

```
    if x % 2 == 1: # If x is odd, skip the rest of the current iteration
```

```
        continue
```

```
    print(x) # Print x
```

# Output: 8 6 4 2 0

# Example 4: Using the else clause with a while loop

```
x = 10
```

```
while x > 0: # x is greater than 0, so this condition is True
```

```
    x -= 1 # Decrement x by 1
```

```
else: # This block is executed when the loop finishes normally (i.e., the  
condition becomes False)
```

```
    print("The loop has finished")
```

# Output: The loop has finished

[OceanofPDF.com](http://OceanofPDF.com)

## for loop in Python:

Code

# Example 1: Iterating over a list of strings

```
fruits = ['edcorner', 'learning', 'cherry']
```

```
for fruit in fruits: # Iterate over the elements of the 'fruits' list
```

```
    print(fruit) # Print the current element
```

# Output: 'edcorner' 'learning' 'cherry'

# Example 2: Iterating over a list of integers and using the enumerate() function

```
numbers = [1, 2, 3, 4, 5]
```

```
for i, number in enumerate(numbers): # Iterate over the elements and their indices of the 'numbers' list
```

```
    print(f'Element {i}: {number}') # Print the current element and its index
```

# Output: 'Element 0: 1' 'Element 1: 2' 'Element 2: 3' 'Element 3: 4' 'Element 4: 5'

# Example 3: Iterating over a string

```
string = 'hello'
```

```
for char in string: # Iterate over the characters of the 'string'
```

```
    print(char) # Print the current character
```

```
# Output: 'h' 'e' 'l' 'l' 'o'
```

[OceanofPDF.com](http://OceanofPDF.com)

# Example 4: Iterating over a dictionary

```
dictionary = {'a': 1, 'b': 2, 'c': 3}
```

```
for key, value in dictionary.items(): # Iterate over the key-value pairs of the  
'dictionary'
```

```
    print(f'Key: {key}, Value: {value}') # Print the current key and value
```

# Output: 'Key: a, Value: 1' 'Key: b, Value: 2' 'Key: c, Value: 3'

[OceanofPDF.com](http://OceanofPDF.com)

## For Else and While Else in Python

In Python, you can use the else clause with both for and while loops. The else clause is executed after the loop finishes normally (i.e., the loop condition becomes False), but is not executed if the loop is exited prematurely by a break statement.

Here are some examples of how you can use the else clause with for and while loops in Python:

Code

# Example 1: Using the else clause with a for loop

```
numbers = [1, 2, 3, 4, 5]
```

```
for number in numbers: # Iterate over the elements of the 'numbers' list
```

```
    if number % 2 == 0: # If the current element is even, skip the rest of the
        current iteration
```

```
        continue
```

```
    print(number)
```

```
else: # This block is executed when the loop finishes normally (i.e., the
    condition becomes False)
```

```
    print('The loop has finished')
```

```
# Output: 1 3 5 The loop has finished
```



# Example 2: Using the else clause with a while loop

```
x = 10
```

```
while x > 0: # x is greater than 0, so this condition is True
```

```
    x -= 1 # Decrement x by 1
```

```
else: # This block is executed when the loop finishes normally (i.e., the  
condition becomes False)
```

```
    print("The loop has finished")
```

# Output: The loop has finished

# Example 3: Exiting the loop prematurely with the break statement

```
numbers = [1, 2, 3, 4, 5]
```

```
for number in numbers: # Iterate over the elements of the 'numbers' list
```

```
    if number == 3: # If the current element is 3, exit the loop
```

```
        break
```

```
    print(number)
```

```
else: # This block is not executed because the loop was exited prematurely
```

```
    print("The loop has finished")
```

# Output: 1 2

[OceanofPDF.com](http://OceanofPDF.com)

## Break, Pass and Continue Statement in Python

In Python, the break, pass, and continue statements are used to control the flow of execution of a loop.

### **break statement**

The break statement is used to exit a loop prematurely. When a break statement is encountered inside a loop, the loop is terminated immediately and the program control is transferred to the statement following the loop.

Here's an example of how you can use the break statement in a for loop in Python:

Code

```
numbers = [1, 2, 3, 4, 5]

for number in numbers: # Iterate over the elements of the 'numbers' list
    if number == 3: # If the current element is 3, exit the loop
        break
    print(number)
```

# Output: 1 2

You can also use the break statement with a while loop:

Code

```
x = 10
while True: # This loop will run indefinitely
    x -= 1 # Decrement x by 1
    if x == 0: # If x is equal to 0, exit the loop
        break
    print(x)
# Output: 9 8 7 6 5 4 3 2 1
```

## pass statement

The pass statement is a null operation in Python, which means that it does nothing. It is used as a placeholder in cases where a statement is required syntactically, but you don't want to execute any code.

Here's an example of how you can use the pass statement in a for loop in Python:

Code

```
numbers = [1, 2, 3, 4, 5]

for number in numbers: # Iterate over the elements of the 'numbers' list
    if number % 2 == 0: # If the current element is even, skip the rest of the
        current iteration
        pass # Do nothing
    else:
```

```
print(number)
```

# Output: 1 3 5

You can also use the pass statement with a while loop:

Code

```
x = 10
```

```
while x > 0: # x is greater than 0, so this condition is True
    x -= 1 # Decrement x by 1
    if x % 2 == 1: # If x is odd, skip the rest of the current iteration
        pass # Do nothing
    else:
        print(x)
```

# Output: 8 6 4 2 0

## Continue Statement in Python

The continue statement is used to skip the rest of the current iteration of a loop and move on to the next iteration. When a continue statement is encountered inside a loop, the program control goes back to the loop condition and the next iteration begins.

Here's an example of how you can use the continue statement in a for loop in Python:

Code

```
numbers = [1, 2, 3, 4, 5]
```

```
for number in numbers: # Iterate over the elements of the 'numbers' list
    if number % 2 == 0: # If the current element is even, skip the rest of the
        current iteration
        continue
    print(number)
```

# Output: 1 3 5

You can also use the continue statement with a while loop:

Code

```
x = 10
```

```
while x > 0: # x is greater than 0, so this condition is True
    x -= 1 # Decrement x by 1
    if x % 2 == 1: # If x is odd, skip the rest of the current iteration
        continue
    print(x)
```

# Output: 8 6 4 2 0

# Strings in Python

In Python, a string is a sequence of characters. Strings are immutable, which means that once you create a string, you cannot change its contents. However, you can create a new string based on an existing string by concatenating, slicing, or modifying the original string.

Here are some examples of how you can work with strings in Python:

Code

# Declaring a string

```
string = 'hello'
```

# Concatenating strings

```
string1 = 'hello'
```

```
string2 = 'edcorner'
```

```
string3 = string1 + ' ' + string2 # Concatenate the strings using the + operator
```

```
print(string3) # Output: 'hello edcorner'
```

# Slicing a string

```
string = 'hello'
```

```
print(string[1:4]) # Output: 'ell' (the slice starts at index 1 and ends at index 4, not including the character at index 4)
```

```
# Modifying a string
string = 'hello'
string = string.upper() # Convert the string to uppercase
print(string) # Output: 'HELLO'
string = string.replace('L', 'l') # Replace the character 'L' with 'l'
print(string) # Output: 'HELLO'
```

[OceanofPDF.com](http://OceanofPDF.com)



```
# Iterating over a string
```

```
string = 'hello'
```

```
for char in string: # Iterate over the characters of the 'string'
```

```
    print(char) # Print the current character
```

```
# Output: 'h' 'e' 'l' 'l' 'o'
```

[OceanofPDF.com](http://OceanofPDF.com)

## String Slicing in Python

In Python, you can use string slicing to extract a sub-string from a string. String slicing involves specifying a range of indices to extract the corresponding characters from a string.

Here's an example of how you can use string slicing in Python:

Code

```
string = 'hello'
```

```
# Extract the characters from index 1 to index 4 (not including the character at index 4)
```

```
substring = string[1:4]
```

```
print(substring) # Output: 'ell'
```

```
# Extract the characters from index 0 to index 3 (not including the character at index 3)
```

```
substring = string[:3]
```

```
print(substring) # Output: 'hel'
```

```
# Extract the characters from index 3 to the end of the string
```

```
substring = string[3:]
```

```
print(substring) # Output: 'lo'
```

```
# Extract all the characters in the string
```

```
substring = string[:]
```

```
print(substring) # Output: 'hello'
```

[OceanofPDF.com](http://OceanofPDF.com)

```
# Extract the characters in the string in reverse order
```

```
substring = string[::-1]
```

```
print(substring) # Output: 'olleh'
```

In the examples above, the syntax for string slicing is `string[start:end:step]`, where `start` is the index of the first character to include in the substring (default is 0), `end` is the index of the first character to exclude from the substring (default is the length of the string), and `step` is the number of indices to skip between characters (default is 1).

[OceanofPDF.com](http://OceanofPDF.com)

## String Methods Python

In Python, you can use string methods to perform various operations on strings. String methods are functions that are associated with string objects and can be called using the `.` operator.

Here are some examples of common string methods in Python:

Code

```
string = 'hello'
```

```
# Convert the string to uppercase
```

```
string = string.upper()
```

```
print(string) # Output: 'HELLO'
```

```
# Convert the string to lowercase
```

```
string = string.lower()
```

```
print(string) # Output: 'hello'
```

```
# Replace a character in the string
```

```
string = string.replace('l', 'L')
```

```
print(string) # Output: 'heLLo'
```

```
# Split the string into a list of substrings based on a delimiter
```

```
string = 'edcorner,learning,cherry'
```

```
substrings = string.split(',')  
substrings
```

```
print(substrings) # Output: ['edcorner', 'learning', 'cherry']
```

```
# Strip leading and trailing whitespace from the string
```

```
string = ' hello '
```

```
string = string.strip()
```

```
print(string) # Output: 'hello'
```

[OceanofPDF.com](http://OceanofPDF.com)

# Check if the string starts with a specific substring

```
string = 'hello'
```

```
print(string.startswith('he')) # Output: True
```

```
print(string.startswith('ho')) # Output: False
```

# Check if the string ends with a specific substring

```
string = 'hello'
```

```
print(string.endswith('lo')) # Output: True
```

```
print(string.endswith('lo')) # Output: False
```

# Check if the string contains a specific substring

```
string = 'hello'
```

```
print('he' in string) # Output: True
```

```
print('ho' in string) # Output: False
```

These are just a few examples of string methods in Python. There are many more methods available, such as `find()`, `rfind()`, `index()`, `rindex()`, `join()`, `format()`, etc. You can find a complete list of string methods in the Python documentation.

## String Formatting in Python

In Python, you can use string formatting to insert the value of variables into a string. String formatting is a way to create a new string based on a template string and a set of values.

There are several ways to format strings in Python, including:

Using the % operator

Using the .format() method

Using f-strings (available in Python 3.6 and later)

Here are some examples of string formatting in Python:

Code

```
# Using the % operator
```

```
name = 'Edcorner'
```

```
age = 30
```

```
string = 'My name is %s and I am %d years old' % (name, age)
```

```
print(string) # Output: 'My name is Edcorner and I am 30 years old'
```

```
# Using the .format() method
```

```
name = 'Edcorner'
```

```
age = 30
```

```
string = 'My name is {} and I am {} years old'.format(name, age)
```

```
print(string) # Output: 'My name is Edcorner and I am 30 years old'
```



```
# Using f-strings (available in Python 3.6 and later)
```

```
name = 'Edcorner'
```

```
age = 30
```

[OceanofPDF.com](https://oceanofpdf.com)

```
string = f'My name is {name} and I am {age} years old'
```

```
print(string) # Output: 'My name is Edcorner and I am 30 years old'
```

In the examples above, the placeholders for the values are represented by %s (for strings) and %d (for integers) in the first method, and by {} in the second and third methods. The values are passed as arguments to the formatting operator or method, and are inserted into the placeholders in the resulting string

[OceanofPDF.com](http://OceanofPDF.com)

## Functions in Python

In Python, a function is a block of code that performs a specific task and can be called from other parts of a program. Functions help you to organize your code and make it reusable.

Here's an example of how you can define and call a function in Python:

Code

```
def greet(name): # Define the function 'greet' with one parameter 'name'  
    print(f'Hello, {name}!') # Print a greeting message
```

```
greet('Edcorner') # Call the function with the argument 'Edcorner'
```

```
# Output: 'Hello, Edcorner!'
```

```
greet('Jane') # Call the function with the argument 'Jane'
```

```
# Output: 'Hello, Jane!'
```

You can also define functions with default values for some or all of their parameters. In this case, you can omit the corresponding arguments when calling the function, and the default values will be used.

Code

```
def greet(name, greeting='Hello'): # Define the function 'greet' with two  
parameters 'name' and 'greeting' (greeting has a default value of 'Hello')  
    print(f'{greeting}, {name}!') # Print a greeting message
```

```
greet('Edcorner') # Call the function with the argument 'Edcorner' (the  
default value of 'greeting' will be used)
```

```
# Output: 'Hello, Edcorner!'
```

```
greet('Jane', greeting='Hi') # Call the function with the arguments 'Jane' and 'Hi'
```

```
# Output: 'Hi, Jane!'
```

You can also define functions that return a value using the return statement.

[OceanofPDF.com](http://OceanofPDF.com)

## Code

```
def add(x, y): # Define the function 'add' with two parameters 'x' and 'y'  
    result = x + y # Calculate the sum of 'x' and 'y'  
    return result # Return the result
```

```
sum = add(3, 4) # Call the function with the arguments 3 and 4, and assign  
the returned value to the variable 'sum'
```

```
print(sum) # Output: 7
```

[OceanofPDF.com](http://OceanofPDF.com)

## Scope of Variable in Python

In Python, the scope of a variable refers to the part of the program where the variable is accessible. There are two types of scopes in Python: global scope and local scope.

A global variable is a variable that is defined outside of any function and is accessible from any part of the program. A local variable is a variable that is defined inside a function and is only accessible within the function.

Here's an example of how global and local variables work in Python:

Code

```
x = 10 # Define a global variable 'x' with the value 10
```

```
def func():
```

```
    y = 20 # Define a local variable 'y' with the value 20
```

```
    print(x) # Access the global variable 'x'
```

```
    print(y) # Access the local variable 'y'
```

```
print(x) # Access the global variable 'x'
```

```
# Output: 10
```

```
try:
```

```
    print(y) # Access the local variable 'y' (this will raise an error because 'y'
is not defined in the global scope)
```

```
except NameError as e:
```

```
print(e) # Output: "name 'y' is not defined"
```

```
func() # Call the function 'func'
```

```
# Output: 10 20
```

[OceanofPDF.com](http://OceanofPDF.com)

In the example above, the global variable `x` is accessible from both the global scope and the local scope, while the local variable `y` is only accessible from within the function `func`.

It's important to note that you cannot modify a global variable from within a function without explicitly declaring it as a global variable using the `global` keyword.

Code

```
x = 10 # Define a global variable 'x' with the value 10
```

```
def func():
```

```
    global x # Declare the global variable 'x'
```

```
    x = 20 # Modify the value of 'x'
```

```
    print(x) # Access the global variable 'x'
```

```
print(x) # Access the global variable 'x'
```

```
# Output: 10
```

```
func() # Call the function 'func'
```

```
print(x) # Access the global variable 'x'
```

```
# Output: 20
```

## Types of Function Arguments in Python



In Python, you can define functions with different types of arguments. Here are some common types of function arguments in Python:

**Positional arguments:** Positional arguments are arguments that are passed to a function in a specific order. The function receives the arguments in the same order in which they are passed.

Code

```
def add(x, y): # Define the function 'add' with two positional arguments 'x' and 'y'
```

```
    result = x + y # Calculate the sum of 'x' and 'y'
```

```
    return result # Return the result
```

```
sum = add(3, 4) # Call the function with the arguments 3 and 4, and assign the returned value to the variable 'sum'
```

```
print(sum) # Output: 7
```

[OceanofPDF.com](http://OceanofPDF.com)

**Keyword arguments:** Keyword arguments are arguments that are passed to a function using their names (also known as keys). The order in which the arguments are passed does not matter, as long as you use the correct names.

Code

```
def greet(name, greeting='Hello'): # Define the function 'greet' with two  
keyword arguments 'name' and 'greeting' (greeting has a default value of  
'Hello')
```

```
    print(f'{greeting}, {name}!') # Print a greeting message
```

```
greet(name='Edcorner') # Call the function with the argument 'Edcorner'  
(the default value of 'greeting' will be used)
```

```
# Output: 'Hello, Edcorner!'
```

```
greet(greeting='Hi', name='Jane') # Call the function with the arguments  
'Jane' and 'Hi' (the order of the arguments does not matter)
```

```
# Output: 'Hi, Jane!'
```

[OceanofPDF.com](http://OceanofPDF.com)

**Variable-length arguments:** Variable-length arguments are arguments that allow you to pass a variable number of arguments to a function. There are two types of variable-length arguments: `*args` and `**kwargs`.

Here's an example of how you can use `*args` to define and call a function with a variable number of positional arguments:

Code

```
def add(*numbers): # Define the function 'add' with a variable-length
argument 'numbers' (using the '*' operator)

    result = 0

    for number in numbers: # Iterate over the elements of the 'numbers' tuple
        result += number # Add the current number to the result

    return result # Return the result

sum = add(1, 2, 3, 4, 5) # Call the function with five arguments
print(sum) # Output: 15
```

```
sum = add(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) # Call the function with ten
arguments
print(sum) # Output: 55
```

And here's an example of how you can use `**kwargs` to define and call a function with a variable number of keyword arguments:

Code

```
def greet(**kwargs): # Define the function 'greet' with a variable-length
argument 'kwargs' (using the '**' operator)
```

```
for key, value in kwargs.items(): # Iterate over the elements of the  
'kwargs' dictionary
```

```
    print(f'{key}: {value}') # Print the key and value of the current element
```

```
greet(name='Edcorner', greeting='Hello') # Call the function with two  
keyword arguments
```

```
# Output: 'name: Edcorner' 'greeting: Hello'
```

```
greet(name='Jane', greeting='Hi', age=30) # Call the function with three  
keyword arguments
```

```
# Output: 'name: Jane' 'greeting: Hi' 'age: 30'
```

## **\*Args and \*\*Kwargs in Python**

In Python, `*args` and `**kwargs` are special syntaxes that allow you to pass a variable number of arguments to a function.

`*args` is a tuple that contains the positional arguments passed to a function. You can use `*args` to define a function that accepts a variable number of positional arguments.

Code

```
def add(*numbers): # Define the function 'add' with a variable-length  
argument 'numbers' (using the '*' operator)
```

```
    result = 0
```

```
    for number in numbers: # Iterate over the elements of the 'numbers' tuple
```

```
        result += number # Add the current number to the result
```

```
    return result # Return the result
```

```
sum = add(1, 2, 3, 4, 5) # Call the function with five arguments  
print(sum) # Output: 15
```

```
sum = add(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) # Call the function with ten  
arguments  
print(sum) # Output: 55
```

**\*\*kwargs is a dictionary that contains the keyword arguments passed to a function. You can use \*\*kwargs to define a function that accepts a variable number of keyword arguments.**

[OceanofPDF.com](http://OceanofPDF.com)

## Code

```
def greet(**kwargs): # Define the function 'greet' with a variable-length
argument 'kwargs' (using the '**' operator)

    for key, value in kwargs.items(): # Iterate over the elements of the
'kwargs' dictionary

        print(f'{key}: {value}') # Print the key and value of the current element
```

```
greet(name='Edcorner', greeting='Hello') # Call the function with two
keyword arguments
```

```
# Output: 'name: Edcorner' 'greeting: Hello'
```

```
greet(name='Jane', greeting='Hi', age=30) # Call the function with three
keyword arguments
```

```
# Output: 'name: Jane' 'greeting: Hi' 'age: 30'
```

You can also use `*args` and `**kwargs` together in the same function definition. In this case, the positional arguments are passed to `*args`, and the keyword arguments are passed to `**kwargs`.

## Code

```
def func(*args, **kwargs): # Define the function 'func' with a variable-
length argument 'args' and a variable-length argument 'kwargs'

    print(args) # Print the 'args' tuple

    print(kwargs) # Print the 'kwargs' dictionary
```

```
func(1, 2, 3, name='Edcorner', greeting='Hello') # Call the function with  
three positional arguments and two keyword arguments
```

```
# Output: (1, 2, 3) {'name': 'Edcorner', 'greeting': 'Hello'}
```

[OceanofPDF.com](http://OceanofPDF.com)

# Packing and Unpacking Arguments in Python

In Python, you can use the \* and \*\* operators to "pack" and "unpack" arguments.

Packing arguments refers to the process of creating a tuple or a dictionary from a list of arguments. You can use the \* operator to pack a list of arguments into a tuple, and the \*\* operator to pack a list of arguments into a dictionary.

Here's an example of how you can use the \* operator to pack a list of arguments into a tuple:

Code

```
def func(x, y, z): # Define the function 'func' with three positional arguments

    print(x, y, z) # Print the three arguments
```

```
numbers = [1, 2, 3] # Define a list of numbers
```

```
func(*numbers) # Call the function with the packed tuple (1, 2, 3)
```

```
# Output: 1 2 3
```

And here's an example of how you can use the \*\* operator to pack a list of arguments into a dictionary:

Code

```
def func(**kwargs): # Define the function 'func' with a variable-length argument 'kwargs' (using the '**' operator)

    print(kwargs) # Print the 'kwargs' dictionary
```



```
data = {'x': 1, 'y': 2, 'z': 3} # Define a dictionary of data
```

```
func(**data) # Call the function with the packed dictionary {'x': 1, 'y': 2, 'z': 3}
```

```
# Output: {'x': 1, 'y': 2, 'z': 3}
```

Unpacking arguments refers to the process of extracting the values of a tuple or a dictionary into a list of arguments. You can use the \* operator to unpack a tuple into a list of arguments, and the \*\* operator to unpack a dictionary into a list of keyword arguments.

[OceanofPDF.com](http://OceanofPDF.com)

Here's an example of how you can use the \* operator to unpack a tuple into a list of arguments:

Code

```
def func(x, y, z): # Define the function 'func' with three positional arguments
```

```
    print(x, y, z) # Print the three arguments
```

```
numbers = (1, 2, 3) # Define a tuple of numbers
```

```
func(*numbers) # Call the function with the unpacked tuple (1, 2, 3)
```

```
# Output: 1 2 3
```

And here's an example of how you can use the \*\* operator to unpack a dictionary into a list of keyword arguments:

Code

```
def func(**kwargs): # Define the function 'func' with a variable-length argument 'kwargs' (using the '**' operator)
```

```
    print(kwargs) # Print the 'kwargs' dictionary
```

```
data = {'x': 1, 'y': 2, 'z': 3} # Define a dictionary of data
```

```
func(**data) # Call the function with the unpacked dictionary {'x': 1, 'y': 2, 'z': 3}
```

```
# Output: {'x': 1, 'y': 2, 'z': 3}
```

[OceanofPDF.com](http://OceanofPDF.com)

# Lambda and Anonymous Function in Python

In Python, you can use lambda expressions to create anonymous functions.

An anonymous function is a function that is defined without a name.

Anonymous functions are often used as throwaway functions, that are used only once and then discarded.

Here's an example of how you can use a lambda expression to define a simple anonymous function that calculates the sum of two numbers:

Code

```
add = lambda x, y: x + y # Define the anonymous function 'add' with two arguments 'x' and 'y' (using the 'lambda' keyword)
```

```
result = add(3, 4) # Call the anonymous function with the arguments 3 and 4
```

```
print(result) # Output: 7
```

You can also use anonymous functions inside other functions, for example as a callback function:

Code

```
def apply_operation(x, y, operation): # Define the function 'apply_operation' with three arguments: 'x', 'y', and 'operation'
```

```
    result = operation(x, y) # Call the 'operation' function with 'x' and 'y' as arguments
```

```
    return result # Return the result
```

```
result = apply_operation(3, 4, lambda x, y: x + y) # Call the function  
'apply_operation' with the anonymous function 'lambda x, y: x + y' as the  
'operation' argument
```

```
print(result) # Output: 7
```

Keep in mind that anonymous functions are limited to a single expression.  
If you need to define a more complex function, you should use a regular  
function definition instead.

[OceanofPDF.com](http://OceanofPDF.com)

# Recursion in Python

In Python, recursion is the process of defining a function in terms of itself. A function that calls itself is called a recursive function.

Recursion can be a useful technique for solving problems that can be broken down into smaller subproblems. For example, you can use recursion to implement a factorial function, which calculates the factorial of a given number:

Code

```
def factorial(n): # Define the function 'factorial' with one argument 'n'
    if n == 0: # If 'n' is 0
        return 1 # Return 1 (the factorial of 0 is 1)
    else: # Otherwise
        return n * factorial(n - 1) # Return 'n' multiplied by the factorial of 'n - 1'
        (recursive call)
```

```
result = factorial(5) # Call the function with the argument 5
print(result) # Output: 120 (5 * 4 * 3 * 2 * 1)
```

It's important to note that recursive functions can be computationally expensive, and they may consume a lot of memory if the recursion is too deep. You should be careful when using recursion, and make sure to include a base case (like the one shown in the example above) to avoid infinite recursion.

## Recursion in Python Simple Example`

Here's a simple example of how you can use recursion in Python to calculate the sum of the elements of a list:

Code

```
def sum_list(lst): # Define the function 'sum_list' with one argument 'lst'
    if not lst: # If the list is empty
        return 0 # Return 0 (the sum of an empty list is 0)
    else: # Otherwise
        return lst[0] + sum_list(lst[1:]) # Return the first element of the list plus
        the sum of the rest of the list (recursive call)

result = sum_list([1, 2, 3, 4, 5]) # Call the function with the argument [1, 2,
3, 4, 5]

print(result) # Output: 15 (1 + 2 + 3 + 4 + 5)
```

In this example, the function `sum_list` calculates the sum of the elements of the list by recursively calling itself with the rest of the list (`lst[1:]`). The base case is when the list is empty, in which case the function returns 0.

[OceanofPDF.com](http://OceanofPDF.com)

## list() in Python

In Python, the list() function is a built-in function that creates a new list from an iterable object.

Here's an example of how you can use the list() function to create a list from a string:

Code

```
string = 'hello' # Define a string
lst = list(string) # Create a list from the string
print(lst) # Output: ['h', 'e', 'l', 'l', 'o']
```

You can also use the list() function to create a list from a tuple:

Code

```
tuple = (1, 2, 3, 4, 5) # Define a tuple
lst = list(tuple) # Create a list from the tuple
print(lst) # Output: [1, 2, 3, 4, 5]
```

Keep in mind that the list() function creates a new list object, which is distinct from the original iterable object. Modifying the new list will not affect the original object.



**Here's another example of how you can use the list() function in Python:**

Code

```
def get_elements(iterable): # Define the function 'get_elements' with one  
argument 'iterable'
```

```
    elements = list(iterable) # Create a list from the 'iterable' object
```

```
    return elements # Return the list
```

```
# Call the function with a range object as the argument
```

```
result = get_elements(range(10))
```

```
print(result) # Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In this example, the `get_elements` function takes an iterable object as an argument and returns a list with the elements of the object. The `range()` function returns a range object that generates a sequence of numbers, which can be passed as an argument to the `get_elements` function.

## Dictionary in Python

In Python, a dictionary is a collection of key-value pairs. Dictionaries are also known as associative arrays or hash maps.

You can create a dictionary in Python by enclosing a comma-separated list of key-value pairs in curly braces ({}), or by using the dict() function.

Here's an example of how you can create a dictionary in Python:

Code

```
# Method 1: Using curly braces
```

```
dict1 = {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

```
print(dict1) # Output: {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

```
# Method 2: Using the dict() function
```

```
dict2 = dict(key1='value1', key2='value2', key3='value3')
```

```
print(dict2) # Output: {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

You can access the values of a dictionary using their keys, using square brackets ([]):

Code

```
dict1 = {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

```
value1 = dict1['key1'] # Get the value of 'key1'
```

```
print(value1) # Output: 'value1'
```

```
value2 = dict1['key2'] # Get the value of 'key2'
```

```
print(value2) # Output: 'value2'
```

You can also use the `get()` method to access the values of a dictionary, which returns a default value if the key is not found:

[OceanofPDF.com](http://OceanofPDF.com)

## Code

```
dict1 = {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

```
value1 = dict1.get('key1', 'default') # Get the value of 'key1', or 'default' if it  
is not found
```

```
print(value1) # Output: 'value1'
```

```
value2 = dict1.get('key4', 'default') # Get the value of 'key4', or 'default' if it  
is not found
```

```
print(value2) # Output: 'default'
```

## Sets in Python

In Python, a set is an unordered collection of unique elements. Sets are often used to eliminate duplicate values and to perform mathematical set operations, such as union, intersection, difference, and symmetric difference.

You can create a set in Python by enclosing a comma-separated list of elements in curly braces (`{}`), or by using the `set()` function.

Here's an example of how you can create a set in Python:

Code

```
# Method 1: Using curly braces
```

```
set1 = {1, 2, 3, 4, 5}
```

```
print(set1) # Output: {1, 2, 3, 4, 5}
```

```
# Method 2: Using the set() function
```

```
set2 = set([1, 2, 3, 4, 5])
```

```
print(set2) # Output: {1, 2, 3, 4, 5}
```

You can add elements to a set using the `add()` method, and remove elements using the `remove()` method:

Code

```
set1 = {1, 2, 3, 4, 5}
```

```
set1.add(6) # Add the element 6 to the set
```

```
print(set1) # Output: {1, 2, 3, 4, 5, 6}
```

```
set1.remove(4) # Remove the element 4 from the set
```

```
print(set1) # Output: {1, 2, 3, 5, 6}
```

You can also perform set operations using the built-in set methods, such as `union()`, `intersection()`, `difference()`, and `symmetric_difference()`:

[OceanofPDF.com](http://OceanofPDF.com)

Code

```
set1 = {1, 2, 3, 4, 5}
```

```
set2 = {3, 4, 5, 6, 7}
```

```
union = set1.union(set2) # Get the union of set1 and set2
```

```
print(union) # Output: {1, 2, 3, 4, 5, 6, 7}
```

```
intersection = set1.intersection(set2) # Get the intersection of set1 and set2
```

```
print(intersection) # Output: {3, 4, 5}
```

```
difference = set1.difference(set2) # Get the difference of set1 and set2
```

```
print(difference) # Output: {1, 2}
```

```
symmetric_difference = set1.symmetric_difference(set2) # Get the  
symmetric difference of set1 and set2
```

```
print(symmetric_difference) # Output: {1, 2, 6, 7}
```

[OceanofPDF.com](http://OceanofPDF.com)

# Numbers in Python

In Python, numbers are used to store numeric values. Python supports various numeric types, including integers, floating-point numbers, and complex numbers.

Here are some examples of how you can use numbers in Python:

Code

# Integers

```
x = 10 # Assign the value 10 to the variable 'x'
```

```
y = -5 # Assign the value -5 to the variable 'y'
```

```
print(x + y) # Output: 5 (10 + (-5) = 5)
```

# Floating-point numbers

```
a = 3.14 # Assign the value 3.14 to the variable 'a'
```

```
b = 2.71 # Assign the value 2.71 to the variable 'b'
```

```
print(a * b) # Output: 8.5094 (3.14 * 2.71 = 8.5094)
```

# Complex numbers

```
c = 1 + 2j # Assign the value 1 + 2j to the variable 'c'
```

```
d = 3 + 4j # Assign the value 3 + 4j to the variable 'd'
```

```
print(c + d) # Output: (4+6j) ((1+2j) + (3+4j) = (4+6j))
```

You can also use the built-in functions `int()`, `float()`, and `complex()` to convert numbers from one type to another:



Code

```
# Convert an integer to a floating-point number
```

```
x = 10
```

```
y = float(x)
```

```
print(y) # Output: 10.0
```

[OceanofPDF.com](http://OceanofPDF.com)

# Convert a floating-point number to an integer

a = 3.14

b = int(a)

print(b) # Output: 3

# Convert a complex number to a floating-point number

c = 1 + 2j

d = float(c)

print(d) # Output: TypeError: can't convert complex to float

[OceanofPDF.com](http://OceanofPDF.com)

## Tuples in Python

In Python, a tuple is an immutable sequence type, similar to a list. Tuples are often used to store a fixed set of values that are not meant to be modified.

You can create a tuple in Python by enclosing a comma-separated list of values in parentheses (), or by using the tuple() function.

Here's an example of how you can create a tuple in Python:

Code

```
# Method 1: Using parentheses
```

```
tuple1 = (1, 2, 3, 4, 5)
```

```
print(tuple1) # Output: (1, 2, 3, 4, 5)
```

```
# Method 2: Using the tuple() function
```

```
tuple2 = tuple([1, 2, 3, 4, 5])
```

```
print(tuple2) # Output: (1, 2, 3, 4, 5)
```

You can access the elements of a tuple using indexing, similar to a list:

Code

```
tuple1 = (1, 2, 3, 4, 5)
```

```
first = tuple1[0] # Get the first element of the tuple
```

```
print(first) # Output: 1
```

```
second = tuple1[1] # Get the second element of the tuple
```

```
print(second) # Output: 2
```

```
last = tuple1[-1] # Get the last element of the tuple
```

```
print(last) # Output: 5
```

Keep in mind that tuples are immutable, which means you cannot modify their values. You cannot add or remove elements from a tuple, and you cannot assign a new value to an element of a tuple.

[OceanofPDF.com](http://OceanofPDF.com)

## Exception Handling in Python

In Python, exception handling is a mechanism to handle runtime errors and unexpected conditions that may occur during the execution of a program.

When an error or exception occurs in a program, Python raises an exception, which is a special object that contains information about the error. You can use exception handling to catch and handle these exceptions, so that your program can continue to run even if an error occurs.

Here's an example of how you can use exception handling in Python:

Code

```
# Define a function that divides two numbers
```

```
def divide(x, y):
```

```
    try:
```

```
        result = x / y # Divide the two numbers
```

```
    except ZeroDivisionError: # If a ZeroDivisionError occurs (division by zero)
```

```
        print("Cannot divide by zero!") # Print an error message
```

```
    else: # If no exceptions occurred
```

```
        print(result) # Print the result
```

```
    finally: # Always execute this block, whether an exception occurred or not
```

```
        print("Execution complete.") # Print a message
```

# Call the function with valid arguments

divide(10, 5) # Output: 2.0

# Output: Execution complete.

[OceanofPDF.com](http://OceanofPDF.com)

```
# Call the function with an invalid argument (division by zero)
```

```
divide(10, 0)
```

```
# Output: Cannot divide by zero!
```

```
# Output: Execution complete.
```

In this example, the `divide()` function divides two numbers and prints the result. If a `ZeroDivisionError` exception occurs, the function prints an error message. The `else` block is executed if no exceptions occurred, and the `finally` block is always executed.

[OceanofPDF.com](http://OceanofPDF.com)

## Assert Keyword in Python

The assert keyword in Python is used to check for conditions that should be true during the execution of a program. If the condition is True, the program continues to run. If the condition is False, the program raises an AssertionError exception.

The assert keyword is often used to test the correctness of a program, especially during development and debugging. It can also be used to check the validity of user input or the output of a function.

Here's an example of how you can use the assert keyword in Python:

### Code

```
# Define a function that calculates the area of a circle
```

```
def area(radius):
```

```
    assert radius > 0, "Radius must be positive" # Check that the radius is positive
```

```
    return 3.14 * radius**2 # Calculate the area
```

```
# Call the function with a valid argument
```

```
print(area(5)) # Output: 78.5
```

```
# Call the function with an invalid argument (radius is zero)
```

```
print(area(0)) # Output: AssertionError
```

In this example, the area() function calculates the area of a circle given the radius. The assert statement checks that the radius is positive, and raises an AssertionError if it is not.



[OceanofPDF.com](http://OceanofPDF.com)

Here's a one more example of how you can use the assert keyword in Python:

Code

```
def add(x, y):  
    assert isinstance(x, int) and isinstance(y, int), "Inputs must be integers" #  
    Test the input types  
    return x + y
```

```
print(add(10, 5)) # Output: 15
```

```
print(add(10, "5")) # Output: AssertionError: Inputs must be integers
```

In this example, the add() function adds two numbers and returns the result. The assert statement tests whether the input values are integers, and raises an AssertionError if they are not. If the inputs are valid, the function returns the result of the addition.

Keep in mind that the assert keyword should not be used to test conditions that can be influenced by external factors, such as user input or external data. It is intended for use in debugging and testing, and should not be used in production code.

# Opening & Closing Files in Python

In Python, you can open a file using the built-in `open()` function. The `open()` function takes two arguments: the file name and the mode in which the file should be opened.

Here are some examples of how you can open a file in Python:

Code

```
# Open a file in read mode
```

```
f = open("myfile.txt", "r")
```

```
# Open a file in write mode
```

```
f = open("myfile.txt", "w")
```

```
# Open a file in append mode
```

```
f = open("myfile.txt", "a")
```

In the first example, the `open()` function opens the file "myfile.txt" in read mode (indicated by the "r" parameter). This allows you to read the contents of the file.

In the second example, the `open()` function opens the file "myfile.txt" in write mode (indicated by the "w" parameter). This creates a new file or overwrites the existing file with the same name.

In the third example, the `open()` function opens the file "myfile.txt" in append mode (indicated by the "a" parameter). This allows you to add new data to the end of the file without overwriting the existing contents.

It is important to close the file when you are finished using it, to free up system resources. You can close a file using the **close() method**:

Code

```
f = open("myfile.txt", "r")  
# Read the contents of the file  
contents = f.read()  
# Close the file  
f.close()
```

Alternatively, you can use a with statement to automatically close the file after the indented block of code is executed:

Code

```
with open("myfile.txt", "r") as f:  
    # Read the contents of the file  
    contents = f.read()  
    # The file is automatically closed when the block of code is finished  
    executing
```

# How to Read a File in Python

In Python, you can read the contents of a file using the `read()` method of the file object.

Here's an example of how you can read a file in Python:

Code

```
# Open the file in read mode
with open("myfile.txt", "r") as f:
    # Read the contents of the file and store them in a variable
    contents = f.read()

    print(contents) # Print the contents of the file
```

In this example, the `open()` function opens the file "myfile.txt" in read mode (indicated by the "r" parameter). The file is opened using a `with` statement, which automatically closes the file after the indented block of code is executed. The `read()` method reads the contents of the file and stores them in the `contents` variable.

The `read()` method returns a string containing the contents of the file. If the file is large, you can use the `read()` method in a loop to read the file in chunks:

Code

```
# Open the file in read mode
with open("myfile.txt", "r") as f:
    # Read the file in chunks of 1024 bytes
    chunk = 1024

    contents = ""
```

```
while True:
    data = f.read(chunk)
    if not data: # If there is no more data to read
        break # Exit the loop
    contents += data # Add the data to the contents
print(contents) # Print the contents of the file
```

[OceanofPDF.com](http://OceanofPDF.com)

# How to Delete File in Python

In Python, you can delete a file using the `os` module and the `remove()` function.

Here's an example of how you can delete a file in Python:

Code

```
import os
```

```
# Delete the file "myfile.txt"
```

```
os.remove("myfile.txt")
```

In this example, the `os` module is imported, and the `remove()` function is used to delete the file `"myfile.txt"`.

Keep in mind that the `remove()` function will raise a `FileNotFoundError` if the file does not exist. You can use a `try-except` block to handle this error:

Code

```
import os
```

```
try:
```

```
    # Try to delete the file "myfile.txt"
```

```
    os.remove("myfile.txt")
```

```
except FileNotFoundError:
```

```
    # The file does not exist, so do nothing
```

Pass

[OceanofPDF.com](http://OceanofPDF.com)



## With Statement in Python

In Python, the with statement is used to wrap the execution of a block of code with methods defined by a context manager. A context manager is an object that defines the methods `__enter__()` and `__exit__()`, which are used to set up and tear down a block of code.

The with statement makes it easier to work with resources that need to be cleaned up after use, such as files or network connections. It ensures that the resource is properly released after the block of code is executed, even if an exception is raised.

Here's an example of how you can use the with statement to open and read a file in Python:

Code

```
with open("myfile.txt", "r") as f:
```

```
    # Read the contents of the file
```

```
    contents = f.read()
```

```
    print(contents) # Print the contents of the file
```

In this example, the `open()` function opens the file "myfile.txt" in read mode (indicated by the "r" parameter). The file is opened using a with statement, which automatically closes the file after the indented block of code is executed. The `read()` method reads the contents of the file and stores them in the `contents` variable.

The with statement is equivalent to the following code:

Code

```
f = open("myfile.txt", "r")
```

try:

# Read the contents of the file

contents = f.read()

print(contents) # Print the contents of the file

finally:

# Close the file

f.close()

[OceanofPDF.com](http://OceanofPDF.com)

# Python Modules

In Python, a module is a Python file (with a .py extension) that contains Python code. Modules allow you to define functions, classes, and variables that you can reuse in other Python code.

You can create a module by saving your Python code in a file with a .py extension. For example, you can create a module called mymodule.py that contains the following code:

Code

```
# mymodule.py
```

```
def add(a, b):
```

```
    return a + b
```

```
def subtract(a, b):
```

```
    return a - b
```

To use the module in your Python code, you need to import it using the import statement. For example:

Code

```
import mymodule
```

```
result = mymodule.add(2, 3)
```

```
print(result) # Output: 5
```

[OceanofPDF.com](http://OceanofPDF.com)

In this example, the import statement imports the module mymodule, and the add() function is called with the arguments 2 and 3. The output of the function is printed to the console.

You can also import specific functions or variables from a module using the from keyword:

Code

```
from mymodule import add
```

```
result = add(2, 3)
```

```
print(result) # Output: 5
```

In this example, only the add() function is imported from the mymodule module. You can then use the function without specifying the module name.

[OceanofPDF.com](http://OceanofPDF.com)

## Packages in Python and Import Statement

In Python, a package is a collection of modules (Python files with Python code) that provide a specific functionality. Packages are a way to organize your Python code and make it easier to reuse and share with others.

To use a package in your Python code, you need to import it using the import statement. The import statement allows you to import a specific module or all the modules in a package.

Here's an example of how you can import a module from a package in Python:

Code

```
# Import the module "mymodule" from the package "mypackage"
```

```
import mypackage.mymodule
```

```
# Use the functions and variables defined in the module
```

```
result = mypackage.mymodule.add(2, 3)
```

```
print(result) # Output: 5
```

In this example, the import statement imports the module "mymodule" from the package "mypackage". You can then use the functions and variables defined in the module by specifying the package name and the module name, separated by a dot.

You can also import all the modules in a package using the \* operator:

Code

```
# Import all the modules in the package "mypackage"
from mypackage import *
# Use the functions and variables defined in the modules
result = add(2, 3)
print(result) # Output: 5
```

In this example, the import statement imports all the modules in the package "mypackage". You can then use the functions and variables defined in the modules without specifying the package name.

## Python Collection Module

The collections module is a built-in Python module that provides additional data types for storing and manipulating collections of data. These data types include specialized containers like Counter, OrderedDict, defaultdict, and namedtuple, which are built on top of the standard Python data types such as list, dict, and tuple.

Here's a brief overview of some of the data types provided by the collections module:

**Counter:** A subclass of dict that counts the number of occurrences of each element in a list or iterable.

**OrderedDict:** A subclass of dict that maintains the order of the keys as they are inserted.

**defaultdict:** A subclass of dict that provides a default value for a key that is not found in the dictionary.

namedtuple: A subclass of tuple that allows you to access the elements of the tuple using named attributes instead of indices.

[OceanofPDF.com](http://OceanofPDF.com)



Here's an example of how you can use the Counter class from the collections module:

Code

```
from collections import Counter
```

```
# Count the number of occurrences of each element in a list
```

```
c = Counter([1, 2, 3, 1, 2, 3, 1, 2, 3])
```

```
print(c) # Output: Counter({1: 3, 2: 3, 3: 3})
```

In this example, the Counter class is imported from the collections module and used to count the number of occurrences of each element in the list [1, 2, 3, 1, 2, 3, 1, 2, 3]. The output is a Counter object that contains the count of each element.

[OceanofPDF.com](http://OceanofPDF.com)

# Regular Expression in Python

In Python, a regular expression is a sequence of characters that defines a search pattern. Regular expressions are used to match and manipulate strings, or to check if a string matches a specific pattern.

You can use regular expressions in Python by importing the `re` module and using the functions and methods provided by the module.

Here's a simple example of how you can use regular expressions in Python to match a string:

Code

```
import re
```

```
# Check if the string "hello" matches the pattern "^h.*o$"
```

```
result = re.match("^h.*o$", "hello")
```

```
if result:
```

```
    print("Match found!")
```

```
else:
```

```
    print("Match not found.")
```

In this example, the `re.match()` function is used to check if the string "hello" matches the pattern `"^h.*o$"`. The `^` and `$` characters indicate the start and end of the string, and the `.*` pattern matches any number of characters in between.

If the string matches the pattern, the `match()` function returns a `Match` object. If the string does not match the pattern, the `match()` function returns

None.

[OceanofPDF.com](http://OceanofPDF.com)

You can also use the `findall()` function to find all the occurrences of a pattern in a string:

Code

```
import re
```

```
# Find all the occurrences of the pattern "e" in the string "hello"
```

```
result = re.findall("e", "hello")
```

```
print(result) # Output: ['e']
```

In this example, the `re.findall()` function returns a list of all the occurrences of the pattern "e" in the string "hello".

[OceanofPDF.com](http://OceanofPDF.com)

## Python Datetime

The datetime module is a built-in Python module that provides classes for working with dates and times. The datetime module defines several classes that represent different aspects of dates and times, such as datetime, date, time, timedelta, and tzinfo.

Here's a brief overview of some of the classes provided by the datetime module:

- datetime: Represents a specific point in time, including the date and the time.
- date: Represents a specific date (year, month, and day) without a time.
- time: Represents a specific time (hour, minute, second, and microsecond) without a date.
- timedelta: Represents a duration, which can be used to perform arithmetic with datetime objects.
- tzinfo: Represents a time zone.

Here's an example of how you can use the datetime module to create a datetime object and perform arithmetic with it:

Code

```
from datetime import datetime, timedelta  
  
# Create a datetime object representing the current date and time  
now = datetime.now()  
  
print(now) # Output: 2022-12-27 15:23:45.547829
```

```
# Add 1 day to the current date  
tomorrow = now + timedelta(days=1)  
print(tomorrow) # Output: 2022-12-28 15:23:45.547829
```

```
# Subtract 3 hours from the current time  
three_hours_ago = now - timedelta(hours=3)  
print(three_hours_ago) # Output: 2022-12-27 12:23:45.547829
```

In this example, the `datetime.now()` function is used to create a `datetime` object representing the current date and time. The `timedelta` class is used to add or subtract a specified number of days or hours from the `datetime` object.

[OceanofPDF.com](https://oceanofpdf.com)

# Module 1 Project 1 -10

[OceanofPDF.com](http://OceanofPDF.com)

## 1. Age Calculator GUI

In this age calculator app, users can type in their date of birth, and the app will calculate and display their age.

## Step 1:

First of all, we need to import two libraries into our code. The first one is the `*tkinter*` library. Then, we need the `*datetime*` library to work with dates.

## Step 2:

Now, let's create a simple window for our app and name it as `*Age Calculator*`.

## Step 3:

Then, we are going to create four labels, each for the name, year, month, and the date, and put them in the grid.

We will create entry fields to get the user inputs corresponding to all the labels created. Put them at the right side of the corresponding labels using the `*grid*` method.

## Step 4:



Then, we are going to define a function, which will calculate the age of the user by subtracting the user's birth date from today's date. We name that function as ``ageCalc()``.

Then, we create a ``Label`` area that will display the age of the user as output in the function itself.

**## Step 5:**

Then, we are going to create a button for users to submit their input values. We link the button to the ``ageCalc`` function.

Finally, let's run everything inside the window using the ``mainloop()`` method.

### **Source Code:**

```
# import libraries
from tkinter import *
from datetime import date
```

```
# initialized window
```

```
root = Tk()
```

```
root.geometry('280x300')
```

```
root.resizable(0, 0)
```

```
root.title('Age Calculator')
```

```
statement = Label(root)
```

```
# defining the function for calculating age
```

```
def ageCalc():
```

```
    global statement
```

```
    statement.destroy()
```

```
    today = date.today()
```

```
    birthDate = date(int(yearEntry.get()), int(
```

```
        monthEntry.get()), int(dayEntry.get()))
```

```
        age = today.year - birthDate.year
```

```
        if today.month < birthDate.month or today.month ==  
        birthDate.month and today.day < birthDate.day:
```

```
            age -= 1
```

```
        statement = Label(text=f"{nameValue.get()}'s age is {age}.")
```

```
        statement.grid(row=6, column=1, pady=15)
```

```
# creating a label for person's name to display
```

```
l1 = Label(text="Name: ")
```

```
l1.grid(row=1, column=0)
```

```
nameValue = StringVar()
```

```
# creating a entry box for input
```

```
nameEntry = Entry(root, textvariable=nameValue)
```

```
nameEntry.grid(row=1, column=1, padx=10, pady=10)
```

```
# label for year in which user was born
```

```
l2 = Label(text="Year: ")
```

```
l2.grid(row=2, column=0)
```

```
yearValue = StringVar()
```

```
yearEntry = Entry(root, textvariable=yearValue)
```

```
yearEntry.grid(row=2, column=1, padx=10, pady=10)
```

```
# label for month in which user was born
```

```
l3 = Label(text="Month: ")
```

```
l3.grid(row=3, column=0)
```

```
monthValue = StringVar()
```

```
monthEntry = Entry(root, textvariable=monthValue)
```

```
monthEntry.grid(row=3, column=1, padx=10, pady=10)
```

```
# label for day/date on which user was born
```

```
l4 = Label(text="Day: ")
```

```
l4.grid(row=4, column=0)
```

```
dayValue = StringVar()
dayEntry = Entry(root, textvariable=dayValue)
dayEntry.grid(row=4, column=1, padx=10, pady=10)
```

```
# create a button for calculating age
button = Button(text="Calculate age", command=ageCalc)
button.grid(row=5, column=1)

# infinite loop to run program
root.mainloop()
```

[OceanofPDF.com](http://OceanofPDF.com)

## 2. Attachment Downloader

The script downloads gmail attachment(s) in no time!

### # Setup Instructions

The script uses ezgmail module (to know more visit <https://pypi.org/project/EZGmail/>). To install please type the following command in your bash.

```
```bash
```

```
pip install EZGmail
```

```
```
```

Once the module is installed you will need to download credentials.json file by going to [developers.google.com] (<https://developers.google.com/gmail/api/quickstart/python>) and clicking the Enable the Gmail API button (select "Desktop app" as OAuth Client in second step).

Once you have credentials.json (in root directory of project folder), the first time you run the script it will open up a browser window asking you to log in to your Gmail account and allow "Quickstart" app to access it. A token.json file will be generated (in root directory of project folder) which your script can use to access your account.

```

import ezgmail

def attachmentdownload(resultthreads):
    # Two Objects used in code are GmailThread and GmailMessage
    # 1. GmailThread - Represents conversation threads
    # 2. GmailMessage - Represents individual emails within Threads
    countofresults = len(resultthreads)
    try:
        for i in range(countofresults):
            # checks whether the count of messages in threads is greater than 1
            if len(resultthreads[i].messages) > 1:
                for j in range(len(resultthreads[i].messages)):
                    resultthreads[i].messages[
                        j].downloadAllAttachments() # downloads attachment(s) for
individual messages
            else:
                # downloads attachment(s) for single message
                resultthreads[i].messages[0].downloadAllAttachments()
        print("Download complete. Please check your root directory.")
    except:
        raise Exception("Error occured while downloading attachment(s).")

if __name__ == '__main__':
    query = input("Enter search query: ")
    # appending to make sure the result threads always has an
attachment

```

```

newquery = query + " + has:attachment"

# search functions accepts all the operators described at
https://support.google.com/mail/answer/7190?hl=en
resultthreads = ezgmail.search(newquery)

if len(resultthreads) == 0:
    # Executed if results don't have attachment
    print("Result has no attachments:")
else:
    print("Result(s) with attachments:")
    for threads in resultthreads:
        # prints the subject line of email thread in results
        print(f"Email Subject: {threads.messages[0].subject}")
    try:
        ask = input(
            "Do you want to download attachment(s) in result(s)
(Yes/No)? ") # Allows user to decide whether they want to download
attachment(s) or not
        if ask == "Yes":
            # calls the function that downloads attachment(s)
            attachmentdownload(resultthreads)
        else:
            print("Program exited")
    except:
        print("Something went wrong")

```

### 3. Auto Birthday Wisher

One forgets to send birthday wishes to friends many times. At such times an automatic birthday wisher comes handy. An automatic birthday wisher via email makes one's life easy. It will send the birthday wishes to friends via email automatically via a server and using an excel sheet to store the data of friends and their birthdays along with email id. It'll send the wishes to friends for all the upcoming years until we stop the server.

## Setup instructions

In order to run this script, You just need the following modules:

- **Pandas** is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool,

built on top of the Python programming language.

```
```bash
```

```
pip install pandas
```

```
```
```

- **Datetime** is a module used for Encapsulation of date/time values.



```
```bash
pip install DateTime
```
```

- **smtplib** module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

## ## Configuration

1. Assign the Gmail Id of sender to the GMAIL\_ID variable in \*line 10\* of **"Auto B'Day Wisher.py"** file. (e.g. 'xyz@gmail.com')
2. Similar to first step assign the Gmail password of sender to the GMAIL\_PSWD variable in \*line 11\* of **"Auto B'Day Wisher.py"** file. (e.g. '1234')
3. In **"data.xlsx"** file insert the name of the receiver in second column under \*Name\*. Similarly update the **Birthday** field with the birth date of receiver in the given format\*("%dd-%mm-%YYYY")\*. Update the **Dailogue** field with a short message you want to send and the **Email** field with the email of the receiver.
4. Make sure to give permission to your google account from which you're sending email to **Allow less secure apps**. Just turn this **"ON"** from [here](<https://support.google.com/accounts/answer/6010255?hl=en#zippy=%2Cif-less-secure-app-access-is-off-for-your-account>).
5. Run the command

```
```bash
python "Auto B'Day Wisher.py"
```
```

Source Code :

```
# Pandas library is used for importing and reading the data
import pandas as pd
# datetime module is used for fetching the dates
import datetime
import smtplib
    # smtp library used for sending mail
import os

current_path = os.getcwd()
print(current_path)
# Changing the Path of the directory in which you are currently
working
os.chdir(current_path)

# Give your mail here from which you want to send the wishes
GMAIL_ID = input("Enter your email: ")
# Give your mail password
GMAIL_PSWD = input("Enter password for your email mentioned
above: ")

def sendEmail(to, sub, msg):
```

```
print(f"Email to {to} sent: \nSubject: {sub} ,\nMessage: {msg}")
# creating server to send mail
s = smtplib.SMTP('smtp.gmail.com', 587)
# start a TLS session
s.starttls()
# the function will login with your Gmail credentials
s.login(GMAIL_ID, GMAIL_PSWD)
# sending the mail
s.sendmail(GMAIL_ID, to, f"Subject: {sub} \n\n {msg}")
s.quit()
```

```
if __name__ == "__main__":
    # the datasheet where the data of the friends is stored
    df = pd.read_excel("data.xlsx")
    today = datetime.datetime.now().strftime("%d-%m")
    yearNow = datetime.datetime.now().strftime("%Y")

    writeInd = []
    for index, item in df.iterrows():
        bday = item['Birthday']
        bday = datetime.datetime.strptime(bday, "%d-%m-%Y")
        bday = bday.strftime("%d-%m")
        if(today == bday) and yearNow not in str(item['LastWishedYear']):
            # calling the sendmail function
```

```
sendEmail(item['Email'], "Happy Birthday", item['Dialogue'])  
writeInd.append(index)
```

```
if writeInd != None:
```

```
    for i in writeInd:
```

```
        oldYear = df.loc[i, 'LastWishedYear']
```

```
        df.loc[i, 'LastWishedYear'] = str(oldYear) + ", " + str(yearNow)
```

```
df.to_excel('data.xlsx', index=False)
```

[OceanofPDF.com](http://OceanofPDF.com)

## 4. Automatic\_Backup

Automatic Backup and Compression of large file, speed up using Threading.

Multithreading helps in achieving MultiTasking using threads.

Along with Multithreading `gzip` has been used for Compressing large files.

I have made sur that the script required no additional library other than the basic standard packages.

## Setup instructions

As explained, there is no specific `requirements.txt`. So, no additional library or packages are required.

There are two files one `python` and one `notebook`. [`Auto_Backup.py`] (`./Auto_Backup.py`)

is the script that can be quickly used to back up the desired file.

For greater understanding of the script and proof of concept, refer to [`Auto_Backup.ipynb`] (`./Auto_Backup.ipynb`).

The Notebook has further illustrated the Script and is much more detailed.

...

Example Usage -

```
python Auto_backup.py -t ./MIREX_Backup -s ./MIREX_Dataset -c 100000
```

Source Code:

```
""" Simple backup script which just creates the root structure in an
other
folder and syncs everything which recursively lies within one of the
source
folders. For files bigger than a threshold they are first gzipped."""
```

```
import argparse
import gzip
import os
import shutil
import sys
import threading
```

```
def parse_input():
    """
    Argument Parser function, for parsing CLI.
    Returns: parse_args()

    """
    parser = argparse.ArgumentParser()
    parser.add_argument('-t',
```

```

        '--target',
        nargs=1,
        required=True,
        help='Target Backup folder')
parser.add_argument('-s',
                    '--source',
                    nargs='+',
                    required=True,
                    help='Source Files to be added')
parser.add_argument('-c',
                    '--compress',
                    nargs=1,
                    type=int,
                    help='Gzip threshold in bytes, Deafault 1024KB',
                    default=[1024000])
# Default Threshold is 1024KB

# Help is triggered when there is no Input Provided
if len(sys.argv) == 1:
    parser.print_help()
    sys.exit()

    return parser.parse_args()
def size_if_newer(source, target):
    """

```

If there is a difference in file size, this function reports the difference.

Args:

source: Source Path

target: Target for ZIP file

Returns: The Size difference

```
"""
src_stat = os.stat(source)
try:
    target_ts = os.stat(target).st_mtime
except FileNotFoundError:
    try:
        target_ts = os.stat(target + '.gz').st_mtime
    except FileNotFoundError:
        target_ts = 0

# The time difference of one second is necessary since subsecond
accuracy
# of os.st_mtime is striped by copy2
return src_stat.st_size if (src_stat.st_mtime - target_ts > 1) else False

def threaded_sync_file(source, target, compress):
    """
```



Multithreading for Synced files.

Args:

source: Source Path

target: Target for ZIP file

compress: The compression threshold

Returns: The threads

```
"""
```

```
size = size_if_newer(source, target)
```

```
if size:
```

```
    thread = threading.Thread(target=transfer_file,
```

```
                               args=(source, target, size > compress))
```

```
    thread.start()
```

```
    return thread
```

```
def sync_file(source, target, compress):
```

```
    """
```

Synchronizing files

Args:

source: Source Path

target: Target for ZIP file

compress: The compression threshold

```
"""
```

```
size = size_if_newer(source, target)
```

```
if size:
```

```
    transfer_file(source, target, size > compress)
```

```
def transfer_file(source, target, compress):
```

```
    """
```

```
    Transferring files
```

```
    Args:
```

```
        source: Source Path
```

```
        target: Target for ZIP file
```

```
        compress: The compression threshold
```

```
    """
```

```
    try:
```

```
        if compress:
```

```
            with gzip.open(target + '.gz', 'wb') as target_fid:
```

```
                with open(source, 'rb') as source_fid:
```

```
                    target_fid.writelines(source_fid)
```

```
            print('Compress {}'.format(source))
```

```
        else:
```

```
            shutil.copy2(source, target)
```

```
            print('Copy {}'.format(source))
```

```
    except FileNotFoundError:
```

```

        os.makedirs(os.path.dirname(target))
        transfer_file(source, target, compress)
def sync_root(root, arg):
    """
    Synchronize Root with Target

    """
    target = arg.target[0]
    compress = arg.compress[0]
    threads = []

    for path, _, files in os.walk(root):
        for source in files:
            source = path + '/' + source
            threads.append(
                threaded_sync_file(source, target + source, compress))
    # sync_file(source, target + source, compress)
    for thread in threads:
        thread.join()
    if __name__ == '__main__':

```

```

arg = parse_input()
    print('----- Start copy -----')

    print('_____')
    for root in arg.source:
        sync_root(root, arg)

    print('_____')
    print('----- Done Done! -----')
"""

```

Example Usage-

```

> python Auto_Backup.py --target ./Backup_Folder --source
./Source_Folder
"""

```

[OceanofPDF.com](http://OceanofPDF.com)

## 5. Python Script to auto fill Google Forms

This is a python script which can helps you to fillout the google form automatically by bot.

## Steps to make it run

1. Clone/Download this repository

'''

git clone clone\_path

'''

2. Downlaod the required packages

'''

pip install -r requirements.txt

'''

3. To run the script you need to use the following command

'''

python app.py

'''

4. To run this script you need to have selenium installed and you need to have geckodriver.

you can download geckodriver from here:

<https://github.com/mozilla/geckodriver/releases>

After downloading the geckodriver, need to set-path for that as shown below:

```
```
```

```
webdriver.Firefox(executable_path=
```

```
'D:\Selenium_RiponAlWasim\geckodriver-v0.18.0-  
win64\geckodriver.exe')
```

```
```
```

```
### working
```

1. You can add or delete the records in .csv file.

2. Initially the number of responses in the google-form will be null(no-entries):

screenshot is attached below:

3. After running the script, the bot will automatically take the records from .csv file and it will the form.

Source Code:

```
import csv
import time
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

inputName =
'/html/body/div/div[2]/form/div[2]/div/div[2]/div[1]/div/div/div[2]/div/div[1]/div/div[1]/input'

inputEmailID =
'/html/body/div/div[2]/form/div[2]/div/div[2]/div[2]/div/div/div[2]/div/div[1]/div/div[1]/input'

inputPhone =
'/html/body/div/div[2]/form/div[2]/div/div[2]/div[3]/div/div/div[2]/div/div[1]/div/div[1]/input'

# Submit Button Xpath
Submit = '/html/body/div/div[2]/form/div[2]/div/div[3]/div[1]/div/div'

def sleep():
    time.sleep(3)

browser = webdriver.Firefox(
    executable_path='C:\geckodriver-v0.28.0-win64\geckodriver.exe')
    browser.get(
```

```
'https://docs.google.com/forms/d/e/1FAIpQLScBejWF809oacjZlvkci  
XREi50fyHcq75l988KDJo3ycG7xkg/viewform'
```

```
)
```

```
name = []
```

```
email = []
```

```
phone = []
```

```
with open("input.csv", "r") as f_input:
```

```
    csv_input = csv.DictReader(f_input)
```

```
    for row in csv_input:
```

```
        name.append(row['name'])
```

```
        email.append(row['email'])
```

```
        phone.append(row['phone_number'])
```

```
    # print(name,email,phone)
```

```
i = 0
```

```
while i < len(name):
```

```
    browser.find_element_by_xpath(inputName).send_keys(name[i])
```

```
    browser.find_element_by_xpath(inputEmailID).send_keys(email[i])
```

```
    browser.find_element_by_xpath(inputPhone).send_keys(phone[i])
```

```
    sleep()
```

```
    browser.find_element_by_xpath(Submit).click()
```

```
    i += 1
```

```
    sleep()
```



```
browser.back()
```

```
sleep()
```

```
# print(name,email,phone)
```

```
browser.quit()
```

[OceanofPDF.com](http://OceanofPDF.com)



## 6. Automate Facebook bot

## Automate Facebook bot Functionalities:

- On running the script, it posts your message in the groups whose id is given by the user

## Automate Facebook bot Instructions:

### Step 1:

Open Terminal

### Step 2:

Locate to the directory where python file is located

### Step 3:

Run the command: `python script.py/python3 script.py`

### Step 4:

Sit back and Relax. Let the Script do the Job.

### Requirements

- pyautogui
- time

**Source Code:**

```
import pyautogui
import time
import webbrowser
from selenium import webdriver
from time import sleep
from webdriver_manager.chrome import ChromeDriverManager
from getpass import getpass
```

```
LOGIN_URL = 'https://www.facebook.com/login.php'
num = str(input("Enter group ids separated by commas: "))
lists = num.split(",")
groupid = []
for i in lists:
    groupid.append(i)
```

```
message = input("Enter your message: ")
```

```
class FacebookLogin():
```

```
    def __init__(self, email, password, browser='Chrome'):
```

```
        # Store credentials for login
```

```
        self.email = email
```

```
        self.password = password
```

```
        if browser == 'Chrome':
```

```
            # Use chrome
```

```
            self.driver = webdriver.Chrome(
```

```
                executable_path=ChromeDriverManager().install())
```

```
        self.driver.get(LOGIN_URL)
```

```
        time.sleep(1) # Wait for some time to load
```

```
    def login(self):
```

```
        email_element = self.driver.find_element_by_id('email')
```

```
        email_element.send_keys(self.email) # Give keyboard input
```

```
        password_element = self.driver.find_element_by_id('pass')
```

```
        password_element.send_keys(self.password) # Give password as  
input too
```

```
        login_button = self.driver.find_element_by_id('loginbutton')
```

```
        login_button.click() # Send mouse click
```

```
        time.sleep(2) # Wait for 2 seconds for the page to show up
```

```
        for i in range(len(groupid)):
```

```

link = 'https://facebook.com/groups/'+groupid[i]
self.driver.get(link)
print("Waiting for few seconds .....")
time.sleep(45)
self.driver.find_element_by_class_name(
    'a8c37x1j ni8dbmo4 stjgntxs l9j0dhe7').click()
time.sleep(7)

self.driver.switch_to.active_element.send_keys("message")
time.sleep(7)

self.driver.find_element_by_class_name(
    'a8c37x1j ni8dbmo4 stjgntxs l9j0dhe7 ltmtdrg g0qnabr5').click()
time.sleep(7)

if __name__ == '__main__':
    # Enter your login credentials here
    usr = input('Enter Email Id:')
    pwd = getpass('Enter Password:')
    fb_login = FacebookLogin(email=usr, password=pwd,
browser='Chrome')
    fb_login.login()

# time.sleep(5)

```

[OceanofPDF.com](http://OceanofPDF.com)

## 7. Automatic-Certificate-Generator

This python script automatically generates certificate by using a certificate template and csv file which contains the list of names to be printed on certificate. It downloads the certificate in the directory i.e. in **\*\*pictures\*\*** folder where the scripts are there.

**## Installation**

First of all install [python](<https://www.python.org/downloads/>) on your system.

```
```bash
```

```
pip install PI
```

```
pip install Pandas
```

```
pip install pillow
```

```
```
```



## Source Code:

```
# importing packages & modules
from PIL import Image, ImageDraw, ImageFont
import pandas as pd
import os

# Implementation to generate certificate
df = pd.read_csv('list.csv')
font = ImageFont.truetype('arial.ttf', 60)
for index, j in df.iterrows():
    img = Image.open('certificate.png')
    draw = ImageDraw.Draw(img)
    draw.text(xy=(150, 250),
              text='{}'.format(j['name']),
              fill=(0, 0, 0),
              font=font) # customization
    img.save('pictures/{}.png'.format(j['name']))
```

[OceanofPDF.com](http://OceanofPDF.com)

## 8. Automate FB login

```
from selenium import webdriver
import time
from selenium.webdriver.common.keys import Keys

user_id=input('Enter User Id of your Fb Account :) # Take user id and
password as input from the user
password=input('Enter the password :)

print(user_id)
print(password)

cd='C:\\webdrivers\\chromedriver.exe' #path to your chrome driver

browser= webdriver.Chrome(cd)
browser.get('https://www.facebook.com/')

user_box = browser.find_element_by_id("email")    # For detecting
the user id box
```

```
user_box.send_keys(user_id) # Enter the  
user id in the box
```

```
password_box = browser.find_element_by_id("pass") # For  
detecting the password box
```

```
password_box.send_keys(password) # For  
detecting the password in the box
```

```
login_box = browser.find_element_by_id("u_0_b") # For detecting  
the Login button
```

```
login_box.click()
```

[OceanofPDF.com](http://OceanofPDF.com)

## 9. Brick Breaker Game

Brick Breaker (The game) is a Breakout clone which the player must smash a wall of bricks by deflecting a bouncing ball with a paddle. The paddle may move horizontally and is controlled with the side arrow keys.

## Setup instructions

Run ``python/python3 brick_breaker.py``

Source Code:

```
import pygame
from pygame.locals import *
```

```
pygame.init()
```

```
'''
```

```
Defining gaming window size and font
```

```
'''
```

```
Window_width = 500
```

```
Window_height = 500
```

```
window = pygame.display.set_mode((Window_width, Window_height))
```

```
pygame.display.set_caption('Brickstroy')
```

```
font = pygame.font.SysFont('Arial', 30)
```

```
'''
```

```
Defining Bricks colour
```

```
'''
```

```
O_brick = (255, 100, 10)
```

```
w_brick = (255, 255, 255)
```

```
g_brick = (0, 255, 0)
```

```
black = (0, 0, 0)
```

```
game_rows = 6
game_coloumns = 6
clock = pygame.time.Clock()
frame_rate = 60
my_ball = False
game_over = 0
score = 0
```

```
class Ball():
    """
    Creating ball for the game
    """
```

```
    def __init__(self, x, y):

        self.radius = 10
        self.x = x - self.radius
        self.y = y - 50
        self.rect = Rect(self.x, self.y, self.radius * 2, self.radius * 2)
        self.x_speed = 4
        self.y_speed = -4
        self.max_speed = 5
```

```
self.game_over = 0
```

```
def motion(self):
```

```
    collision_threshold = 5
```

```
    block_object = Block.bricks
```

```
    brick_destroyed = 1
```

```
    count_row = 0
```

```
    for row in block_object:
```

```
        count_item = 0
```

```
        for item in row:
```

```
            # check collision with gaming window
```

```
            if self.rect.colliderect(item[0]):
```

```
                if abs(self.rect.bottom - item[0].top) < collision_threshold and  
self.y_speed > 0:
```

```
                    self.y_speed *= -1
```

```
                if abs(self.rect.top - item[0].bottom) < collision_threshold and  
self.y_speed < 0:
```

```
                    self.y_speed *= -1
```

```
                if abs(self.rect.right - item[0].left) < collision_threshold and  
self.x_speed > 0:
```

```
                    self.x_speed *= -1
```

```
                if abs(self.rect.left - item[0].right) < collision_threshold and  
self.x_speed < 0:
```

```
                    self.x_speed *= -1
```

```
            if block_object[count_row][count_item][1] > 1:
```

```

        block_object[count_row][count_item][1] -= 1
    else:
        block_object[count_row][count_item][0] = (0, 0, 0, 0)

    if block_object[count_row][count_item][0] != (0, 0, 0, 0):
        brick_destroyed = 0
        count_item += 1
    count_row += 1

if brick_destroyed == 1:
    self.game_over = 1

# check for collision with bricks
if self.rect.left < 0 or self.rect.right > Window_width:
    self.x_speed *= -1

if self.rect.top < 0:
    self.y_speed *= -1
if self.rect.bottom > Window_height:
    self.game_over = -1

# check for collision with base
if self.rect.colliderect(user_basepad):
    if abs(self.rect.bottom - user_basepad.rect.top) < collision_threshold
and self.y_speed > 0:
        self.y_speed *= -1

```



```
self.x_speed += user_basepad.direction
if self.x_speed > self.max_speed:
    self.x_speed = self.max_speed
elif self.x_speed < 0 and self.x_speed < -self.max_speed:
    self.x_speed = -self.max_speed
else:
    self.x_speed *= -1

self.rect.x += self.x_speed
self.rect.y += self.y_speed

return self.game_over
```

```
def draw(self):
    pygame.draw.circle(window, (0, 0, 255), (self.rect.x +
        self.radius, self.rect.y + self.radius), self.radius)
    pygame.draw.circle(window, (255, 255, 255), (self.rect.x +
        self.radius, self.rect.y + self.radius), self.radius, 1)
```

```
def reset(self, x, y):

    self.radius = 10
    self.x = x - self.radius
    self.y = y - 50
    self.rect = Rect(self.x, self.y, self.radius * 2, self.radius * 2)
```

```
self.x_speed = 4
self.y_speed = -4
self.max_speed = 5
self.game_over = 0
```

```
class Block():
```

```
    """
```

```
This class will help me create Blocks/bricks of the game
```

```
    """
```

```
    def __init__(self):
```

```
        self.width = Window_width // game_coloumns
```

```
        self.height = 40
```

```
    def make_brick(self):
```

```
        self.bricks = []
```

```
        single_brick = []
```

```
        for row in range(game_rows):
```

```
            brick_row = []
```

```
            for coloumn in range(game_coloumns):
```

```
                x_brick = coloumn * self.width
```

```

        y_brick = row * self.height
        rect = pygame.Rect(x_brick, y_brick, self.width, self.height)
        # assign power to the bricks based on row
    if row < 2:
        power = 3
    elif row < 4:
        power = 2
    elif row < 6:
        power = 1

    single_brick = [rect, power]

    brick_row.append(single_brick)

self.bricks.append(brick_row)

def draw_brick(self):
    for row in self.bricks:
        for brick in row:

            if brick[1] == 3:
                brick_colour = O_brick
            elif brick[1] == 2:
                brick_colour = w_brick
            elif brick[1] == 1:

```

```
brick_colour = g_brick
pygame.draw.rect(window, brick_colour, brick[0])
pygame.draw.rect(window, black, (brick[0]), 1)
```

```
class base():
```

```
'''
```

```
This class is to create the base pad of the game
```

```
'''
```

```
def __init__(self):
```

```
    self.height = 20
```

```
    self.width = int(Window_width / game_coloumns)
```

```
    self.x = int((Window_width / 2) - (self.width / 2))
```

```
    self.y = Window_height - (self.height * 2)
```

```
    self.speed = 8
```

```
    self.rect = Rect(self.x, self.y, self.width, self.height)
```

```
    self.direction = 0
```

```
def slide(self):
```

```
    self.direction = 0
```

```
    key = pygame.key.get_pressed()
```

```
    if key[pygame.K_LEFT] and self.rect.left > 0:
```

```

        self.rect.x -= self.speed
        self.direction = -1
    if key[pygame.K_RIGHT] and self.rect.right < Window_width:
        self.rect.x += self.speed
        self.direction = 1

def draw(self):
    pygame.draw.rect(window, (0, 0, 255), self.rect)
    pygame.draw.rect(window, (255, 255, 255), self.rect, 1)

def reset(self):

    self.height = 20
    self.width = int(Window_width / game_coloumns)
    self.x = int((Window_width / 2) - (self.width / 2))
    self.y = Window_height - (self.height * 2)
    self.speed = 8
    self.rect = Rect(self.x, self.y, self.width, self.height)
    self.direction = 0

def draw_text(text, font, w_brick, x, y):
    """
    Funtion for showing text in gaming window
    """

```

```

    image = font.render(text, True, w_brick)
    window.blit(image, (x, y))

Block = Block()
# Creating Brick
Block.make_brick()
# Defining base pad
user_basepad = base()
ball = Ball(user_basepad.x + (user_basepad.width // 2),
            user_basepad.y - user_basepad.height) # Defining ball

game = True
while game:

    clock.tick(frame_rate)
    window.fill(black)                # Gaming window Background
    Block.draw_brick()                 # Drawing bricks
    user_basepad.draw()                # Drawing user basepad
    ball.draw()                        # Drawing gaming ball

if my_ball:
    user_basepad.slide()
    game_over = ball.motion()
    if game_over != 0:

```

```

my_ball = False

# Game Info on the gaming window
if not my_ball:
    if game_over == 0:
        draw_text('CLICK ANYWHERE TO START', font,
                    w_brick, 90, Window_height // 2 + 100)
    elif game_over == 1:
        draw_text('YOU WON!', font, w_brick, 180, Window_height // 2 +
50)
        draw_text('CLICK ANYWHERE TO RESTART', font,
                    w_brick, 90, Window_height // 2 + 100)
    elif game_over == -1:
        draw_text('GAME OVER!', font, w_brick,
                    180, Window_height // 2 + 50)
        draw_text('CLICK ANYWHERE TO RESTART', font,
                    w_brick, 90, Window_height // 2 + 100)

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        game = False
    if event.type == pygame.MOUSEBUTTONDOWN and my_ball ==
False:
        my_ball = True
        ball.reset(user_basepad.x + (user_basepad.width // 2),
                    user_basepad.y - user_basepad.height)

```

```
user_basepad.reset()
```

```
Block.make_brick()
```

```
pygame.display.update()
```

```
pygame.quit()
```

[OceanofPDF.com](http://OceanofPDF.com)



## 10. Bubble shooter game

How to play Bubble shooter game

- In order to shoot the bubbles click on the space bar.
- Any 3 consecutive same color bubbles.
- Control the arrow with left, right and up arrow keys.

Source Code:

```
import math
import pygame
import copy
import time
import sys
import os
import random
import pygame.gfxdraw
from pygame.locals import *

FPS = 120
winwidth = 940
winhgt = 740
    txthgt = 20
    bubblerad = 20
    bubblewidth = bubblerad * 2
```

```
bubblelyrs = 5  
bubadjst = 5  
strx = winwidth / 2  
strY = winhgt - 26  
arywidth = 25  
aryhgt = 20
```

```
RIGHT = 'right'  
LEFT = 'left'  
blank = '.'
```

```
vblue = (51, 255, 255)  
black = (0, 0, 0)  
white = (255, 255, 255)  
grey = (100, 100, 100)  
blue = (0, 0, 205)  
red = (255, 0, 0)  
white = (255, 255, 255)  
pink = (255, 192, 203)  
lightpink = (255, 182, 193)  
hotpink = (255, 105, 180)  
deeppink = (255, 20, 147)  
cyan = (0, 255, 255)  
peacockblue = (0, 164, 180)  
grapecolor = (128, 49, 167)
```

```
amber = (255, 198, 0)
comic = (0, 174, 239)
lytgray = (217, 217, 214)
peach = (255, 229, 180)
green = (0, 255, 0)
GRAY = (100, 100, 100)
white = (255, 255, 255)
cyan = (0, 255, 255)
black = (0, 0, 0)
```

```
bgcolor = vblue
clrlist = [grey, blue, red, white, pink, peach, hotpink, green,
           deeppink, peacockblue, grapecolor, amber, comic, lytgray]
```

```
class Bubble(pygame.sprite.Sprite):
    def __init__(self, color, row=0, col=0):
        pygame.sprite.Sprite.__init__(self)

        self.rect = pygame.Rect(0, 0, 30, 30)
        self.rect.centerx = int(strx)
        self.rect.centery = strY
        self.speed = 10
        self.color = color
        self.radius = bubblerad
```

```
self.angle = 0
self.row = row
self.col = col
```

```
def update(self):
```

```
    if self.angle == 90:
        xmove = 0
        ymove = self.speed * -1
    elif self.angle < 90:
        xmove = self.xcalc(self.angle)
        ymove = self.ycalc(self.angle)
    elif self.angle > 90:
        xmove = self.xcalc(180 - self.angle) * -1
        ymove = self.ycalc(180 - self.angle)
```

```
    self.rect.x += int(xmove)
    self.rect.y += int(ymove)
```

```
def draw(self):
```

```
    pygame.gfxdraw.filled_circle(
        dispsurf, self.rect.centerx, self.rect.centery, self.radius, self.color)
    pygame.gfxdraw.aacircle(
        dispsurf, self.rect.centerx, self.rect.centery, self.radius, GRAY)
```

```
def xcalc(self, angle):  
    radians = math.radians(angle)  
  
    xmove = math.cos(radians)*(self.speed)  
    return xmove
```

```
def ycalc(self, angle):  
    radians = math.radians(angle)  
  
    ymove = math.sin(radians)*(self.speed) * -1  
    return ymove
```

```
class Ary(pygame.sprite.Sprite):  
    def __init__(self):  
        pygame.sprite.Sprite.__init__(self)  
  
        self.angle = 90  
        arrimg = pygame.image.load('Arrow.png')  
        arrimg.convert_alpha()  
  
        arrowRect = arrimg.get_rect()  
        self.image = arrimg  
        self.transformImage = self.image  
        self.rect = arrowRect
```

```
self.rect.centerx = int(strx)
```

```
self.rect.centery = strY
```

```
def update(self, dir):
```

```
    if (dir == LEFT and self.angle < 180):
```

```
        self.angle += 2
```

```
    elif (dir == RIGHT and self.angle > 0):
```

```
        self.angle -= 2
```

```
    self.transformImage = pygame.transform.rotate(self.image, self.angle)
```

```
    self.rect = self.transformImage.get_rect()
```

```
    self.rect.centerx = int(strx)
```

```
    self.rect.centery = strY
```

```
def draw(self):
```

```
    dispsurf.blit(self.transformImage, self.rect)
```

```
class Score(object):
```

```
    def __init__(self):
```

```
        self.total = 0
```

```
        self.font = pygame.font.SysFont('merlin', 35)
```

```
        self.render = self.font.render(
```

```
            'Score: ' + str(self.total), True, black, white)
```

```
self.rect = self.render.get_rect()
```

```
self.rect.left = 5
```

```
self.rect.bottom = winhgt - 5
```

```
def update(self, dellst):
```

```
    self.total += ((len(dellst)) * 10)
```

```
    self.render = self.font.render(
```

```
        'Score: ' + str(self.total), True, black, white)
```

```
def draw(self):
```

```
    dispsurf.blit(self.render, self.rect)
```

```
def main():
```

```
    global fpsclock, dispsurf, disprect, mainfont
```

```
    pygame.init()
```

```
    fpsclock = pygame.time.Clock()
```

```
    pygame.display.set_caption('Bubble Shooter')
```

```
    mainfont = pygame.font.SysFont('Comic Sans MS', txthgt)
```

```
    dispsurf, disprect = makeDisplay()
```

```
    while True:
```

```
        score, winorlose = rngame()
```

```
endScreen(score, winorlose)
```

```
def rngame():
```

```
    musclist = ['Whatever_It _Takes_OGG.ogg', 'bgmusic.ogg',  
'Goofy_Theme.ogg']
```

```
    pygame.mixer.music.load(musclist[0])
```

```
    pygame.mixer.music.play()
```

```
    track = 0
```

```
    gameclrlst = copy.deepcopy(clrlst)
```

```
    dir = None
```

```
    launchbb = False
```

```
    newbb = None
```

```
    arrow = Ary()
```

```
    bbarr = mkeblkbrd()
```

```
    setbb(bbarr, gameclrlst)
```

```
    nxtbb = Bubble(gameclrlst[0])
```

```
    nxtbb.rect.right = winwidth - 5
```

```
    nxtbb.rect.bottom = winhgt - 5
```

```
    score = Score()
```

```
    while True:
```

```
        dispsurf.fill(bgcolor)
```



```
for event in pygame.event.get():
    if event.type == QUIT:
        terminate()

    elif event.type == KEYDOWN:
        if (event.key == K_LEFT):
            dir = LEFT
        elif (event.key == K_RIGHT):
            dir = RIGHT

    elif event.type == KEYUP:
        dir = None
        if event.key == K_SPACE:
            launchbb = True
        elif event.key == K_ESCAPE:
            terminate()

if launchbb == True:
    if newbb == None:
        newbb = Bubble(nxtbb.color)
        newbb.angle = arrow.angle

    newbb.update()
    newbb.draw()
```

```

if newbb.rect.right >= winwidth - 5:
    newbb.angle = 180 - newbb.angle
elif newbb.rect.left <= 5:
    newbb.angle = 180 - newbb.angle
launchbb, newbb, score = stbb(bbarr, newbb, launchbb, score)

fbblast = []
for row in range(len(bbarr)):
    for col in range(len(bbarr[0])):
        if bbarr[row][col] != blank:
            fbblast.append(bbarr[row][col])
            if bbarr[row][col].rect.bottom > (winhgt - arrow.rect.height -
10):
                return score.total, 'lose'

if len(fbblast) < 1:
    return score.total, 'win'
gameclrlst = updtclrlst(bbarr)
random.shuffle(gameclrlst)
if launchbb == False:

    nxtbb = Bubble(gameclrlst[0])
    nxtbb.rect.right = winwidth - 5
    nxtbb.rect.bottom = winhgt - 5

```

```
nxtbb.draw()
```

```
if launchbb == True:
```

```
    covnxtbb()
```

```
arrow.update(dir)
```

```
arrow.draw()
```

```
setarrpos(bbarr)
```

```
drawbbary(bbarr)
```

```
score.draw()
```

```
if pygame.mixer.music.get_busy() == False:
```

```
    if track == len(musclist) - 1:
```

```
        track = 0
```

```
    else:
```

```
        track += 1
```

```
    pygame.mixer.music.load(musclist[track])
```

```
    pygame.mixer.music.play()
```

```
pygame.display.update()
```

```
fpsclock.tick(FPS)
```

```

def mkeblkbrd():
    array = []

    for row in range(aryhgt):
        col = []
        for i in range(arywdth):
            col.append(blank)
        array.append(col)

    return array

def setbb(array, gameclrlst):
    for row in range(bubblelyrs):
        for col in range(len(array[row])):
            random.shuffle(gameclrlst)
            newbb = Bubble(gameclrlst[0], row, col)
            array[row][col] = newbb

    setarrpos(array)

def setarrpos(array):
    for row in range(aryhgt):
        for col in range(len(array[row])):

```

```
if array[row][col] != blank:
    array[row][col].rect.x = (bubblewidth * col) + 5
    array[row][col].rect.y = (bubblewidth * row) + 5
```

```
for row in range(1, aryhgt, 2):
    for col in range(len(array[row])):
        if array[row][col] != blank:
            array[row][col].rect.x += bubblerad
```

```
for row in range(1, aryhgt):
    for col in range(len(array[row])):
        if array[row][col] != blank:
            array[row][col].rect.y -= (bubadjst * row)
```

```
delextrbb(array)
```

```
def delextrbb(array):
    for row in range(aryhgt):
        for col in range(len(array[row])):
            if array[row][col] != blank:
                if array[row][col].rect.right > winwidth:
                    array[row][col] = blank
```

```

def updtclrlist(bbarr):
    newColorList = []
    for row in range(len(bbarr)):
        for col in range(len(bbarr[0])):
            if bbarr[row][col] != blank:
                newColorList.append(bbarr[row][col].color)

    colorSet = set(newColorList)

    if len(colorSet) < 1:
        colorList = []
        colorList.append(white)
        return colorList
    else:

        return list(colorSet)


def chkfflotrs(bbarr):
    bubbleList = [col for col in range(len(bbarr[0]))
                   if bbarr[0][col] != blank]

    newbbList = []

    for i in range(len(bubbleList)):

```

```

        if i == 0:
            newbbList.append(bubbleList[i])
        elif bubbleList[i] > bubbleList[i - 1] + 1:
            newbbList.append(bubbleList[i])

cpyofbrd = copy.deepcopy(bbarr)

for row in range(len(bbarr)):
    for col in range(len(bbarr[0])):
        bbarr[row][col] = blank

for col in newbbList:
    popflotrs(bbarr, cpyofbrd, col)

def popflotrs(bbarr, cpyofbrd, col, row=0):
    if (row < 0 or row > (len(bbarr)-1)
        or col < 0 or col > (len(bbarr[0])-1)):
        return
    elif cpyofbrd[row][col] == blank:
        return
    elif bbarr[row][col] == cpyofbrd[row][col]:
        return
    bbarr[row][col] = cpyofbrd[row][col]
    if(row == 0):

```

```

    popflotrs(bbarr, cpyofbrd, col + 1, row)
    popflotrs(bbarr, cpyofbrd, col - 1, row)
    popflotrs(bbarr, cpyofbrd, col, row + 1)
    popflotrs(bbarr, cpyofbrd, col - 1, row + 1)
elif(row % 2 == 0):
    popflotrs(bbarr, cpyofbrd, col + 1, row)
    popflotrs(bbarr, cpyofbrd, col - 1, row)
    popflotrs(bbarr, cpyofbrd, col, row + 1)
    popflotrs(bbarr, cpyofbrd, col - 1, row + 1)
    popflotrs(bbarr, cpyofbrd, col, row - 1)
    popflotrs(bbarr, cpyofbrd, col - 1, row - 1)
else:
    popflotrs(bbarr, cpyofbrd, col + 1, row)
    popflotrs(bbarr, cpyofbrd, col - 1, row)
    popflotrs(bbarr, cpyofbrd, col, row + 1)
    popflotrs(bbarr, cpyofbrd, col + 1, row + 1)
    popflotrs(bbarr, cpyofbrd, col, row - 1)
    popflotrs(bbarr, cpyofbrd, col + 1, row - 1)

def stbb(bbarr, newbb, launchbb, score):
    dellst = []
    popSound = pygame.mixer.Sound('popcork.ogg')

    for row in range(len(bbarr)):

```



```

for col in range(len(bbarr[row])):

    if (bbarr[row][col] != blank and newbb != None):
        if (pygame.sprite.collide_rect(newbb, bbarr[row][col])) or
newbb.rect.top < 0:
            if newbb.rect.top < 0:
                newRow, newcol = addbbtotop(bbarr, newbb)

            elif newbb.rect.centery >= bbarr[row][col].rect.centery:

                if newbb.rect.centerx >= bbarr[row][col].rect.centerx:
                    if row == 0 or (row) % 2 == 0:
                        newRow = row + 1
                        newcol = col
                        if bbarr[newRow][newcol] != blank:
                            newRow = newRow - 1
                            bbarr[newRow][newcol] = copy.copy(newbb)
                            bbarr[newRow][newcol].row = newRow
                            bbarr[newRow][newcol].col = newcol

                    else:
                        newRow = row + 1
                        newcol = col + 1
                        if bbarr[newRow][newcol] != blank:
                            newRow = newRow - 1
                            bbarr[newRow][newcol] = copy.copy(newbb)

```

```

        bbarr[newRow][newcol].row = newRow
        bbarr[newRow][newcol].col = newcol

elif newbb.rect.centerx < bbarr[row][col].rect.centerx:
    if row == 0 or row % 2 == 0:
        newRow = row + 1
        newcol = col - 1
        if newcol < 0:
            newcol = 0
        if bbarr[newRow][newcol] != blank:
            newRow = newRow - 1
        bbarr[newRow][newcol] = copy.copy(newbb)
        bbarr[newRow][newcol].row = newRow
        bbarr[newRow][newcol].col = newcol
    else:
        newRow = row + 1
        newcol = col
        if bbarr[newRow][newcol] != blank:
            newRow = newRow - 1
        bbarr[newRow][newcol] = copy.copy(newbb)
        bbarr[newRow][newcol].row = newRow
        bbarr[newRow][newcol].col = newcol

elif(newbb.rect.centery < bbarr[row][col].rect.centery):
    if(newbb.rect.centerx >= bbarr[row][col].rect.centerx):

```

```

if(row == 0 or row % 2 == 0):
    newRow = row - 1
    newcol = col
    if(bbarr[newRow][newcol] != blank):
        newRow = newRow + 1
        bbarr[newRow][newcol] = copy.copy(newbb)
        bbarr[newRow][newcol].row = newRow
        bbarr[newRow][newcol].col = newcol
    else:
        newRow = row - 1
        newcol = col + 1
        if bbarr[newRow][newcol] != blank:
            newRow = newRow + 1
            bbarr[newRow][newcol] = copy.copy(newbb)
            bbarr[newRow][newcol].row = newRow
            bbarr[newRow][newcol].col = newcol

elif newbb.rect.centerx <= bbarr[row][col].rect.centerx:
    if(row == 0 or row % 2 == 0):
        newRow = row - 1
        newcol = col - 1
        if(bbarr[newRow][newcol] != blank):
            newRow = newRow + 1
            bbarr[newRow][newcol] = copy.copy(newbb)
            bbarr[newRow][newcol].row = newRow

```

```
bbarr[newRow][newcol].col = newcol
```

```
else:
```

```
    newRow = row - 1
```

```
    newcol = col
```

```
    if(bbarr[newRow][newcol] != blank):
```

```
        newRow = newRow + 1
```

```
    bbarr[newRow][newcol] = copy.copy(newbb)
```

```
    bbarr[newRow][newcol].row = newRow
```

```
    bbarr[newRow][newcol].col = newcol
```

```
popbb(bbarr, newRow, newcol, newbb.color, dellst)
```

```
if(len(dellst) >= 3):
```

```
    for pos in dellst:
```

```
        popSound.play()
```

```
        row = pos[0]
```

```
        col = pos[1]
```

```
        bbarr[row][col] = blank
```

```
    chkfflotrs(bbarr)
```

```
    score.update(dellst)
```

```
launchbb = False
```

```
newbb = None
```

```
return launchbb, newbb, score
```

```
def addbbtotop(bbarr, bubble):
```

```
    posx = bubble.rect.centerx
```

```
    leftSidex = posx-bubblerad
```

```
    coldiv = math.modf(float(leftSidex)/float(bubblewidth))
```

```
    col = int(coldiv[1])
```

```
    if (coldiv[0] < 0.5):
```

```
        bbarr[0][col] = copy.copy(bubble)
```

```
    else:
```

```
        col += 1
```

```
        bbarr[0][col] = copy.copy(bubble)
```

```
    row = 0
```

```
    return row, col
```

```
def popbb(bbarr, row, col, color, dellst):
```

```
    if(row < 0 or col < 0 or row > (len(bbarr)-1) or col >
        (len(bbarr[0])-1)):
```

```
        return
```

```
    elif (bbarr[row][col] == blank):
```

```
        return
```

```
    elif(bbarr[row][col].color != color):
```

```
        return
```

```

        for bubble in dellst:
            if(bbarr[bubble[0]][bubble[1]] == bbarr[row][col]):
                return
dellst.append((row, col))
if(row == 0):
    popbb(bbarr, row, col-1, color, dellst)
    popbb(bbarr, row, col+1, color, dellst)
    popbb(bbarr, row+1, col, color, dellst)
    popbb(bbarr, row+1, col-1, color, dellst)
elif(row % 2 == 0):
    popbb(bbarr, row + 1, col, color, dellst)
    popbb(bbarr, row + 1, col-1, color, dellst)
    popbb(bbarr, row - 1, col, color, dellst)
    popbb(bbarr, row - 1, col - 1, color, dellst)
    popbb(bbarr, row, col+1, color, dellst)
    popbb(bbarr, row, col-1, color, dellst)

else:
    popbb(bbarr, row - 1, col, color, dellst)
    popbb(bbarr, row - 1, col + 1, color, dellst)
    popbb(bbarr, row + 1, col, color, dellst)
    popbb(bbarr, row + 1, col + 1, color, dellst)
    popbb(bbarr, row, col + 1, color, dellst)
    popbb(bbarr, row, col - 1, color, dellst)

```

```

def drawbbary(array):
    for row in range(aryhgt):
        for col in range(len(array[row])):
            if(array[row][col] != blank):
                array[row][col].draw()


def makeDisplay():
    dispsurf = pygame.display.set_mode((winwidth, winhgt))
    disprect = dispsurf.get_rect()
    dispsurf.fill(bgcolor)
    dispsurf.convert()
    pygame.display.update()

    return dispsurf, disprect


def terminate():
    pygame.quit()
    sys.exit()


def covnxtbb():
    whiteRect = pygame.Rect(0, 0, bubblewidth, bubblewidth)

```

```
whiteRect.bottom = winhgt
whiteRect.right = winwidth
pygame.draw.rect(dispsurf, bgcolor, whiteRect)
```

```
def endScreen(score, winorlose):
    endFont = pygame.font.SysFont('merlin', 50)
    endMessage1 = endFont.render('You ' + winorlose + '! Hey Your Scored
' + str(
    score) + '. Press Enter to Play Again.', True, black, bgcolor)
    endMessage1Rect = endMessage1.get_rect()
    endMessage1Rect.center = disprect.center

    dispsurf.fill(bgcolor)
    dispsurf.blit(endMessage1, endMessage1Rect)
    pygame.display.update()
```

```
while True:
    for event in pygame.event.get():
        if(event.type == QUIT):
            terminate()
        elif(event.type == KEYUP):
            if(event.key == K_RETURN):
                return
            elif(event.key == K_ESCAPE):
                terminate()
```



```
if __name__ == '__main__':  
    main()
```

[OceanofPDF.com](http://OceanofPDF.com)

## Module 2 Project 11-20

### 11. Calculate Your Age!

<!--Remove the below lines and add yours -->

This script prints your age in three different ways :

1. Years
2. Months
3. Days

## Prerequisites

<!--Remove the below lines and add yours -->

You only need Python to run this script. You can visit [here] (<https://www.python.org/downloads/>) to download Python.

## How to run the script

<!--Remove the below lines and add yours -->

Running the script is really simple! Just open a terminal in the folder where your script is located and run the following command :

```
`python calculate.py`
```

## Sample use of the script

<!--Remove the below lines and add yours -->

```

\$ python calculate.py

input your name: XYZ

input your age: 33

XYZ's age is 33 years or 406 months or 12328 days

```

### **Source Code :**

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

```
import time
```

```
from calendar import isleap
```

```
# judge the leap year
```

```
def judge_leap_year(year):
```

```
    if isleap(year):
```

```
        return True
```

```
    else:
```

```
        return False
```

```
# returns the number of days in each month
```

```
def month_days(month, leap_year):
```

```
    if month in [1, 3, 5, 7, 8, 10, 12]:
```

```
        return 31
```

```
    elif month in [4, 6, 9, 11]:
```

```
    return 30
elif month == 2 and leap_year:
    return 29
elif month == 2 and (not leap_year):
    return 28
```

```
name = input("input your name: ")
age = input("input your age: ")
localtime = time.localtime(time.time())
```

```
year = int(age)
month = year * 12 + localtime.tm_mon
day = 0
```

```
begin_year = int(localtime.tm_year) - year
end_year = begin_year + year
```

```
    # calculate the days
    for y in range(begin_year, end_year):
        if (judge_leap_year(y)):
            day = day + 366
        else:
            day = day + 365
```

```
leap_year = judge_leap_year(localtime.tm_year)
for m in range(1, localtime.tm_mon):
    day = day + month_days(m, leap_year)

day = day + localtime.tm_mday
print("%s's age is %d years or " % (name, year), end="")
print("%d months or %d days" % (month, day))
```

[OceanofPDF.com](http://OceanofPDF.com)

## 12. Calculator GUI

<h1 align="center">Calculator (GUI)</h1>

It is GUI based calculator used to calculate any simple mathematical equation.

## Modules Used

- tkinter

## How it works

- It takes the mathematical equation by the User.
- It returns the result of the mathematical equation.

**Source code:**

```
import tkinter as tk
```

```

root = tk.Tk() # Main box window
root.title("Standard Calculator") # Title shown at the title bar
root.resizable(0, 0) # disabling the resizing of the window

# Creating an entry field:
e = tk.Entry(root,
              width=35,
              bg='#f0ffff',
              fg='black',
              borderwidth=5,
              justify='right',
              font='Calibri 15')
e.grid(row=0, column=0, columnspan=3, padx=12, pady=12)

def buttonClick(num): # function for clicking
    temp = e.get(
    ) # temporary variable to store the current input in the screen
    e.delete(0, tk.END) # clearing the screen from index 0 to END
    e.insert(0, temp + num) # inserting the incoming number input

def buttonClear(): # function for clearing
    e.delete(0, tk.END)

```

```

def buttonGet(
    oper
): # function for storing the first input and printing '+, -, /, *'
    global num1, math # global variable num1 and math to use in function
    buttonEqual()
    num1 = e.get() # getting first number
    math = oper # oper variable is the type of operation being performed
    e.insert(tk.END, math)
    try:
        num1 = float(num1) # converting the number to float type
    except ValueError: # in case there is a character other than numerals,
        clear the screen
        buttonClear()

```

```

def buttonEqual(): # function for printing the sum
    inp = e.get() # getting the inserted input
    num2 = float(inp[inp.index(math) + 1:]) # getting the second
    number
    e.delete(0, tk.END)
    if math == '+': # Addition
        e.insert(0, str(num1 + num2))
    elif math == '-': # Subtraction
        e.insert(0, str(num1 - num2))

```



```
elif math == 'x': # Multiplication
    e.insert(0, str(num1 * num2))
elif math == '/': # Division
    try:
        e.insert(0, str(num1 / num2))
    except ZeroDivisionError:
        # in case there is a zero in the denominator, answer is
        undefined
        e.insert(0, 'Undefined')
```

# Defining Buttons:

```
b1 = tk.Button(root,
                text='1',
                padx=40,
                pady=10,
                command=lambda: buttonClick('1'),
                font='Calibri 12')
b2 = tk.Button(root,
                text='2',
                padx=40,
                pady=10,
                command=lambda: buttonClick('2'),
                font='Calibri 12')
b3 = tk.Button(root,
                text='3',
```

```
        padx=40,
        pady=10,
        command=lambda: buttonClick('3'),
        font='Calibri 12')
b4 = tk.Button(root,
               text='4',
               padx=40,
               pady=10,
               command=lambda: buttonClick('4'),
               font='Calibri 12')
b5 = tk.Button(root,
               text='5',
               padx=40,
               pady=10,
               command=lambda: buttonClick('5'),
               font='Calibri 12')
b6 = tk.Button(root,
               text='6',
               padx=40,
               pady=10,
               command=lambda: buttonClick('6'),
               font='Calibri 12')
b7 = tk.Button(root,
               text='7',
               padx=40,
```

```
        pady=10,
        command=lambda: buttonClick('7'),
        font='Calibri 12')
b8 = tk.Button(root,
        text='8',
        padx=40,
        pady=10,
        command=lambda: buttonClick('8'),
        font='Calibri 12')
b9 = tk.Button(root,
        text='9',
        padx=40,
        pady=10,
        command=lambda: buttonClick('9'),
        font='Calibri 12')
b0 = tk.Button(root,
        text='0',
        padx=40,
        pady=10,
        command=lambda: buttonClick('0'),
        font='Calibri 12')
bdot = tk.Button(root,
        text='.',
        padx=41,
        pady=10,
```

```
command=lambda: buttonClick('.'),  
font='Calibri 12')
```

```
badd = tk.Button(root,  
    text='+',  
    padx=29,  
    pady=10,  
    command=lambda: buttonGet('+'),  
    font='Calibri 12')
```

```
bsub = tk.Button(root,  
    text='-',  
    padx=30,  
    pady=10,  
    command=lambda: buttonGet('-'),  
    font='Calibri 12')
```

```
bmul = tk.Button(root,  
    text='x',  
    padx=30,  
    pady=10,  
    command=lambda: buttonGet('x'),  
    font='Calibri 12')
```

```
bdiv = tk.Button(root,  
    text='/',  
    padx=30.5,  
    pady=10,
```

```
command=lambda: buttonGet('/'),  
font='Calibri 12')
```

```
bclear = tk.Button(root,  
    text='AC',  
    padx=20,  
    pady=10,  
    command=buttonClear,  
    font='Calibri 12')
```

```
bequal = tk.Button(root,  
    text='=',  
    padx=39,  
    pady=10,  
    command=buttonEqual,  
    font='Calibri 12')
```

# Putting the buttons on the screen:

```
b1.grid(row=3, column=0)  
b2.grid(row=3, column=1)  
b3.grid(row=3, column=2)  
badd.grid(row=3, column=3)
```

```
b4.grid(row=2, column=0)  
b5.grid(row=2, column=1)  
b6.grid(row=2, column=2)
```

```
bmul.grid(row=2, column=3)
```

```
b7.grid(row=1, column=0)
```

```
b8.grid(row=1, column=1)
```

```
b9.grid(row=1, column=2)
```

```
bdiv.grid(row=1, column=3)
```

```
b0.grid(row=4, column=0)
```

```
bdot.grid(row=4, column=1)
```

```
bequal.grid(row=4, column=2)
```

```
bsub.grid(row=4, column=3)
```

```
bclear.grid(row=0, column=3)
```

```
# Looping the window:
```

```
root.mainloop()
```

## 13. Calendar GUI

Using this code you will be able to create a Calendar GUI where calendar of a specific year appears .The year is specified by the user.

## Dependencies

You will use `Tkinter` and `calendar` python module.

### Source Code:

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

```
from tkinter import *
```

```
import calendar
```

```
def showCal():
```

```
    box = Tk()
```

```
    box.title("Calendar For The Year")
```

```
    box.geometry("550x600")
```

```
        find_year = int(year_field.get())
```

```
        first_label = Label(box, text='CALENDAR', bg='dark grey',
```

```
        font=("times", 28, 'bold'))
first_label.grid(row=1, column=1)

box.config(background="white")

cal_data = calendar.calendar(find_year)
cal_year = Label(box, text=cal_data, font="consolas 10 bold",
justify=LEFT)

cal_year.grid(row=2, column=1, padx=20,)

box.mainloop()

if __name__ == "__main__":

    gui = Tk()

    gui.config(background="misty rose")

    gui.title("CALENDAR")

    gui.geometry("250x250")

    cal = Label(gui, text="CALENDAR", bg="lavender",
        font=("Helvetica", 28, 'bold', 'underline'))
```



```
year = Label(gui, text="Enter Year", bg="peach puff", padx=10,  
pady=10)
```

```
year_field = Entry(gui)
```

```
Show = Button(gui, text="Show Calendar", fg="Black",  
bg="lavender", command=showCal)
```

```
Exit = Button(gui, text="CLOSE", bg="peach puff", command=exit)
```

```
cal.grid(row=1, column=1)
```

```
year.grid(row=3, column=1)
```

```
year_field.grid(row=4, column=1)
```

```
Show.grid(row=5, column=1)
```

```
Exit.grid(row=7, column=1)
```

```
gui.mainloop()
```

## 14. Capture Screenshot

captures screenshot at regular interval of time.

## Dependencies

```
```bash
pip install -r requirements.txt
```
```

## Usage

```
```bash
python screenshot.py          # takes screenshot at interval of 1 hour
python screenshot.py -t m -f 5    # takes 5 screenshots in 1 minute
python screenshot.py -p path_to_directory  # screenshots will be saved to
path_to_directory
```
```

**Requirement - PyAutoGUI==0.9.50**

### Source Code:

```
import os
import argparse
import pyautogui
import time

parser = argparse.ArgumentParser()

parser.add_argument("-p", "--path", help="absolute path to store
screenshot.", default=r"./images")

parser.add_argument("-t", "--type", help="h (in hour) or m (in
minutes) or s (in seconds)", default='h')

parser.add_argument("-f", "--frequency", help="frequency for taking
screenshot per h/m/s.", default=1, type=int)

args = parser.parse_args()

sec = 0.

if args.type == 'h':
    sec = 60 * 60 / args.frequency
elif args.type == 'm':
    sec = 60 / args.frequency

if sec < 1.:
    sec = 1.
```

```
if os.path.isdir(args.path) != True:
```

```
    os.mkdir(args.path)
```

```
try:
```

```
    while True:
```

```
        t = time.localtime()
```

```
        current_time = time.strftime("%H_%M_%S", t)
```

```
        file = current_time + ".jpg"
```

```
        image = pyautogui.screenshot(os.path.join(args.path, file))
```

```
        print(f"{file} saved successfully.\n")
```

```
        time.sleep(sec)
```

```
except KeyboardInterrupt:
```

```
    print("End of script by user interrupt")
```

[OceanofPDF.com](http://OceanofPDF.com)

## 15. Capture Video Frames

# In order to Capture Video Frames

##### Execute

```
`python capture_video_frames.py <video_file>`
```

# In order to get live video stream from webcam

#### EXecute

```
python capture_video_from_webcam.py
```

# In order to stream video from webcam to AWS Kinesis follow the below instructions

Amazon Kinesis Video Streams CPP Producer, GStreamer Plugin and JNI

<h4 align="center"> Amazon Kinesis Video Streams | Secure Video Ingestion for Analysis & Storage </h4>

<p align="center">

<a href="https://travis-ci.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp">  </a>

<a href="https://codecov.io/gh/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp">  </a>

</p>

<p align="center">

<a href="#key-features">Key Features</a> •

<a href="#build">Build</a> •

<a href="#run">Run</a> •

<a href="#documentation">Documentation</a> •

<a href="#related">Related</a> •

<a href="#license">License</a>

</p>

## ## Key Features

- \* C++ SDK
- \* GStreamer Plugin (kvssink)
- \* JNI

Amazon Kinesis Video Streams Producer SDK for C/C++ makes it easy to build an on-device application that securely connects to a video stream, and reliably publishes video and other media data to Kinesis Video Streams. It takes care of all the underlying tasks required to package the frames and fragments generated by the device's media pipeline. The SDK also hand

les stream creation, token rotation for secure and uninterrupted streaming, processing acknowledgements returned by Kinesis Video Streams, and other tasks.

## ## Build

### ### Download

To download run the following command:

```
`git clone https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp.git`
```

Note: You will also need to install `pkg-config`, `CMake`, `m4` and a build environment. If you are building the GStreamer plugin you will also need GStreamer and GStreamer (Development Libraries).

Refer to the [FAQ](#FAQ) for platform specific instructions.

### ### Configure

Prepare a build directory in the newly checked out repository:

```
```\n\nmkdir -p amazon-kinesis-video-streams-producer-sdk-cpp/build\n\n cd amazon-kinesis-video-streams-producer-sdk-cpp/build\n\n```\n
```

If you are building on Windows you need to generate `NMake Makefiles`, you should run `cmake .. -G "NMake Makefiles"`

GStreamer and JNI is NOT built by default, if you wish to build both you MUST execute `cmake .. -DBUILD\_GSTREAMER\_PLUGIN=ON -DBUILD\_JNI=TRUE`

By default we download all the libraries from GitHub and build them locally, so should require nothing to be installed ahead of time.

If you do wish to link to existing libraries you can do ``cmake .. - DBUILD_DEPENDENCIES=OFF``

Libraries needed to build producer are: Curl, Openssl and Log4cplus. If you want to build the gstreamer plugin you will need to have gstreamer in your system.

On Mac OS you can get the libraries using homebrew

...

```
$ brew install pkg-config openssl cmake gstreamer gst-plugins-base gst-plugins-good gst-plugins-bad gst-plugins-ugly log4cplus gst-libav
```

...

On Ubuntu and Raspberry Pi OS you can get the libraries by running

...

```
$ sudo apt-get install libssl-dev libcurl4-openssl-dev liblog4cplus-dev libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-bad gstreamer1.0-plugins-good gstreamer1.0-plugins-ugly gstreamer1.0-tools
```

...

#### #### Cross-Compilation

If you wish to cross-compile ``CC`` and ``CXX`` are respected when building the library and all its dependencies. See our `[.travis.yml](.travis.yml)` for an example of this. Every commit is cross compiled to ensure that it continues to work.

#### #### CMake Arguments



You can pass the following options to `cmake ..`.

- \* ``-DBUILD_GSTREAMER_PLUGIN`` -- Build kvssink GStreamer plugin
- \* ``-DBUILD_JNI`` -- Build C++ wrapper for JNI to expose the functionality to Java/Android
- \* ``-DBUILD_DEPENDENCIES`` -- Build depending libraries from source
- \* ``-DBUILD_TEST=TRUE`` -- Build unit/integration tests, may be useful for confirm support for your device. ``.tst/producerTest``
- \* ``-DCODE_COVERAGE`` -- Enable coverage reporting
- \* ``-DCOMPILER_WARNINGS`` -- Enable all compiler warnings
- \* ``-DADDRESS_SANITIZER`` -- Build with AddressSanitizer
- \* ``-DMEMORY_SANITIZER`` -- Build with MemorySanitizer
- \* ``-DTHREAD_SANITIZER`` -- Build with ThreadSanitizer
- \* ``-DUNDEFINED_BEHAVIOR_SANITIZER`` Build with UndefinedBehaviorSanitizer
- \* ``-DALIGNED_MEMORY_MODEL`` Build for aligned memory model only devices. Default is OFF.

#### To Include JNI

JNI examples are NOT built by default. If you wish to build JNI you MUST add `-DBUILD_JNI=TRUE` when running cmake:

...

`cmake -DBUILD_JNI=TRUE`

...

#### #### To Include Building GStreamer Sample Programs

The GStreamer plugin and samples are NOT built by default. If you wish to build them you MUST add `-DBUILD_GSTREAMER_PLUGIN=TRUE` when running `cmake`:

...

```
cmake -DBUILD_GSTREAMER_PLUGIN=TRUE
```

...

#### ### Compiling

After running `cmake`, in the same build directory run `make`:

...

```
make
```

...

On Windows you should run ``nmake`` instead of ``make``

In your build directory you will now have shared objects for all the targets you have selected

#### ## Run

#### #### GStreamer Plugin (kvssink)

#### #### Loading Element

The GStreamer plugin is located in your `build` directory.

To load this plugin set the following environment variables. This should be run from the root of the repo, NOT the `build` directory.

...

```
export GST_PLUGIN_PATH=`pwd`/build
```

```
export LD_LIBRARY_PATH=`pwd`/open-source/local/lib
```

...

The equivalent for Windows is

...

```
set GST_PLUGIN_PATH=%CD%\build
```

```
set PATH=%PATH%;%CD%\open-source\local\bin;%CD%\open-  
source\local\lib
```

...

Now if you execute `gst-inspect-1.0 kvssink` you should get information on the plugin like

```text

Factory Details:

|      |                    |
|------|--------------------|
| Rank | primary + 10 (266) |
|------|--------------------|

|           |          |
|-----------|----------|
| Long-name | KVS Sink |
|-----------|----------|

|             |  |
|-------------|--|
| Klass       | Sink/Video/Network                         |
| Description | GStreamer AWS KVS plugin                   |
| Author      | AWS KVS <kinesis-video-support@amazon.com> |

#### Plugin Details:

|                |   |
|----------------|---|
| Name           | kvssink   |
| Description    | GStreamer AWS KVS plugin  |
| Filename       | /Users/seaduboi/workspaces/amazon-kinesis-video-streams-producer-sdk-cpp/build/libgstkvssink.so |
| Version        | 1.0   |
| License        | Proprietary   |
| Source module  | kvssinkpackage  |
| Binary package | GStreamer   |
| Origin URL     | <a href="http://gstreamer.net">http://gstreamer.net</a>   |

...

If the build failed, or `GST\_PLUGIN\_PATH` is not properly set you will get output like

```
```text
```

```
No such element or plugin 'kvssink'
```

```
```
```

#### #### Using Element

The kvssink element has the following required parameters:

- \* ``stream-name`` -- The name of the destination Kinesis video stream.
- \* ``storage-size`` -- The storage size of the device in megabytes. For information about configuring device storage, see `StorageInfo`.
- \* ``access-key`` -- The AWS access key that is used to access Kinesis Video Streams. You must provide either this parameter or `credential-path`.
- \* ``secret-key`` -- The AWS secret key that is used to access Kinesis Video Streams. You must provide either this parameter or `credential-path`.
- \* ``credential-path`` -- A path to a file containing your credentials for accessing Kinesis Video Streams. For example credential files, see `Sample Static Credential` and `Sample Rotating Credential`. For more information on rotating credentials, see `Managing Access Keys for IAM Users`. You must provide either this parameter or `access-key` and `secret-key`.

For examples of common use cases you can look at [Example: Kinesis Video Streams Producer SDK GStreamer Plugin]  
(<https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/examples-gstreamer-plugin.html>)

## ## Dockerscripts

- \* The sample docker scripts for RTSP plugin, raspberry pi and linux can be found in the [Kinesis demos repository](<https://github.com/aws-samples/amazon-kinesis-video-streams-demos/tree/master/producer-cpp>).

## ## DEBUG

- \* If you are successfully streaming but run into issue with playback. You can do ``export``

KVS\_DEBUG\_DUMP\_DATA\_FILE\_DIR=/path/to/directory` before streaming. Producer will then dump MKV files into that path. The file is exactly what KVS will receive. You can use [MKVToolNIX] (<https://mkvtoolnix.download/index.html>) to check that everything looks correct. You can also try to play the MKV file in compatible players.

## ## FAQ

\* Is CPP-SDK and GStreamer supported on Mac/Windows/Linux (Supported Platforms)

Yes! We have FAQs and platform specific instructions for [Windows] (docs/windows.md), [MacOS](docs/macos.md) and [Linux] (docs/linux.md)

## ## Development

The repository is using master branch as the aggregation and all of the feature development is done in appropriate feature branches. The PRs (Pull Requests) are cut on a feature branch and once approved with all the checks passed they can be merged by a click of a button on the PR tool. The master branch should always be build-able and all the tests should be passing. We are welcoming any contribution to the code base.

## ### Release

The repository is under active development and even with incremental unit test coverage where some of the tests are actually full integration tests, we require more rigorous internal testing in order to 'cut' release versions. The release is cut against a particular commit that gets approved. The general philosophy is to cut a release when a set of commits contribute to a self-containing feature or when we add major internal functionality improvements.

### ### Versioning

We deploy 3 digit version strings in a form of 'Major.Minor.Revision' scheme.

- \* Major version update - Major functionality changes. Might not have direct backward compatibility. For example, multiple public API parameter changes.
- \* Minor version update - Additional features. Major bug fixes. Might have some minor backward compatibility issues. For example, an extra parameter on a callback function.
- \* Revision version update - Minor features. Bug fixes. Full backward compatibility. For example, an extra fields added to the public structures with version bump.

### ## Related

- \* [What Is Amazon Kinesis Video Streams]  
(<https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/what-is-kinesis-video.html>)
- \* [C SDK](<https://github.com/aws-labs/amazon-kinesis-video-streams-producer-c>)
- \* [Example: Kinesis Video Streams Producer SDK GStreamer Plugin]  
(<https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/examples-gstreamer-plugin.html>)

### ## License

This library is licensed under the Apache 2.0 License.

**Requirement - opencv-python==4.3.0.36**

**Source Code:**

```
import os
import shutil
import sys
import cv2

class FrameCapture:
    """
        Class definition to capture frames
    """
    def __init__(self, file_path):
        """
            initializing directory where the captured frames will be stored.
            Also truncating the directory where captured frames are
            stored, if exists.
        """
        self.directory = "captured_frames"
        self.file_path = file_path
```



```

        if os.path.exists(self.directory):
            shutil.rmtree(self.directory)
        os.mkdir(self.directory)

def capture_frames(self):
    """
    This method captures the frames from the video file provided.
    This program makes use of openCV library
    """
    cv2_object = cv2.VideoCapture(self.file_path)

    frame_number = 0
    frame_found = 1
    while frame_found:
        frame_found, image = cv2_object.read()
        capture = f'{self.directory}/frame{frame_number}.jpg'
        cv2.imwrite(capture, image)
        frame_number += 1

if __name__ == '__main__':
    file_path = sys.argv[1]
    fc = FrameCapture(file_path)
    fc.capture_frames()

```

## 16. Check Website Connectivity

This directory contains a simple tool to check connectivity to a number of web sites.

The input file `websites.txt` should contain web site URLs, one per line.

The output file `website\_status.csv` contains a two-column report with

the URL of each checked site and its status.

The script simply checks whether the web server returns a 200 status code.

The output file will be overwritten each time you run the tool.

## Prerequisites

This project uses the third-party library

[requests](<https://requests.readthedocs.io/>)

as well as the `csv` module from the Python standard library.

## How to run the Script

To run this script, type

```
...
```

```
python check_connectivity.py
```

```
...
```

in the directory where you have checked out these files.

(If you have an IDE which lets you run Python files,

and prefer to use that instead,

make sure you configure it to set the working directory to

the one which contains the input file.)

## Development ideas

The CSV should perhaps contain a date stamp, too.

Perhaps add the `logging` library and optionally print progress information.

**Source Code:**

```
import csv
```

```
import requests
```

```
status_dict = {"Website": "Status"}
```

```
def main():
```

```
    with open("websites.txt", "r") as fr:
```

```
        for line in fr:
```

```
            website = line.strip()
```

```
            status = requests.get(website).status_code
```

```
            status_dict[website] = "working" if status == 200 \
                else "not working"
```

```
    # print(status_dict)
```

```
    with open("website_status.csv", "w", newline="") as fw:
```

```
        csv_writers = csv.writer(fw)
```

```
        for key in status_dict.keys():
```

```
            csv_writers.writerow([key, status_dict[key]])
```

```
if __name__ == "__main__":
```

main()

[OceanofPDF.com](http://OceanofPDF.com)

## 17. Chrome Automation

```
import webbrowser as wb
```

```
def webauto():  
    chrome_path = 'C:/Program Files  
(x86)/Google/Chrome/Application/chrome.exe %s'  
    URLS = ("stackoverflow.com", "edcredibly.com", "gmail.com",  
            "google.co.in", "youtube.com")  
  
    for url in URLS:  
        print("Opening: " + url)  
        wb.get(chrome_path).open(url)  
  
webauto()
```

## 18. Cli Proxy Tester

### ## Usage

This script tests proxies by querying (GET request) a testing website that returns the IP of the client. If the returned IP matches the IP of the proxy, we consider the proxy to be good.

### ### Testing Single Proxies

```
`python cli.py single http://1.1.1.1`
```

This will test the HTTP proxy 1.1.1.1 against the default testing website [iptest.ingokleiber.de](http://iptest.ingokleiber.de).

You can run your own testing service using the PHP script in `/ipinfo`. This service should be offered both via HTTP and HTTPS.

```
`python cli.py single http://1.1.1.1 --iptest iptest.yourdomain.com`
```

### ### (Re)Testing a CSV File

```
`python cli.py csv-file proxies.csv`
```

This will (re)test all proxies in the given file.

### ### Adding and Testing Proxies From a Text File

```
`python cli.py add-from-txt-file proxy_candidates.txt`
```

This will add and test each proxy (one per line) in  
`proxy\_candidates.txt`.

### **Requirements:**

**click==7.1.2**

**proxytest==0.5.4**

**pandas==1.0.5**

### **Cli Source Code:**

```
import re
```

```
import click
```

```
from proxytest import add_from_text_file
```

```
from proxytest import test_csv_file
```

```
from proxytest import test_single_proxy
```

```
def validate_proxy(ctx, param, value):
```

```
    """Validate proxy input. The RegEx crudely matches both IPv4 and  
    URLs."""
```

```
    validator = re.compile(r'(https|http|socks4|socks5):\\V'
```

```
        r'((?:[0-9]{1,3}\\.){3}[0-9]{1,3}(:[0-9]{2,5})?'
```

```
        r'|([\\da-z\\.-]+)\\.([a-z\\.]{{2,6}})([\\w \\.-]*\\V?)')'
```



```
if not validator.match(value):
    raise click.BadParameter('Please provide a proxy in the format'
                              'type://address (e.g., https://42.42.42.42)')
else:
    return value
```

```
@click.group()
```

```
def cli():
```

```
    pass
```

```
@cli.command()
```

```
@click.argument('proxy', callback=validate_proxy)
```

```
@click.option('--iptest', default='iptest.ingokleiber.de',
              help='iptest address')
```

```
@click.option('--csv', default='proxies.csv', help='CSV path')
```

```
def single(proxy, iptest, csv):
```

```
    test_single_proxy(proxy, iptest, csv)
```

```
@cli.command()
```

```
@click.argument('csv')
```

```
@click.option('--iptest', default='iptest.ingokleiber.de',
              help='iptest address')
```

```

def csv_file(ipctest, csv):
    test_csv_file(ipctest, csv)

@cli.command()
@click.argument('txt')
@click.option('--ipctest', default='ipctest.ingokleiber.de',
              help='ipctest address')
@click.option('--csv', default='proxies.csv', help='CSV path')
def add_from_txt_file(ipctest, txt, csv):
    add_from_text_file(ipctest, txt, csv)

if __name__ == '__main__':
    cli()

```

### **Proxy Test Source Code:**

```

import logging
from json.decoder import JSONDecodeError
from pathlib import Path

import pandas as pd
import requests
from requests.exceptions import ProxyError

```

```
logging.basicConfig(level=logging.INFO)
```

```
def add_proxies_to_file(csv_path: str, proxies: list):
```

```
    """This function will add one or multiple proxies to the CSV file."""
```

```
    if not csv_path.exists():
```

```
        pr_file: pd.DataFrame = pd.DataFrame(
            columns=['proxy_type', 'proxy_address', 'proxy_status'])
        logging.info('New CSV file will be created')
```

```
    else:
```

```
        pr_file: pd.DataFrame = pd.read_csv(csv_path)
        logging.info('Existing CSV file has been loaded')
```

```
    for proxy in proxies:
```

```
        if len(pr_file) == 0:
```

```
            # First proxy in the file
```

```
            pr_file = pr_file.append(proxy, ignore_index=True)
```

```
        else:
```

```
            if len(pr_file.loc[(pr_file['proxy_type'] == proxy['proxy_type']) &
                                (pr_file['proxy_address'] == proxy['proxy_address'])]) >
```

```
0:
```

```
                # Proxy is already in the file
```

```
                pr_file.loc[(pr_file['proxy_type'] == proxy['proxy_type']) &
                             (pr_file['proxy_address'] == proxy['proxy_address']),
```

```
        ['proxy_status']] = proxy['proxy_status']
    else:
        # Proxy is not yet in the file
        pr_file = pr_file.append(proxy, ignore_index=True)

pr_file = pr_file.drop_duplicates()
pr_file.to_csv(csv_path, index=False)
logging.info('CSV file has been written')
```

```
def test_proxy(proxy_type: str, proxy_address: str, iptest: str):
```

```
    """This function takes a proxy (type, address)
    and tests it against a given iptest address."""
```

```
    logging.info(f'Testing proxy: {proxy_address}')
```

```
    try:
```

```
        proxies = {proxy_type: proxy_address}
```

```
        proxy_status: str = "
```

```
    if proxy_type == 'https':
```

```
        r = requests.get(f'https://{iptest}', proxies=proxies)
```

```
    else:
```

```
        r = requests.get(f'http://{iptest}', proxies=proxies)
```

```

try:
    json_response: dict = r.json()

    if json_response["ip"] in proxy_address:
        proxy_status = 'Proxy functional'
    else:
        logging.warning(f'Proxy "{proxy_address}"'
                        f'returned {json_response}')
        proxy_status = 'Proxy not functional'
except JSONDecodeError:
    proxy_status = 'Invalid response'
except ProxyError:
    proxy_status = 'Proxy error'

logging.info(f'Proxy {proxy_address}: {proxy_status}')
return {'proxy_type': proxy_type,
        'proxy_address': proxy_address,
        'proxy_status': proxy_status}

```

```

def test_single_proxy(proxy: str, iptest: str, csv_path: str):
    """This function tests an individual proxy and adds it to the CSV file."""
    proxy_type, proxy_address = proxy.split('/:/')
    result: dict = test_proxy(proxy_type, proxy_address, iptest)

```

```

    add_proxies_to_file(Path(csv_path), [result])
def test_csv_file(iptest: str, csv_path: str):
    """This function (re)tests every proxy in a given CSV file."""

    csv_path: Path = Path(csv_path)

    if csv_path.exists():
        pr_file: pd.DataFrame = pd.read_csv(csv_path)
    else:
        raise FileNotFoundError

    proxies: list = []

    for index, proxy in pr_file.iterrows():
        proxies.append(test_proxy(proxy['proxy_type'],
                                   proxy['proxy_address'],
                                   iptest))

    add_proxies_to_file(csv_path, proxies)


def add_from_text_file(iptest: str, text_path: str, csv_path: str):
    """ This function adds a list of proxies
        from a text file (line by line)."""
    text_path: Path = Path(text_path)

```

```
if text_path.exists():
    proxies: list = text_path.read_text().splitlines()

    for proxy in proxies:
        """We will treat each proxy as a single proxy
        and leverage the existing function"""
        test_single_proxy(proxy, iptest, csv_path)
else:
    raise FileNotFoundError
```

[OceanofPDF.com](http://OceanofPDF.com)

## 19. Simple CLI Todo App

Simple Todo app with command line interface. Supports adding, deleting, and viewing task entries.

## Dependencies

Requires Python 3 and Click

Install Click: `pip install click`

## How to use

### Running

either run it from your code editor or Ide or type `python todo.py [command]` in your command line.

(insted of [command] add desired command u want)

### Commands

| Command | Description |
|---------|-------------|
|---------|-------------|

|       |       |
|-------|-------|
| ----- | ----- |
|-------|-------|

|       |  |
|-------|--|
| `add` | Adds a task. Prompts user for task text. |
|-------|--|

|        |   |
|--------|---|
| `done` | Deletes a task. Prompts user for task id. |
|--------|---|

|         |                              |
|---------|------------------------------|
| `tasks` | Displays all inputted tasks. |
|---------|------------------------------|



## Source Code:

```
import click

@click.group()
@click.pass_context
def todo(ctx):
    """Simple CLI Todo App"""
    ctx.ensure_object(dict)
    #Open todo.txt – first line contains latest ID, rest contain tasks and
    IDs
    with open('./todo.txt') as f:
        content = f.readlines()
    #Transfer data from todo.txt to the context
    ctx.obj['LATEST'] = int(content[:1][0])
    ctx.obj['TASKS'] = {en.split('`')[0]:en.split('`')[1][:-1] for en in
content[1:]}

@todo.command()
@click.pass_context
def tasks(ctx):
    """Display tasks"""
    if ctx.obj['TASKS']:
click.echo('YOUR TASKS\n*****')
    #Iterate through all the tasks stored in the context
    for i, task in ctx.obj['TASKS'].items():
```

```

        click.echo('• ' + task + ' (ID: ' + i + ')')
    click.echo("")
else:
    click.echo('No tasks yet! Use ADD to add one.\n')

```

```

@todo.command()
@click.pass_context
@click.option('-add', '--add_task', prompt='Enter task to add')
def add(ctx, add_task):
    "Add a task"
    if add_task:
        #Add task to list in context
        ctx.obj['TASKS'][ctx.obj['LATEST']] = add_task
        click.echo('Added task "' + add_task + '" with ID ' +
str(ctx.obj['LATEST']))
        #Open todo.txt and write current index and tasks with IDs (separated
by " `` ")
        curr_ind = [str(ctx.obj['LATEST']) + 1]
        tasks = [str(i) + '``' + t for (i, t) in ctx.obj['TASKS'].items()]
        with open('./todo.txt', 'w') as f:
            f.writelines(['%s\n' % en for en in curr_ind + tasks])

@todo.command()
@click.pass_context
@click.option('-fin', '--fin_taskid', prompt='Enter ID of task to finish',
type=int)

```

```

def done(ctx, fin_taskid):
    """Delete a task by ID"""
    #Find task with associated ID
    if str(fin_taskid) in ctx.obj['TASKS'].keys():
        task = ctx.obj['TASKS'][str(fin_taskid)]
        #Delete task from task list in context
        del ctx.obj['TASKS'][str(fin_taskid)]
        click.echo('Finished and removed task "' + task + '" with id ' +
str(fin_taskid))

        #Open todo.txt and write current index and tasks with IDs
        (separated by " `` ")
        if ctx.obj['TASKS']:
            curr_ind = [str(ctx.obj['LATEST'] + 1)]
            tasks = [str(i) + '``' + t for (i, t) in ctx.obj['TASKS'].items()]
            with open('./todo.txt', 'w') as f:
                f.writelines(['%s\n' % en for en in curr_ind + tasks])
        else:
            #Resets ID tracker to 0 if list is empty
            with open('./todo.txt', 'w') as f:
                f.writelines([str(0) + '\n'])
    else:
        click.echo('Error: no task with id ' + str(fin_taskid))

if __name__ == '__main__':
    todo()

```

[OceanofPDF.com](http://OceanofPDF.com)

## 20. PDF to TXT converter

This script takes in a .pdf file and outputs a .txt file

### ### Requirements

- Python
- PyPDF2

### ### Steps

In this program you have to provide the path for the pdf file that you want to convert into text and you may also provide the path where you want your output text file to be stored.

By default the output files created will be stored in temp folder in the same directory.

## Source Code :

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

```
import PyPDF2
```

```
import os
```

```
if(os.path.isdir("temp") == False):
```

```
    os.mkdir("temp")
```

```
txtpath = ""
```

```
pdfpath = ""
```

```
pdfpath = input("Enter the name of your pdf file - please use backslash  
when typing in directory path: ") #Provide the path for your pdf here
```

```
txtpath = input("Enter the name of your txt file - please use backslash when  
typing in directory path: ") #Provide the path for the output text file
```

```
BASEDIR = os.path.realpath("temp") # This is the sample base  
directory where all your text files will be stored if you do not give a  
specific path
```

```
print(BASEDIR)
```

```
if(len(txtpath) == 0):
```

```
txtpath =  
os.path.join(BASEDIR,os.path.basename(os.path.normpath(pdfpath)).  
replace(".pdf", "").+".txt")  
pdfobj = open(pdfpath, 'rb')  
  
pdfread = PyPDF2.PdfFileReader(pdfobj)  
  
x = pdfread.numPages  
for i in range(x):  
    pageObj = pdfread.getPage(i)  
    with open(txtpath, 'a+') as f:  
        f.write((pageObj.extractText()))  
  
    print(pageObj.extractText()) #This just provides the overview of  
    what is being added to your output, you can remove it if  
    want  
  
pdfobj.close()
```

[OceanofPDF.com](http://OceanofPDF.com)

## Module 3 Projects 21-30

[OceanofPDF.com](http://OceanofPDF.com)



## 21. Convert an Image to PDF

<!--Remove the below lines and add yours -->

The Python script enables the user to convert Images into PDF files. However, you must note that the script can only work well for JPG file formats. You can use the converter for revamping JPG images into PDF format.

### Requirements

<!--Remove the below lines and add yours -->

**\*\*img2pdf module\*\***

The `img2pdf` is an external Python module which enables you to convert a JPG image into a PDF.

```
pip install img2pdf
```

### How to run the script

<!--Remove the below lines and add yours -->

### Using Terminal

- Add the image in the JPG format with name as 'input' in this folder.
  - Run converter\_terminal.py script
  - Output PDF file will be generated in this folder

**Source Code:**

```
import sys
import img2pdf
import os

filepath = sys.argv[1]
if os.path.isdir(filepath):
    with open("output.pdf", "wb") as f:
        imgs = []
        for fname in os.listdir(filepath):
            if not fname.endswith(".jpg"):
                continue
            path = os.path.join(filepath, fname)
            if os.path.isdir(path):
                continue
            imgs.append(path)
        f.write(img2pdf.convert(imgs))
elif os.path.isfile(filepath):
    if filepath.endswith(".jpg"):
        with open("output.pdf", "wb") as f:
            f.write(img2pdf.convert(filepath))
else:
    print("please input file or dir")
```

## 22. Dictionary to Python Object

<!--Remove the below lines and add yours -->

A Class in python to convert dictionary to a object

### Prerequisites

<!--Remove the below lines and add yours -->

None, only a running Python installation is required.

### How to run the script

<!--Remove the below lines and add yours -->

- Add the `class obj` in your code.
- Modify the code according to your need or use it directly:  
`ob = obj({'a':1, 'b': 2, 'c':3})`

### Source Code:

```
class obj(object):  
    def __init__(self, d):
```

```
for x, y in d.items():  
    setattr(self, x, obj(y) if isinstance(y, dict) else y)  
data = {'a':5,'b':7,'c':{'d':8}}  
ob = obj(data)
```

[OceanofPDF.com](http://OceanofPDF.com)

## 23. Convert Images

# Convert Image Format

These scripts can change format of images from PNG to JPG and JPG to PNG

### Prerequisites

Required Modules

- PIL==1.1.6

To install:

```

\$ pip install -r requirements.txt

```

### How to run the script

- Dynamic Change

Copy the script `convertDynamic.py` into the directory where images

are (PNG and/or JPG). And run:

``` bash

\$ python convertDynamic.py

```

This will convert all JPG images to PNG and PNG images to JPG

in the present directory tree recursively

(i.e. will change format in images inside sub-directories too.)

- JPG to PNG (single image)

1. Copy the JPG image to the directory where `JPGtoPNG.py` exists
2. Replace file name `naruto\_first.jpg` inside `JPGtoPNG.py`  
(line 3) to input file name (JPG).
3. Replace file name `naruto.png` inside `JPGtoPNG.py` (line 4) to  
output file name (PNG).
4. Run following command:

```
...
```

```
$ python JPGtoPNG.py
```

```
...
```

- PNG to JPG (single image)

1. Copy the PNG image in directory where `PNGtoJPG.py` exists
2. Replace file name `naruto\_first.png` inside `PNGtoJPG.py`  
(line 3) to input file name (PNG).
3. Replace file name `naruto.jpg` inside `PNGtoJPG.py`  
(line 4) to output file name (JPG).
4. Run following command:

```
...
```

```
$ python PNGtoJPG.py
```

```
...
```

**convertDynamic Source code:**

```
from PIL import Image
```

```
import sys
```

```
import os

try:
    im = None
    for root, dirs, files in os.walk("."):
        for filename in files:
            if filename.endswith('.jpg'):
                im = Image.open(filename).convert("RGB")
                im.save(filename.replace('jpg', 'png'), "png")
            elif filename.endswith('.png'):
                im = Image.open(filename).convert("RGB")
                im.save(filename.replace('png', 'jpg'), "jpeg")
            else:
                print('dont have image to convert')
except IOError:
    print('directory empty!')
    sys.exit()
```

### **JPGtoPNG Source Code:**

```
from PIL import Image

im = Image.open("naruto_first.jpg").convert("RGB")
im.save("naruto.png", "png")
```

## **PNGtoJPG Source Code:**

```
from PIL import Image

im = Image.open("naruto_first.png").convert("RGB")
im.save("naruto.jpg", "jpeg")
```

[OceanofPDF.com](http://OceanofPDF.com)



## 24. CONVERT\_JPEG\_to\_PNG

This project contains a simple python script to change file extension from .jpeg to .png

## Requirements

Pillow module

`pip install pillow`

## Two methods:

I accomplished this task in two ways

### Using Terminal

- Add the image in jpeg format with name as 'input' in this folder.
- Run converter\_terminal.py script
- output image will be generated in this folder

### Using GUI

Just run the converter\_GUI.py script and pick any jpeg image from any location and then press 'Convert Jpeg to Png'

### **Converter Terminal Source Code:**

```
from PIL import Image
im1 = Image.open('input.jpeg') # takes input image from present folder
im1.save('output.png')        # output image is generated the folder
```

### **Converter GUI Source Code:**

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image
root = tk.Tk() # Tkinter window initialized
root.title('Converter') # Title of the window
canvas1 = tk.Canvas(root, width=300, height=250, bg='orange',
relief='raised')
canvas1.pack()
label1 = tk.Label(root, text='File Converter', bg='lightsteelblue2') # giving
a title to the screen
label1.config(font=('helvetica', 20))
canvas1.create_window(150, 60, window=label1)
im1 = None # variable to store path of image
```

```
def getJPG():
```

```
    """Function to get image location and open it with pillow"""
```

```

global im1
import_file_path = filedialog.askopenfilename()
im1 = Image.open(import_file_path)


font = ('helvetica', 12, 'bold')
bg = 'royalblue'
fg = 'white'

browseButton_JPG = tk.Button(text="    Import JPEG File    ",
command=getJPG, bg=bg, fg=fg, font=font) # Browse button
canvas1.create_window(150, 130, window=browseButton_JPG)


def convertToPNG():
    """Function to change file extension to png and save it to User's preferred
location """
    global im1
    if im1 is None:
        tk.messagebox.showerror("Error", "No File selected")
    else:
        export_file_path =
filedialog.asksaveasfilename(defaultextension='.png')
        im1.save(export_file_path)


saveAsButton_PNG = tk.Button(text='Convert JPEG to PNG',
command=convertToPNG, bg=bg, fg=fg, font=font) # Convert button

```

```
canvas1.create_window(150, 180, window=saveAsButton_PNG)  
root.mainloop()
```

[OceanofPDF.com](https://oceanofpdf.com)

## 25. Convert a json file into a csv

This script take a json file as input and generate a csv file in output.

### Prerequisites modules

\* json

\* Run `pip install json` to install required external modules.

### How to run the script

- Execute `python3 converter.py`

### Source code:

```
import json

if __name__ == '__main__':
    try:
        with open('input.json', 'r') as f:
            data = json.loads(f.read())

            output = ','.join([*data[0]])
            for obj in data:
                output += f'\n{obj["Name"]},{obj["age"]},{obj["birthyear"]}'

        with open('output.csv', 'w') as f:
            f.write(output)
    except Exception as ex:
        print(f'Error: {str(ex)}')
```

[OceanofPDF.com](http://OceanofPDF.com)

## 26. Convert Numbers To Words

<!--Remove the below lines and add yours -->

Convert a number to the written word form

### Prerequisites

<!--Remove the below lines and add yours -->

None

### How to run the script

<!--Remove the below lines and add yours -->

Execute `python3 converter.py`

## Screenshot/GIF showing the sample use of the script

<!--Remove the below lines and add yours -->

![Screenshot of the converter.py file](Screenshot.png)

### Source Code:

```
one_digit_words = {  
    '0': ["zero"],
```

```
'1': ["one"],
'2': ["two", "twen"],
'3': ["three", "thir"],
'4': ["four", "for"],
'5': ["five", "fif"],
'6': ["six"],
'7': ["seven"],
'8': ["eight"],
'9': ["nine"],
}
```

```
two_digit_words = ["ten", "eleven", "twelve"]
```

```
hundred = "hundred"
```

```
large_sum_words = ["thousand", "million", "billion", "trillion",
"quadrillion", "quintillion", "sextillion", "septillion", "octillion",
"nonillion"]
```

```
def converter(n):
```

```
    word = []
```

```
    if n.startswith('-):
```

```
        word.append("(negative)")
```

```
        n = n[1:]
```

```
    if len(n) % 3 != 0 and len(n) > 3:
```

```
        n = n.zfill(3 * (((len(n)-1) // 3) + 1))
```



```
sum_list = [n[i:i + 3] for i in range(0, len(n), 3)]
```

```
skip = False
```

```
for i, num in enumerate(sum_list):
```

```
    if num != '000': skip = False
```

```
    for _ in range(len(num)):
```

```
        num = num.lstrip('0')
```

```
        if len(num) == 1:
```

```
            if (len(sum_list) > 1 or (len(sum_list) == 1 and len(sum_list[0])  
== 3)) and i == len(sum_list) - 1 and (word[-1] in large_sum_words or  
hundred in word[-1]):
```

```
                word.append("and")
```

```
                word.append(one_digit_words[num][0])
```

```
                num = num[1:]
```

```
                break
```

```
    if len(num) == 2:
```

```
        if num[0] != '0':
```

```
            if (len(sum_list) > 1 or (len(sum_list) == 1 and len(sum_list[0])  
== 3)) and i == len(sum_list) - 1:
```

```
                word.append("and")
```

```
                if num.startswith('1'):
```

```
                    if int(num[1]) in range(3):
```

```
                        word.append(two_digit_words[int(num[1])])
```

```

        else:
            number = one_digit_words[num[1]][1 if int(num[1]) in
range(3, 6, 2) else 0]
            word.append(number + ("teen" if not number[-1] == 't'
else "een"))
        else:
            word.append(one_digit_words[num[0]][1 if int(num[0]) in
range(2, 6) else 0] + ("ty " if num[0] != '8' else 'y ') +
(one_digit_words[num[1]][0] if num[1] != '0' else ""))
            break
    else:
        num = num[1:]
        continue

if len(num) == 3:
    if num[0] != '0':
        word.append(one_digit_words[num[0]][0] + " " + hundred)
        if num[1:] == '00': break
    num = num[1:]
if len(sum_list[i:]) > 1 and not skip:
    word.append(large_sum_words[len(sum_list[i:]) - 2])
    skip = True
word = " ".join(map(str.strip, word))
return word[0].lstrip().upper() + word[1:].rstrip().lower() if "negative"
not in word else word[:11].lstrip() + word[11].upper() +
word[12:].rstrip().lower()
if __name__ == "__main__":

```

```
while True:
    try:
        n = input("Enter any number to convert it into words or 'exit' to stop: ")
        if n == "exit":
            break
        int(n)
        print(n, "-->", converter(n))
    except ValueError:
        print("Error: Invalid Number!")
```

[OceanofPDF.com](http://OceanofPDF.com)

## 27. CONVERT\_PNG\_to\_ICON

This project contains a simple python script to convert a .png image to .ICO format

## Requirements

Pillow module

```
`pip install pillow`
```

## Two methods:

The conversion can be done in two ways:

### Using Terminal

- Add the image in png format with name as 'input' in this folder.
- Run the convert.py script.
- output image will be generated and saved in this folder with the name 'output'.

### Using GUI

- Run the convertUI.py script.
- Select the 'Import PNG File' button and pick any png image from any location.
- Select the 'Convert PNG to ICO' button and pick the location where the file will be saved.

Source Code:

```
from PIL import Image
# Take input image from present folder
img = Image.open('input.png')
# Generate and save output image to present folder
img.save('output.ico')
```

Source Code:

```
import tkinter as tk
from PIL import Image

# Initialize Tkinter window
root = tk.Tk()
# Initialize variable to store image path
img = None
# Initialize font, background color, foreground color and width for the
buttons
font = ('helvetica', 12, 'bold')
bg = 'blue'
fg = 'white'
width = 15
```

def getPNG():

"""Function to get png image location and open it with pillow"""

global img

```

import_file_path = tk.filedialog.askopenfilename(filetypes=[("PNG
File",'.png')])

img = Image.open(import_file_path)

def convertToICO():
    global img

    """Function to convert image from png to ico format with pillow and save
    to user specified location"""

    if img is None:
        tk.messagebox.showerror("Error", "No File selected")
    else:
        export_file_path =
tk.filedialog.asksaveasfilename(defaultextension='.ico')
        img.save(export_file_path)
        tk.messagebox.showinfo("Success", "File converted and saved")

# Set the window title
root.title('PNG to ICO Converter')
canvas1 = tk.Canvas(root, width=500, height=350, bg='lightblue')
canvas1.pack()
# Set the screen title
label1 = tk.Label(root, text='PNG to ICO Converter', bg='lightblue')
label1.config(font=('helvetica', 20))
canvas1.create_window(250, 100, window=label1)
# Browse button to browse for image
browseButton = tk.Button(text="Import PNG File", command=getPNG,
bg=bg, fg=fg, font=font, width=width)

```

```
canvas1.create_window(250, 150, window=browseButton)
# Convert button to convert selected image and save
saveAsButton = tk.Button(text='Convert PNG to ICO',
command=convertToICO, bg=bg, fg=fg, font=font, width=width)
canvas1.create_window(250, 200, window=saveAsButton)
root.mainloop()
```

[OceanofPDF.com](http://OceanofPDF.com)

## 28. Convert XML to JSON

### prerequisite

- you need to install below library using pip
- \$ pip install xmltodict

### Description

- It converts any input.xml file into output.json.

### How to run the script

- First rename your file to input.xml
- Execute `python3 converter.py`
- The Output will be shown below as output.json

**Source Code:**

```
import json
import xmltodict
```



```
with open('input.xml') as xml_file:
    parsed_data = xmltodict.parse(xml_file.read())

    xml_file.close()

    json_conversion = json.dumps(parsed_data)

    with open('output.json', 'w') as json_file:
        json_file.write(json_conversion)

    json_file.close()
```

[OceanofPDF.com](http://OceanofPDF.com)

## 29. REAL TIME COVID-19 OUTBREAK NOTIFICATION

This is a notifier built with the use of BeautifulSoup, which will keep updating the user about the situation of covid in a particular state the user wants to know about after a particular interval of time.

# What is the use ?

In this time of crisis, it is needed to keep ourselves updated with the cases of covid for our own safety and benefit, and to get a better view of the situation.

Everyone would like to have a system, that keeps them updated about the total cases, cured and deaths at an interval on its own. This script does exactly that for the user!

# How to Use :

Once the user starts the program, it will ask you about the state whose data you want to know, after it is entered, the notification system will do the work by itself and keep updating the user after an interval of an hour. The script will keep running in the background until the user exits it manually.

### Source Code:

```
from plyer import notification
import requests
from bs4 import BeautifulSoup
import time
from englisttohindi.englisttohindi import EngtoHindi

def notify_user(title, message):
    notification.notify(
        title = title,
        message = message,
        app_icon = "./Covid-19_Real-time_Notification/Notify_icon.ico" ,
        timeout = 5)

def getInfo(url):
    r = requests.get(url)
    return r.text

if __name__ == '__main__':
    t = int(input("Enter interval in secs: "))
    li = list(map(str, input("Enter name of states: ").split(",")))
    states = []
    for i in li:
```

```

states.append(i + " ( " + str(((EngtoHindi(i)).convert)) + " )")

while True:
    HtmlData = getInfo('https://www.medtalks.in/live-corona-counter-india')
    soup = BeautifulSoup(HtmlData, 'html.parser')

    myData = ""
    for tr in soup.find('tbody').find_all('tr'):
        myData += tr.get_text()
    myData = myData[1:]

    itemList = myData.split("\n\n")
    for item in itemList[:-2]:
        dataList = item.split('\n')

        if dataList[0] in states:
            nTitle = 'Cases of Covid-19'
            nText = f"State: {dataList[0]}: Total: {dataList[1]}\n Active: {dataList[2]}\n Death: {dataList[3]}"
            notify_user(nTitle, nText)
            time.sleep(2)
    time.sleep(t)

```

## 30. Covid-19-Update-Bot

Covid-19 Update Bot that will Notify about the current covid-19 Cases, Deaths, Recovered

# Dependencies:

\*json\*

...

pip install json

...

\*win 10 toast\*

...

pip install win10toast

...

\*Initially it shows the cases worldwide it will notify after every hour\*

\*If you want to see cases for india or any other country then change this line in the code\*

...

r = requests.get('https://coronavirus-19-api.herokuapp.com/countries/india')

...

### Source Code:

```
import requests
import json
from win10toast import ToastNotifier
from time import sleep

r = requests.get('https://coronavirus-19-api.herokuapp.com/all')

data = r.json()
text = f'Confirmed Cases : {data["cases"]} \nDeaths : {data["deaths"]} \nRecovered : {data["recovered"]}'
while True:
    toast = ToastNotifier()
    toast.show_toast("Covid-19 Notification",text ,duration=20)
    sleep(60)
```

[OceanofPDF.com](https://oceanofpdf.com)

## Module 4 Projects 31-40

[OceanofPDF.com](http://OceanofPDF.com)

## 31. Create Audio Book in Python

```
import PyPDF2

pdfReader = PyPDF2.PdfFileReader(open('file.pdf', 'rb'))

import pyttsx3
speaker = pyttsx3.init()

for page_num in range(pdfReader.numPages):
    text = pdfReader.getPage(page_num).extractText()
    speaker.say(text)
    speaker.runAndWait()
speaker.stop()

engine.save_to_file(text, 'audio.mp3')
engine.runAndWait()
```

[OceanofPDF.com](http://OceanofPDF.com)



## 32. create script to encrypt files and folder

### usage

python encrypt.py path(file or folder)

examples:

python encrypt.py test.txt(file)

or

python eccrypt.py ./testdir(folder)

encrypted files("original\_file\_name.bin") will be generated in original location after the program running

### Source Code:

```
import sys
```

```
import os
from Cryptodome.Cipher import AES
from Cryptodome import Random
from binascii import b2a_hex
```

```
def encrypt_dir(path):
    for root, _, files in os.walk("."):
        for file in files:
            file_path = os.path.join(root, file)
            print(file_path + " is encrypting.")
            encrypt_file(file_path)
```

```
def encrypt_file(path):
    # get the plaintext
    with open(path) as f:
        plain_text = f.read()

    # The key length must be 16 (AES-128), 24 (AES-192), or 32
    (AES-256) Bytes.
    key = b'this is a 16 key'

    iv = Random.new().read(AES.block_size)
    mycipher = AES.new(key, AES.MODE_CFB, iv)
    ciphertext = iv + mycipher.encrypt(plain_text.encode())
```

```
# output
with open(path + ".bin", "wb") as file_out:
    file_out.write(ciphertext[16:])

path = sys.argv[1]
if os.path.isdir(path) and os.path.exists(path):
    encrypt_dir(path)
elif os.path.isfile(path) and os.path.exists(path):
    encrypt_file(path)
else:
    print("it's a special file(socket,FIFO,device file)")
```

[OceanofPDF.com](http://OceanofPDF.com)

### 33. Create a stopwatch

```
import tkinter as Tkinter
from datetime import datetime
counter = 0
running = False

def counter_label(label):
    def count():
        if running:
            global counter
            # To manage the initial delay.
            if counter == 0:
                display = 'Ready!'
            else:
                tt = datetime.utcnow().timestamp()
                string = tt.strftime('%H:%M:%S')
                display = string
            label['text'] = display
            # label.after(arg1, arg2) delays by
            # first argument given in milliseconds
            # and then calls the function given as second argument.
            # Generally like here we need to call the
            # function in which it is present repeatedly.
```

```
# Delays by 1000ms=1 seconds and call count again.  
    label.after(1000, count)  
    counter += 1  
# Triggering the start of the counter.  
count()
```

```
# start function of the stopwatch
```

```
def Start(label):  
    global running  
    running = True  
    counter_label(label)  
    start['state'] = 'disabled'  
    stop['state'] = 'normal'  
    reset['state'] = 'normal'
```

```
# Stop function of the stopwatch
```

```
def Stop():  
    global running  
    start['state'] = 'normal'  
    stop['state'] = 'disabled'  
    reset['state'] = 'normal'  
    running = False
```

```
# Reset function of the stopwatch
```

```
def Reset(label):  
    global counter
```

```
counter = 0

# If reset is pressed after pressing stop.
if not running:
    reset['state'] = 'disabled'
    label['text'] = '00:00:00'
# If reset is pressed while the stopwatch is running.
else:
    label['text'] = '00:00:00'


root = Tkinter.Tk()
root.title("Stopwatch")

# Fixing the window size.
root.minsize(width=250, height=70)

label = Tkinter.Label(root, text='Ready!', fg='black', font='Verdana 30
bold')
label.pack()

f = Tkinter.Frame(root)

start = Tkinter.Button(f, text='Start', width=6, command=lambda:
Start(label))

stop = Tkinter.Button(f, text='Stop', width=6, state='disabled',
command=Stop)

reset = Tkinter.Button(f, text='Reset', width=6, state='disabled',
command=lambda: Reset(label))

f.pack(anchor='center', pady=5)
start.pack(side='left')
```

```
stop.pack(side='left')  
reset.pack(side='left')  
root.mainloop()
```

[OceanofPDF.com](http://OceanofPDF.com)

### 34. Create Calculator App.

<!--Remove the below lines and add yours -->

A small python program that creates a calculator app

### Prerequisites

<!--Remove the below lines and add yours -->

Python 3

### How to run the script

<!--Remove the below lines and add yours -->

> python calculator.py

**Source Code:**

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



```
from tkinter import Tk, END, Entry, N, E, S, W, Button
from tkinter import font
from tkinter import Label
from functools import partial
```

```
def get_input(entry, argu):
    entry.insert(END, argu)
```

```
def backspace(entry):
    input_len = len(entry.get())
    entry.delete(input_len - 1)
```

```
def clear(entry):
    entry.delete(0, END)
```

```
def calc(entry):
    input_info = entry.get()
    try:
        output = str(eval(input_info.strip()))
    except ZeroDivisionError:
        popupmsg()
```

```
    output = ""  
    clear(entry)  
    entry.insert(END, output)
```

```
def popupmsg():  
    popup = Tk()  
    popup.resizable(0, 0)  
    popup.geometry("120x100")  
    popup.title("Alert")  
    label = Label(popup, text="Cannot divide by 0 ! \n Enter valid values")  
    label.pack(side="top", fill="x", pady=10)  
    B1 = Button(popup, text="Okay", bg="#DDDDDD",  
command=popup.destroy)  
    B1.pack()
```

```
def cal():  
    root = Tk()  
    root.title("Calc")  
    root.resizable(0, 0)  
  
    entry_font = font.Font(size=15)  
    entry = Entry(root, justify="right", font=entry_font)  
    entry.grid(row=0, column=0, columnspan=4,  
        sticky=N + W + S + E, padx=5, pady=5)
```

```
cal_button_bg = '#FF6600'
num_button_bg = '#4B4B4B'
other_button_bg = '#DDDDDD'
text_fg = '#FFFFFF'
button_active_bg = '#C0C0C0'
```

```
num_button = partial(Button, root, fg=text_fg, bg=num_button_bg,
                      padx=10, pady=3, activebackground=button_active_bg)
cal_button = partial(Button, root, fg=text_fg, bg=cal_button_bg,
                     padx=10, pady=3, activebackground=button_active_bg)
```

```
button7 = num_button(text='7', bg=num_button_bg,
                    command=lambda: get_input(entry, '7'))
button7.grid(row=2, column=0, pady=5)
```

```
button8 = num_button(text='8', command=lambda: get_input(entry, '8'))
button8.grid(row=2, column=1, pady=5)
```

```
button9 = num_button(text='9', command=lambda: get_input(entry, '9'))
button9.grid(row=2, column=2, pady=5)
```

```
button10 = cal_button(text='+', command=lambda: get_input(entry, '+'))
button10.grid(row=4, column=3, pady=5)
```

```
button4 = num_button(text='4', command=lambda: get_input(entry, '4'))  
button4.grid(row=3, column=0, pady=5)
```

```
button5 = num_button(text='5', command=lambda: get_input(entry, '5'))  
button5.grid(row=3, column=1, pady=5)
```

```
button6 = num_button(text='6', command=lambda: get_input(entry, '6'))  
button6.grid(row=3, column=2, pady=5)
```

```
button11 = cal_button(text='-', command=lambda: get_input(entry, '-'))  
button11.grid(row=3, column=3, pady=5)
```

```
button1 = num_button(text='1', command=lambda: get_input(entry, '1'))  
button1.grid(row=4, column=0, pady=5)
```

```
button2 = num_button(text='2', command=lambda: get_input(entry, '2'))  
button2.grid(row=4, column=1, pady=5)
```

```
button3 = num_button(text='3', command=lambda: get_input(entry, '3'))  
button3.grid(row=4, column=2, pady=5)
```

```
button12 = cal_button(text='*', command=lambda: get_input(entry, '*'))  
button12.grid(row=2, column=3, pady=5)
```

```
button0 = num_button(text='0', command=lambda: get_input(entry, '0'))
```

```
#button0.grid(row=5, column=0, columnspan=2, padx=3, pady=5,  
sticky=N + S + E + W)
```

```
button0.grid(row=5, column=0, pady=5)
```

```
button13 = num_button(text='.', command=lambda: get_input(entry, '.'))
```

```
button13.grid(row=5, column=1, pady=5)
```

```
button14 = Button(root, text='/', fg=text_fg, bg=cal_button_bg, padx=10,  
pady=3,
```

```
command=lambda: get_input(entry, '/'))
```

```
button14.grid(row=1, column=3, pady=5)
```

```
button15 = Button(root, text='<-', bg=other_button_bg, padx=10,  
pady=3,
```

```
command=lambda: backspace(entry),  
activebackground=button_active_bg)
```

```
button15.grid(row=1, column=0, columnspan=2,  
padx=3, pady=5, sticky=N + S + E + W)
```

```
button16 = Button(root, text='C', bg=other_button_bg, padx=10, pady=3,  
command=lambda: clear(entry),  
activebackground=button_active_bg)
```

```
button16.grid(row=1, column=2, pady=5)
```

```
button17 = Button(root, text='=', fg=text_fg, bg=cal_button_bg,  
padx=10, pady=3,
```

```
        command=lambda: calc(entry),  
activebackground=button_active_bg)
```

```
    button17.grid(row=5, column=3, pady=5)
```

```
    button18 = Button(root, text='^', fg=text_fg, bg=cal_button_bg, padx=10,  
pady=3,
```

```
        command=lambda: get_input(entry, '**'))
```

```
    button18.grid(row=5, column=2, pady=5)
```

```
root.mainloop()
```

```
if __name__ == '__main__':
```

```
    cal()
```

## 35. Currency Converter

<!--Remove the below lines and add yours -->

A small python program that converts currency with live info

### Prerequisites

<!--Remove the below lines and add yours -->

- requests

- Python 3

### How to run the script

<!--Remove the below lines and add yours -->

> python cc.py

**Source Code:**

''''''

This program is capable of converting from one currency to another as of today itself.

It uses the api at fixer.io and then calculates the value of the currency in terms of the other as of today.

```
"""
```

```
import requests
```

```
import json
```

```
import sys
```

```
from pprint import pprint
```

```
# The below 4 lines bring out the value of currency from the api at fixer.io.  
I had to register there, the key is unique to me.
```

```
url = "http://data.fixer.io/api/latest?  
access_key=33ec7c73f8a4eb6b9b5b5f95118b2275"
```

```
data = requests.get(url).text
```

```
data2 = json.loads(data) #brings whether request was successful,timestamp  
etc
```

```
fx = data2["rates"]
```

```
currencies = [
```

```
    "AED : Emirati Dirham,United Arab Emirates Dirham",
```

```
    "AFN : Afghan Afghani,Afghanistan Afghani",
```

```
    "ALL : Albanian Lek,Albania Lek",
```

```
    "AMD : Armenian Dram,Armenia Dram",
```

```
    "ANG : Dutch Guilder,Netherlands Antilles  
Guilder,Bonaire,Cura&#231;ao,Saba,Sint Eustatius,Sint Maarten",
```



"AOA : Angolan Kwanza,Angola Kwanza",

"ARS : Argentine Peso,Argentina Peso,Islas Malvinas",

"AUD : Australian Dollar,Australia Dollar,Christmas Island,Cocos (Keeling) Islands,Norfolk Island,Ashmore and Cartier Islands,Australian Antarctic Territory,Coral Sea Islands,Heard Island,McDonald Islands,Kiribati,Nauru",

"AWG : Aruban or Dutch Guilder,Aruba Guilder",

"AZN : Azerbaijan Manat,Azerbaijan Manat",

"BAM : Bosnian Convertible Mark,Bosnia and Herzegovina Convertible Mark",

"BBD : Barbadian or Bajan Dollar,Barbados Dollar",

"BDT : Bangladeshi Taka,Bangladesh Taka",

"BGN : Bulgarian Lev,Bulgaria Lev",

"BHD : Bahraini Dinar,Bahrain Dinar",

"BIF : Burundian Franc,Burundi Franc",

"BMD : Bermudian Dollar,Bermuda Dollar",

"BND : Bruneian Dollar,Brunei Darussalam Dollar",

"BOB : Bolivian Boliviano,Bolivia Boliviano",

"BRL : Brazilian Real,Brazil Real",

"BSD : Bahamian Dollar,Bahamas Dollar",

"BTC : Bitcoin,BTC, XBT",

"BTN : Bhutanese Ngultrum,Bhutan Ngultrum",

"BWP : Botswana Pula,Botswana Pula",

"BYN : Belarusian Ruble,Belarus Ruble",

"BYR : Belarusian Ruble,Belarus Ruble",

"BZD : Belizean Dollar,Belize Dollar",

"CAD : Canadian Dollar,Canada Dollar",

"CDF : Congolese Franc,Congo/Kinshasa Franc",

"CHF : Swiss Franc,Switzerland Franc,Liechtenstein,Campione d&#039;Italia,B&#252;singen am Hochrhein",

"CLF : Chilean Unit of Account",

"CLP : Chilean Peso,Chile Peso",

"CNY : Chinese Yuan Renminbi,China Yuan Renminbi",

"COP : Colombian Peso,Colombia Peso",

"CRC : Costa Rican Colon,Costa Rica Colon",

"CUC : Cuban Convertible Peso,Cuba Convertible Peso",

"CUP : Cuban Peso,Cuba Peso",

"CVE : Cape Verdean Escudo,Cape Verde Escudo",

"CZK : Czech Koruna,Czech Republic Koruna",

"DJF : Djiboutian Franc,Djibouti Franc",

"DKK : Danish Krone,Denmark Krone,Faroe Islands,Greenland",

"DOP : Dominican Peso,Dominican Republic Peso",

"DZD : Algerian Dinar,Algeria Dinar",

"EGP : Egyptian Pound,Egypt Pound,Gaza Strip",

"ERN : Eritrean Nakfa,Eritrea Nakfa",

"ETB : Ethiopian Birr,Ethiopia Birr,Eritrea",

"EUR : Euro,Euro Member Countries,Andorra,Austria,Azores,Baleares (Balearic Islands),Belgium,Canary Islands,Cyprus,Finland,France,French Guiana,French Southern Territories,Germany,Greece,Guadeloupe,Holland (Netherlands),Holy See (Vatican City),Ireland (Eire),Italy,Luxembourg,Madeira Islands,Malta,Monaco,Montenegro,Netherlands",

"FJD : Fijian Dollar,Fiji Dollar",

"FKP : Falkland Island Pound,Falkland Islands (Malvinas) Pound",

"GBP : British Pound,United Kingdom Pound,United Kingdom (UK),England,Northern Ireland,Scotland,Wales,Falkland Islands,Gibraltar,Guernsey,Isle of Man,Jersey,Saint Helena and Ascension,South Georgia and the South Sandwich Islands,Tristan da Cunha",

"GEL : Georgian Lari,Georgia Lari",

"GGP : Guernsey Pound,Guernsey Pound",

"GHS : Ghanaian Cedi,Ghana Cedi",

"GIP : Gibraltar Pound,Gibraltar Pound",

"GMD : Gambian Dalasi,Gambia Dalasi",

"GNF : Guinean Franc,Guinea Franc",

"GTQ : Guatemalan Quetzal,Guatemala Quetzal",

"GYD : Guyanese Dollar,Guyana Dollar",

"HKD : Hong Kong Dollar,Hong Kong Dollar",

"HNL : Honduran Lempira,Honduras Lempira",

"HRK : Croatian Kuna,Croatia Kuna",

"HTG : Haitian Gourde,Haiti Gourde",

"HUF : Hungarian Forint,Hungary Forint",

"IDR : Indonesian Rupiah,Indonesia Rupiah,East Timor",

"ILS : Israeli Shekel,Israel Shekel,Palestinian Territories",

"IMP : Isle of Man Pound,Isle of Man Pound",

"INR : Indian Rupee,India Rupee,Bhutan,Nepal",

"IQD : Iraqi Dinar,Iraq Dinar",

"IRR : Iranian Rial,Iran Rial",

"ISK : Icelandic Krona,Iceland Krona",

"JEP : Jersey Pound,Jersey Pound",

"JMD : Jamaican Dollar,Jamaica Dollar",

"JOD : Jordanian Dinar,Jordan Dinar",  
"JPY : Japanese Yen,Japan Yen",  
"KES : Kenyan Shilling,Kenya Shilling",  
"KGS : Kyrgyzstani Som,Kyrgyzstan Som",  
"KHR : Cambodian Riel,Cambodia Riel",  
"KMF : Comorian Franc,Comorian Franc",  
"KPW : North Korean Won,Korea (North) Won",  
"KRW : South Korean Won,Korea (South) Won",  
"KWD : Kuwaiti Dinar,Kuwait Dinar",  
"KYD : Caymanian Dollar,Cayman Islands Dollar",  
"KZT : Kazakhstani Tenge,Kazakhstan Tenge",  
"LAK : Lao Kip,Laos Kip",  
"LBP : Lebanese Pound,Lebanon Pound",  
"LKR : Sri Lankan Rupee,Sri Lanka Rupee",  
"LRD : Liberian Dollar,Liberia Dollar",  
"LSL : Basotho Loti,Lesotho Loti",  
"LTL : Lithuanian litas",  
"LVL : Latvia Lats",  
"LYD : Libyan Dinar,Libya Dinar",  
"MAD : Moroccan Dirham,Morocco Dirham,Western Sahara",  
"MDL : Moldovan Leu,Moldova Leu",  
"MGA : Malagasy Ariary,Madagascar Ariary",  
"MKD : Macedonian Denar,Macedonia Denar",  
"MMK : Burmese Kyat,Myanmar (Burma) Kyat",  
"MNT : Mongolian Tughrik,Mongolia Tughrik",

"MOP : Macau Pataca,Macau Pataca",  
"MRU : Mauritanian Ouguiya,Mauritania Ouguiya",  
"MUR : Mauritian Rupee,Mauritius Rupee",  
"MVR : Maldivian Rufiyaa,Maldives (Maldiv Islands) Rufiyaa",  
"MWK : Malawian Kwacha,Malawi Kwacha",  
"MXN : Mexican Peso,Mexico Peso",  
"MYR : Malaysian Ringgit,Malaysia Ringgit",  
"MZN : Mozambican Metical,Mozambique Metical",  
"NAD : Namibian Dollar,Namibia Dollar",  
"NGN : Nigerian Naira,Nigeria Naira",  
"NIO : Nicaraguan Cordoba,Nicaragua Cordoba",  
"NOK : Norwegian Krone,Norway Krone,Bouvet Island,Svalbard,Jan Mayen,Queen Maud Land,Peter I Island",  
"NPR : Nepalese Rupee,Nepal Rupee,India (unofficially near India-Nepal border)",  
"NZD : New Zealand Dollar,New Zealand Dollar,Cook Islands,Niue,Pitcairn Islands,Tokelau",  
"OMR : Omani Rial,Oman Rial",  
"PAB : Panamanian Balboa,Panama Balboa",  
"PEN : Peruvian Sol,Peru Sol",  
"PGK : Papua New Guinean Kina,Papua New Guinea Kina",  
"PHP : Philippine Peso,Philippines Peso",  
"PKR : Pakistani Rupee,Pakistan Rupee",  
"PLN : Polish Zloty,Poland Zloty",  
"PYG : Paraguayan Guarani,Paraguay Guarani",  
"QAR : Qatari Riyal,Qatar Riyal",

"RON : Romanian Leu,Romania Leu",  
"RSD : Serbian Dinar,Serbia Dinar",  
"RUB : Russian Ruble,Russia Ruble,Tajikistan,Abkhazia,South Ossetia",  
"RWF : Rwandan Franc,Rwanda Franc",  
"SAR : Saudi Arabian Riyal,Saudi Arabia Riyal",  
"SBD : Solomon Islander Dollar,Solomon Islands Dollar",  
"SCR : Seychellois Rupee,Seychelles Rupee",  
"SDG : Sudanese Pound,Sudan Pound",  
"SEK : Swedish Krona,Sweden Krona",  
"SGD : Singapore Dollar,Singapore Dollar",  
"SHP : Saint Helenian Pound,Saint Helena Pound",  
"SLL : Sierra Leonean Leone,Sierra Leone Leone",  
"SOS : Somali Shilling,Somalia Shilling",  
"SRD : Surinamese Dollar,Suriname Dollar",  
"STN : Sao Tomean Dobra,S&#227;o Tom&#233; and Pr&#237;ncipe Dobra",  
"SVC : Salvadoran Colon,El Salvador Colon",  
"SYP : Syrian Pound,Syria Pound",  
"SZL : Swazi Lilangeni,eSwatini Lilangeni",  
"THB : Thai Baht,Thailand Baht",  
"TJS : Tajikistani Somoni,Tajikistan Somoni",  
"TMT : Turkmenistani Manat,Turkmenistan Manat",  
"TND : Tunisian Dinar,Tunisia Dinar",  
"TOP : Tongan Pa&#039;anga,Tonga Pa&#039;anga",  
"TRY : Turkish Lira,Turkey Lira,North Cyprus",  
"TTD : Trinidadian Dollar,Trinidad and Tobago Dollar,Trinidad,Tobago",

"TWD : Taiwan New Dollar,Taiwan New Dollar",

"TZS : Tanzanian Shilling,Tanzania Shilling",

"UAH : Ukrainian Hryvnia,Ukraine Hryvnia",

"UGX : Ugandan Shilling,Uganda Shilling",

"USD : US Dollar,United States Dollar,America,American Samoa,American Virgin Islands,British Indian Ocean Territory,British Virgin Islands,Ecuador,El Salvador,Guam,Haiti,Micronesia,Northern Mariana Islands,Palau,Panama,Puerto Rico,Turks and Caicos Islands,United States Minor Outlying Islands,Wake Island,East Timor",

"UYU : Uruguayan Peso,Uruguay Peso",

"UZS : Uzbekistani Som,Uzbekistan Som",

"VEF : Venezuelan Bolívar,Venezuela Bolívar",

"VND : Vietnamese Dong,Viet Nam Dong",

"VUV : Ni-Vanuatu Vatu, Vanuatu Vatu",

"WST : Samoan Tala,Samoa Tala",

"XAF : Central African CFA Franc BEAC,Communauté Financière Africaine (BEAC) CFA Franc BEAC,Cameroon,Central African Republic,Chad,Congo/Brazzaville,Equatorial Guinea,Gabon",

"XAG : Silver Ounce,Silver",

"XAU : Gold Ounce,Gold",

"XCD : East Caribbean Dollar,East Caribbean Dollar,Anguilla,Antigua and Barbuda,Dominica,Grenada,The Grenadines and Saint Vincent,Montserrat",

"XDR : IMF Special Drawing Rights,International Monetary Fund (IMF) Special Drawing Rights",

"XOF : CFA Franc,Communauté Financière Africaine (BCEAO) Franc,Benin,Burkina Faso,Ivory Coast,Guinea-Bissau,Mali,Niger,Senegal,Togo",

```

    "XPF : CFP Franc,Comptoirs Fran&#231;ais du Pacifique (CFP)
    Franc,French Polynesia,New Caledonia,Wallis and Futuna Islands",
    "YER : Yemeni Rial,Yemen Rial",
    "ZAR : South African Rand,South Africa Rand,Lesotho,Namibia",
    "ZMK : Zambian Kwacha,Zambia Kwacha",
    "ZMW : Zambian Kwacha,Zambia Kwacha",
    "ZWL : Zimbabwean Dollar,Zimbabwe Dollar",
]

```

# The below function calculates the actual conversion

```
def function1():
```

```
    query = input(
```

```
        "Please specify the amount of currency to convert, from currency, to
        currency (with space in between).\nPress SHOW to see list of currencies
        available. \nPress Q to quit. \n"
```

```
    )
```

```
    if query == "Q":
```

```
        sys.exit()
```

```
    elif query == "SHOW":
```

```
        pprint(currencies)
```

```
        function1()
```

```
    else:
```

```
        qty, fromC, toC = query.split(" ")
```

```
        fromC = fromC.upper()
```

```
        toC = toC.upper()
```



```
qty = float(round(int(qty), 2))
amount = round(qty * fx[toC] / fx[fromC], 2)
print(f"{qty} of currency {fromC} amounts to {amount} of currency
{toC} today")
```

```
try:
```

```
    function1()
```

```
except KeyError:
```

```
    print("You seem to have inputted wrongly, retry!")
```

```
    function1()
```

[OceanofPDF.com](http://OceanofPDF.com)

## 36. Currency Exchange Rate

```
from bs4 import BeautifulSoup
import requests as req

currencies = []

page = req.get('https://www.x-rates.com/').text

soup = BeautifulSoup(page, 'html.parser')

options = soup.find_all('option')[:-11]

for option in options:
    currency_short = option.text[:option.text.find(" ")]
    currency_name = option.text[(option.text.find(" ") + 3):]
    current_element = {'name': currency_name, 'short': currency_short}
    currencies.append(current_element)
    print('{} . {} ({} )'.format(len(currencies), current_element['name'],
                                  current_element['short']))

currency_index = int(input('Enter your currency\'s position number: '))
- 1
currency = currencies[currency_index]
amount = input(
```

\033cEnter amount of {}s (if amount isn't integer, then write it with a dot, not comma): '

```
.format(currency['name'].lower()))
```

```
currencies_table_url = 'https://www.x-rates.com/table/?from=
{}&amount={}'.format(
    currency['short'], amount)
```

```
currencies_table_page = req.get(currencies_table_url).text
```

```
soup = BeautifulSoup(currencies_table_page, 'html.parser')
```

```
table_rows = soup.findChild('table', attrs={
    'class': 'tablesorter'
}).findChildren('tr')[1:]
```

```
print('\033cFor {} {}s you'll get:'.format(amount,
currency['name'].lower()))
```

```
for table_row in table_rows:
```

```
    row_data = table_row.findChildren('td')
```

```
    exchange_rate = {
```

```
        'currency': row_data[0].text,
```

```
        'amount': float(row_data[1].text)
```

```
    }
```

```
print('{:.3f} {}s'.format(exchange_rate['amount'],
```

```
exchange_rate['currency']))
```

[OceanofPDF.com](http://OceanofPDF.com)

## 37. Decimal to Binary Converter

<!--Remove the below lines and add yours -->

A small python program that converts binary and decimal

### Prerequisites

<!--Remove the below lines and add yours -->

- Python 3

### How to run the script

<!--Remove the below lines and add yours -->

> python decimal\_to\_binary.py

**Source Code:**

```
try:
    menu = int(input("Choose an option: \n 1. Decimal to binary \n 2. Binary
to decimal\n Option: "))
    if menu < 1 or menu > 2:
        raise ValueError
    if menu == 1:
        dec = int(input("Input your decimal number:\nDecimal: "))
        print("Binary: {}".format(bin(dec)[2:]))
    elif menu == 2:
        binary = input("Input your binary number:\n Binary: ")
        print("Decimal: {}".format(int(binary, 2)))
except ValueError:
    print ("please choose a valid option")
```

## 38. Desktop Notification Application

```
import time
from plyer import notification

if __name__ == "__main__":
    while True:
        notification.notify(
            title = "ALERT!!!",
            message = "Take a break! It has been an hour!",
            timeout = 10
        )
        time.sleep(3600)
```

[OceanofPDF.com](http://OceanofPDF.com)

### 39. Dictionary GUI

This script lets the user search for the meaning of words like a dictionary.

## Setup instructions

In order to run this script, you need to have Python and pip installed on your system. After you're done installing Python and pip, run the following command from your terminal to install the requirements from the same folder (directory) of the project.

...

`pip install -r requirements.txt`

...

After satisfying all the requirements for the project, Open the terminal in the project folder and run

...

`python dictionary.py`

...

or

...

`python3 dictionary.py`

...

depending upon the python version. Make sure that you are running the command from the same virtual environment in which the required modules are installed.

Source Code:



```

from tkinter import *
from tkinter import messagebox
from PyDictionary import PyDictionary

# Creating Tkinter Scaffold
root = Tk()
root.title("Dictionary")
root.geometry("500x400")

# Initialize dictionary objecy
dictionary = PyDictionary()

def getMeaning():
    response = dictionary.meaning(word.get())
    if(response):
        if('Noun' in response):
            meaning = response['Noun'][0]
        elif('Verb' in response):
            meaning = response['Verb'][0]
    elif('Adjective' in response):
        meaning = response['Adjective'][0]
    else:
        meaning = "Invalid word"
else:

```

```
messagebox.showinfo(  
    "Error", "Please add a Noun, Pronoun, verb or a valid word.")  
# Show meaning in frame  
meaning_label.config(text=meaning)
```

```
# Heading Label
```

```
heading_label = Label(root, text="DICTIONARY", font=(  
    "Helvetica 35 bold"), foreground='Blue')  
heading_label.config(anchor=CENTER)  
heading_label.pack(pady=10)
```

```
# Frame for search box and search button
```

```
frame = Frame(root)  
Label(frame, text="Enter Word", font=("Helvetica 15  
bold")).pack(side=LEFT)  
word = Entry(frame, font=("Helvetica 15 bold"))  
word.pack(padx=10)  
frame.pack()
```

```
search_button = Button(root, text="Search Word", font=("Helvetica  
15 bold"), relief=RIDGE,  
    borderwidth=3, cursor="hand2", foreground='Green',  
    command=getMeaning)  
search_button.config(anchor=CENTER)  
search_button.pack(pady=10)
```

```
# Frame to display meaning
frame1 = Frame(root)
Label(frame1, text="Meaning : ", font=("Helvetica 15
bold")).pack(side=LEFT)
meaning_label = Label(frame1, text="", font=("Helvetica 12"))
meaning_label.pack(pady=5)
frame1.pack(pady=10)

# Execute Tkinter
root.mainloop()
```

[OceanofPDF.com](http://OceanofPDF.com)

## 40. Digital Clock GUI:

```
from tkinter import Label, Tk
import time

app_window = Tk()
app_window.title("Digital Clock")
app_window.geometry("420x150")
app_window.resizable(1,1)

text_font= ("Boulder", 68, 'bold')
background = "#f2e750"
foreground= "#363529"
border_width = 25

label = Label(app_window, font=text_font, bg=background, fg=foreground,
              bd=border_width)
label.grid(row=0, column=1)

def digital_clock():
    time_live = time.strftime("%H:%M:%S")
    label.config(text=time_live)
```

```
label.after(200, digital_clock)
```

```
digital_clock()
```

```
app_window.mainloop()
```

[OceanofPDF.com](http://OceanofPDF.com)

## Module 5 Projects 41-50

[OceanofPDF.com](http://OceanofPDF.com)

## 41. Digital Clock using Python and Tkinter

This script create a digital clock as per the system's current time.

## Library Used

- \* tkinter

- \* time

### To install required external modules

- \* Run `pip install tkinter`

### How to run the script

- Execute `python3 digital\_clock.py`

### Source Code:

```
import tkinter as tk
```

```
from time import strftime
```

```
def light_theme():
```

```
frame = tk.Frame(root, bg="white")
frame.place(relx=0.1, rely=0.1, relwidth=0.8, relheight=0.8)
lbl_1 = tk.Label(frame, font=('calibri', 40, 'bold'),
                 background='White', foreground='black')
lbl_1.pack(anchor="s")
```

```
def time():
    string = strftime('%I:%M:%S %p')
    lbl_1.config(text=string)
    lbl_1.after(1000, time)
time()
```

```
def dark_theme():
    frame = tk.Frame(root, bg="#22478a")
    frame.place(relx=0.1, rely=0.1, relwidth=0.8, relheight=0.8)
    lbl_2 = tk.Label(frame, font=('calibri', 40, 'bold'),
                    background='#22478a', foreground='black')
    lbl_2.pack(anchor="s")
```

```
def time():
    string = strftime('%I:%M:%S %p')
    lbl_2.config(text=string)
    lbl_2.after(1000, time)
```



```
time()
```

```
root = tk.Tk()
```

```
root.title("Digital-Clock")
```

```
canvas = tk.Canvas(root, height=140, width=400)
```

```
canvas.pack()
```

```
frame = tk.Frame(root, bg='#22478a')
```

```
frame.place(relx=0.1, rely=0.1, relwidth=0.8, relheight=0.8)
```

```
lbl = tk.Label(frame, font=('calibri', 40, 'bold'),
```

```
                background='#22478a', foreground='black')
```

```
lbl.pack(anchor="s")
```

```
def time():
```

```
    string = strftime('%I:%M:%S %p')
```

```
    lbl.config(text=string)
```

```
    lbl.after(1000, time)
```

```
time()
```

```
menubar = tk.Menu(root)
```

```
theme_menu = tk.Menu(menubar, tearoff=0)
```

```
theme_menu.add_command(label="Light", command=light_theme)
```

```
theme_menu.add_command(label="Dark", command=dark_theme)
```

```
menubar.add_cascade(label="Theme", menu=theme_menu)
```

```
root.config(menu=menubar)
```

```
root.mainloop()
```

[OceanofPDF.com](http://OceanofPDF.com)

## 42. Discord Bot

Then insert your token at the top of the file.

Make sure you have discord.py installed:

```
```bash  
pip install discord  
```
```

Then just run the bot using python:

```
```bash  
python bot.py  
```
```

### Source Code:

```
import random
```

```
import discord
from discord.ext import commands

# your token here, inside the ""
TOKEN = ""

# channel to send welcome messages to
WELCOME_CHANNEL = "welcome"

# all the nicknames for the random_nickname command
NICKS = ["example1", "example2", "example3"]

# you can change the prefix here
bot = commands.Bot(command_prefix="!")
```

```
@bot.event
async def on_ready():
    print("bot started")
```

```
@bot.event
async def on_member_join(member):
    welcome_channel = discord.utils.get(member.guild.channels,
                                         name=WELCOME_CHANNEL)
    # feel free to change this message!
```

```
    await welcome_channel.send(
        f"welcome {member.mention}, please read our rules and have a great
time!"
    )
```

```
@commands.has_permissions(ban_members=True)
@bot.command()
async def ban(ctx, user: discord.Member):
    """Ban the given user"""
    await ctx.guild.ban(user, delete_message_days=0)
    await ctx.send(f"banned {user}")
```

```
@commands.has_permissions(ban_members=True)
@bot.command()
async def unban(ctx, user: discord.User):
    "Unban the given user"
    await ctx.guild.unban(user)
    await ctx.send(f"unbanned {user}")
```

```
@commands.has_permissions(kick_members=True)
@bot.command()
async def kick(ctx, user: discord.User):
    "Kick the given user"
```

```
await ctx.guild.kick(user)
await ctx.send(f"kicked {user}")
```

```
@bot.command(aliases=["rnick"])
async def random_nick(ctx):
    """Set your nickname to a random one"""
    new_nick = random.choice(NICKS)
    await ctx.author.edit(nick=new_nick)
    await ctx.send(f"Your new nickname is {new_nick}")
```

```
@commands.has_permissions(manage_nicknames=True)
@bot.command(aliases=["change_name"])
async def change_nick(ctx, user: discord.Member, *, new_nick):
    """Change somebody else's nickname."""
    await user.edit(nick=new_nick)
    await ctx.send(f"Changed the nick of {user.mention} to `{new_nick}`")
```

```
if __name__ == "__main__":
    bot.run(TOKEN)
```

## 43. DNS Record

This script takes the website name as input and returns its dns records.

#Requirements to run this file:

External library called dnspython has been used here and it can be installed easily by using the following command:

```
pip install -r requirements.txt
```

#How to use this script?

- 1.Install the requirements.
2. Type the following command

```
python dns_record.py
```

- 3.It will ask for a website:

You can give any website name for example: google.com

### Source Code:

```
#Simple program to fetch dns record of a given website
```

```
import dns.resolver

#Dictionary to store the dns record of a website
dns_record = {}

#User defined website
website = input("Enter the name of the website: ")

#Fetching the 'A' record of the website and storing it in the dictionary
a_record = dns.resolver.resolve(website, 'A')
for ipval in a_record:
    dns_record['A_Record_IP'] = ipval.to_text()

#List to store the mx records of a website
mx_record_list = []

#Fetching the mx records and storing them in the dictionary
mx_record = dns.resolver.resolve(website, 'MX')
for server in mx_record:
    mx_record_list.append(server)
for i, element in enumerate(mx_record_list):
    dns_record['MX_Record', i+1] = element

#Displaying the record on the screen
for key, value in dns_record.items():
```



```
print(f"{key} = {value}")
```

[OceanofPDF.com](https://oceanofpdf.com)

## 44. Document-Summary-Creator

A python script to create a sentence summary

## Prerequisites

##### This script needs Python 3.\*

pip install these libraries from requirements.txt

- \* sumy

- \* spacy

- \* neologdn

and run the command to download some libraries

```
```bash
```

```
$ python -m spacy download en_core_web_sm
```

```
$ python -c "import nltk; nltk.download('punkt')"
```

```
``
```

## Usage:

- \* Run main.py and enter the path of the text file

- \* After that, a text file that summarizes the read text file into two tenths is created

### Requirements:

**sumy==0.8.1**

**spacy==2.3.2**

**neologdn==0.4**

### **Main.py Source Code:**

```
from summary_make import summarize_sentences
```

```
def main():
```

```
    filepath = input("please input text's filepath->")
```

```
    with open(filepath) as f:
```

```
        sentences = f.readlines()
```

```
    sentences = ' '.join(sentences)
```

```
    summary = summarize_sentences(sentences)
```

```
    filepath_index = filepath.find('.txt')
```

```
    outputpath = filepath[:filepath_index] + '_summary.txt'
```

```
    with open(outputpath, 'w') as w:
```

```
        for sentence in summary:
```

```
            w.write(str(sentence) + '\n')
```

```
if __name__ == "__main__":
```

```
    main()
```

## **Pre-processing Source Code:**

```
import spacy
import neologdn
class EnglishCorpus:
    # Preparation of morphological analyzer
    def __init__(self):
        self.nlp = spacy.load("en_core_web_sm")

    # Pre-processing of line breaks and special characters
    def preprocessing(self, text: str) -> str:
        text = text.replace("\n", "")
        text = neologdn.normalize(text)

        return text

    # Divide sentences into sentences while retaining the results of
    morphological analysis
    def make_sentence_list(self, sentences: str) -> list:
        doc = self.nlp(sentences)
        self.ginza_sents_object = doc.sents
        sentence_list = [s for s in doc.sents]

        return sentence_list

    # Put a space between words
```

```
def make_corpus(self) -> list:
    corpus = []
    for s in self.ginza_sents_object:
        tokens = [str(t) for t in s]
        corpus.append(" ".join(tokens))

    return corpus
```

### **Summary Make Source Code:**

```
from preprocessing import EnglishCorpus

from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.utils import get_stop_words
from sumy.summarizers.lex_rank import LexRankSummarizer
```

```
def summarize_sentences(sentences: str, language="english") -> list:
    # Preparation sentences
```

```
corpus_maker = EnglishCorpus()
preprocessed_sentences = corpus_maker.preprocessing(sentences)
preprocessed_sentence_list = corpus_maker.make_sentence_list(
    preprocessed_sentences)
corpus = corpus_maker.make_corpus()
parser = PlaintextParser.from_string(" ".join(corpus),
Tokenizer(language))

# Call the summarization algorithm and do the summarization
summarizer = LexRankSummarizer()
summarizer.stop_words = get_stop_words(language)
summary = summarizer(document=parser.document,
                      sentences_count=len(corpus) * 2 // 10)

return summary
```

[OceanofPDF.com](http://OceanofPDF.com)

## 45. Document Word Detection

Detect a word present in the document (pages) using OpenCV

## Package

1. **OpenCV** : `pip install opencv-python``
2. **Numpy** : `pip install numpy``
3. **Imutils** : `pip install imutils``

<table>

<tr>

<td align="center"><b>Input image</b></td>

<td align="center"><b>Output image</b></td>

</tr>

<tr>

<td><a href="https://postimg.cc/2qSY8dWD" target="\_blank"></a><br/><br/>

</td>

<td><a href="https://postimg.cc/18HPQ7fB" target="\_blank"></a><br/><br/>

</td>

</tr>

</table>

## Source Code:

```
'''
output : word detection on document (Simple OCR type of application)
'''

import cv2
import numpy as np
import imutils

# frame read
frame = cv2.imread('test.jpeg')

# resize
frame = cv2.resize(frame, (600, 600))

# grayscale
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

# remove noise
blur = cv2.GaussianBlur(gray, (5, 5), 0)

# otsu thresh (bimodel threshold)
thresh = cv2.threshold(blur, 0, 255,
                        cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]

# get structuring element

horizontal_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (25, 1))
vertical_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (1, 25))
print('horizontal kernel : {}'.format(horizontal_kernel))
```



```

print('vertical kernel : {}'.format(vertical_kernel))

# opening (erosion followed by dilation)

horizontal_lines = cv2.morphologyEx(thresh,
                                     cv2.MORPH_OPEN,
                                     horizontal_kernel,
                                     iterations=2)
vertical_lines = cv2.morphologyEx(thresh,
                                  cv2.MORPH_OPEN,
                                  vertical_kernel,
                                  iterations=2)

# contours apply on detected lines
# First one is source image, second is contour retrieval mode, third is contour approximation
method

cnts = cv2.findContours(horizontal_lines, cv2.RETR_EXTERNAL,
                        cv2.CHAIN_APPROX_SIMPLE)
cntsv = cv2.findContours(vertical_lines, cv2.RETR_EXTERNAL,
                          cv2.CHAIN_APPROX_SIMPLE)

# find contours
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
cntsv = cntsv[0] if len(cntsv) == 2 else cntsv[1]

for c in cnts:
    cv2.drawContours(frame, [c], -1, (255, 255, 255), 2)
for c in cntsv:
    cv2.drawContours(frame, [c], -1, (255, 255, 255), 2)

```

```
# imshow
cv2.imshow('thresh', thresh)
cv2.imshow('horizontal_lines', horizontal_lines)
cv2.imshow('vertical_lines', vertical_lines)
cv2.imshow('frame', frame)

# grayscale

gray1 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
thresh1 = cv2.adaptiveThreshold(gray1, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                cv2.THRESH_BINARY, 23, 30)
canny = imutils.auto_canny(thresh1)

output = cv2.bitwise_not(canny)
kernel = np.ones((5, 5), np.uint8)

opening = cv2.morphologyEx(canny, cv2.MORPH_CLOSE, kernel)

dilation = cv2.dilate(canny, kernel, iterations=1)

contour, hierachy = cv2.findContours(dilation, cv2.RETR_TREE,
                                      cv2.CHAIN_APPROX_SIMPLE)

for i in contour:
    area = cv2.contourArea(i)
    if area > 20:
        x, y, w, h = cv2.boundingRect(i)
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 120, 255), 2)

cv2.imshow('output', output)
cv2.imshow('dilate', dilation)
```

```
cv2.imshow('opening', opening)
cv2.imshow('original_frame', frame)
cv2.imshow('canny', canny)
cv2.imshow('thresh1', thresh1)
```

```
# Saving output image
```

```
cv2.imwrite('output.jpg', frame)
```

```
# destroy all window
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

[OceanofPDF.com](http://OceanofPDF.com)

## 46. Finding Dominant Color

- This script will take a image and it will find dominant color in it .

### ### Prerequisites

- You only need to have installed opencv which is used for image preprocesssing.
- Run the below script to install opencv
- \$ pip install opencv-python

### ### How to run the script

- Run below command
- python find-color.py
- Now Enter the path for image
- Copy Your image path and enter in the command

### Requirements:

**opencv-python==4.3.0.36**

**numpy==1.19.1**

### Source Code:

```
import cv2
import numpy as np

path = input("Enter Path :- ")
try:
    img = cv2.imread(path)
    cv2.imshow("img", img)
except Exception:
    print("Path not found")
    exit()

array = np.array(img)
unique, counts = np.unique(array, return_counts=True)

ocurance = dict(zip(unique, counts))

a1_sorted_keys = sorted(ocurance, key=ocurance.get, reverse=True)
print(a1_sorted_keys[:3])
# Create a blank 300x300 black image
image = np.zeros((300, 300, 3), np.uint8)
# Fill image with red color(set each pixel to red)
```

```
image[:] = a1_sorted_keys[:3]
c = a1_sorted_keys[0]
# Create a blank 300x300 black image
color = np.zeros((300, 300, 3), np.uint8)
# Fill image with red color(set each pixel to red)
color[:] = (c, c, c)
print("Tone : " + str(a1_sorted_keys[:3]))
cv2.imshow("Tone", image)
print("color : " + str([c, c, c]))
cv2.imshow("color", color)
```

[OceanofPDF.com](http://OceanofPDF.com)

## 47. DOMINANT COLOR DETECTION FROM IMAGE

### ## Introduction

1. Finding Dominant Colour: Finding Colour with Kmeans and giving appropriate colors to the

pixel/data points of the image.

2. Analyse different K in K-means : Different Ks can give totally different results, and usually, there is one K which

is consistent with the data points, the actual K.

3. Good Segmentation(Colour Extraction) : Given an image in a generic viewpoint, we wish to produce the best possible

segmentation, by this we mean that most people would choose this segmentation if asked to make one, and it would

make a good precursor for object recognition.

### ## How to use it :

1. Download or clone the repository

2. Install Required Libraries

...

pip install python-opencv

pip install sklearn

pip install pandas

pip install matplotlib

OR

```
pip3 install -r requirements.txt
```

```

3. Run all the cells of Dominant color Extraction.ipynb

4. Output :Analyse the Dominant color in Image<br>

### **Requirements:**

**Python==3.8.5**

**jupyter==4.7.0**

**jupyter-notebook==6.1.4**

**pandas==1.1.5**

**cv2==4.4.0**

**matplotlib==3.3.3**

**sklearn==0.23.2**

### **Source Code File:**



Dominant color  
Extraction.ipynb



## 48. Download website Articles as pdf

<!--Remove the below lines and add yours -->

This script take a link of website article as input and download the complete article as a pdf at default download location.

### Prerequisites

<!--Remove the below lines and add yours -->

- \* selenium

- \* requests

- \* webdriver-manager

- \* Run `pip install -r requirements.txt` to install required external modules.

### How to run the script

<!--Remove the below lines and add yours -->

- Execute `python3 downloader.py`

- Type in URL of article when prompted.

**Requirements:**

**requests==2.24.0**

**selenium==3.141.0**

**webdriver-manager==3.2.2**

**Source Code:**

```
# !/usr/bin/env python
```

```
from selenium import webdriver
```

```
from webdriver_manager.chrome import ChromeDriverManager
import json
import requests

# article url
# URL = " https://www.nytimes.com/topic/company/amazoncom-inc

def get_driver():
    # chrome options settings
    chrome_options = webdriver.ChromeOptions()
    settings = {
        "recentDestinations": [
            {"id": "Save as PDF", "origin": "local", "account": ""}
        ],
        "selectedDestinationId": "Save as PDF",
        "version": 2,
    }
    prefs = {
        "printing.print_preview_sticky_settings.appState":
json.dumps(settings)
    }
    chrome_options.add_experimental_option("prefs", prefs)
    chrome_options.add_argument("--kiosk-printing")
```

```
# launch browser with predefined settings
browser = webdriver.Chrome(
    executable_path=ChromeDriverManager().install(),
    options=chrome_options
)
return browser
```

```
def download_article(URL):
```

```
    browser = get_driver()
    browser.get(URL)
```

```
# launch print and save as pdf
browser.execute_script("window.print();")
browser.close()
```

```
if __name__ == "__main__":
    URL = input("provide article URL: ")
    # check if the url is valid/reachable
    if requests.get(URL).status_code == 200:
        try:
            download_article(URL)
            print("Your article is successfully downloaded")
        except Exception as e:
            print(e)
```

else:

```
print("Enter a valid working URL")
```

[OceanofPDF.com](http://OceanofPDF.com)

## 49. Scrap images from URL

1. Download Chrome Drive From Chrome.
2. Run scrap-img.py file `py scrap-img.py`
3. `Enter Path : E:\webscraping\chromedriver\_win32\chromedriver.exe`  
<br/>  
`Enter URL : https://dribbble.com/`

### Requirements:

**selenium==3.141.0**

### Source Code:

```
from selenium import webdriver
import requests as rq
import os
from bs4 import BeautifulSoup
import time

# path= E:\web scraping\chromedriver_win32\chromedriver.exe
path = input("Enter Path : ")

url = input("Enter URL : ")

output = "output"
```

```
def get_url(path, url):  
    driver = webdriver.Chrome(executable_path=r"{}".format(path))  
    driver.get(url)  
    print("loading.....")  
    res = driver.execute_script("return  
document.documentElement.outerHTML")  
  
    return res
```

```
def get_img_links(res):  
    soup = BeautifulSoup(res, "lxml")  
    imglinks = soup.find_all("img", src=True)  
    return imglinks
```

```
def download_img(img_link, index):  
    try:  
        extensions = [".jpeg", ".jpg", ".png", ".gif"]  
    extension = ".jpg"  
    for exe in extensions:  
        if img_link.find(exe) > 0:  
            extension = exe  
            break
```

```
img_data = rq.get(img_link).content
with open(output + "\\" + str(index + 1) + extension, "wb+") as f:
    f.write(img_data)
```

```
f.close()
except Exception:
    pass
```

```
result = get_url(path, url)
time.sleep(60)
img_links = get_img_links(result)
if not os.path.isdir(output):
    os.mkdir(output)

for index, img_link in enumerate(img_links):
    img_link = img_link["src"]
    print("Downloading...")
    if img_link:
        download_img(img_link, index)
    print("Download Complete!!")
```

## 50. Duplicate Files Remover

This script removes duplicate files in the directory where the script runs.

### Prerequisites

- \* No external libraries are used
- \* os
- \* hashlib

### How to run the script

Execute `python3 duplicatefileremover.py`

## Working

The script first lists all the files in the directory. It takes MD5 hash of each file, when hash of 2 files become same it deletes the file.

**Source Code:**



```
import hashlib
```

```
import os
```

```
# Returns the hash string of the given file name
```

```
def hashFile(filename):
```

```
    # For large files, if we read it all together it can lead to memory  
    overflow, So we take a blocksize to read at a time
```

```
    BLOCKSIZE = 65536
```

```
    hasher = hashlib.md5()
```

```
    with open(filename, 'rb') as file:
```

```
        # Reads the particular blocksize from file
```

```
        buf = file.read(BLOCKSIZE)
```

```
        while(len(buf) > 0):
```

```
            hasher.update(buf)
```

```
            buf = file.read(BLOCKSIZE)
```

```
    return hasher.hexdigest()
```

```
if __name__ == "__main__":
```

```
    # Dictionary to store the hash and filename
```

```
    hashMap = {}
```

```
    # List to store deleted files
```

```
    deletedFiles = []
```

```
filelist = [f for f in os.listdir() if os.path.isfile(f)]
for f in filelist:
    key = hashFile(f)
    # If key already exists, it deletes the file
    if key in hashMap.keys():
        deletedFiles.append(f)
        os.remove(f)
    else:
        hashMap[key] = f
if len(deletedFiles) != 0:
    print('Deleted Files')
```

```
    for i in deletedFiles:
        print(i)
else:
    print('No duplicate files found')
```

## How to download this project:

As you are our special readers you deserve special privileges. Please download all this projects for further practise using following steps.

1. Goto - <https://www.edcredibly.com/s/store/courses/description/50-Python-Projects> which is our own website.
2. Apply coupon code – **SPECIAL** to make this course free available for you.
3. Checkout without paying anything and enrol.
4. Download the file and enjoy.

Cheers, Happy learning!!

[OceanofPDF.com](https://www.oceanofpdf.com)

## ABOUT THE AUTHOR

**“Edcorner Learning”** and have a significant number of students on **Udemy** with more than **90000+ Student and Rating of 4.1 or above.**

**Edcorner Learning is Part of Edcredibly.**

Edcredibly is an online eLearning platform provides Courses on all trending technologies that maximizes learning outcomes and career opportunity for professionals and as well as students. Edcredibly have a significant number of 100000+ students on their own platform and have a **Rating of 4.9 on Google Play Store – Edcredibly App.**

Feel Free to check or join our courses on:

**Edcredibly Website - <https://www.edcredibly.com/>**

**Edcredibly App – <https://play.google.com/store/apps/details?id=com.edcredibly.courses>**

**Edcorner Learning Udemy - <https://www.udemy.com/user/edcorner/>**

**Do check our other eBooks available on Kindle Store.**

[OceanofPDF.com](http://OceanofPDF.com)