



*PROJECT ON:* Python Programming

*TOPIC:* An Encryption and Decryption Tool

***NAME-* ARCHISMAN CHAKRABORTY**

***COLLEGE-* TECHNO INDIA SALT LAKE**

***WINTER TRAINING COURSE ON* INTRODUCTION TO PYTHON**

**INDEX**

Topic Page No.

1. Acknowledgement……………………………….
2. Introduction………………………………………
3. About…………..…………………………………
4. Project Description……………………………….
5. Source codes……………………………………...
6. Output Screenshot………………………………...
7. Conclusion………………………………………..
8. Bibliography………………………………………

Acknowledgement

I acknowledge the help and support rendered to me by our instructor, Abhisek Karmakar, and Sreejit Dey, in completion of this project.

**Introduction**

The Oxford English Dictionary describes the process of Encryption as:

“the process of converting information or data into a code, especially to prevent unauthorized access.”

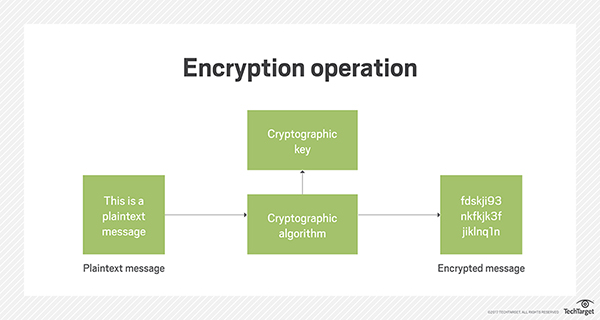
In [cryptography](https://en.wikipedia.org/wiki/Cryptography), **encryption** is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot. Encryption does not itself prevent interference, but denies the intelligible content to a would-be interceptor. In an encryption scheme, the intended information or message, referred to as [plaintext](https://en.wikipedia.org/wiki/Plaintext), is encrypted using an encryption algorithm – a [cipher](https://en.wikipedia.org/wiki/Cipher) – generating [cipher text](https://en.wikipedia.org/wiki/Ciphertext) that can only be read if decrypted. For technical reasons, an encryption scheme usually uses a [pseudo-random](https://en.wikipedia.org/wiki/Pseudo-random) encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, considerable computational resources and skills are required. An authorized recipient can easily decrypt the message with the [key](https://en.wikipedia.org/wiki/Key_(cryptography)) provided by the originator to recipients but not to unauthorized users.

In computing, encryption is the method by which plaintext or any other type of data is converted from a readable form to an encoded version that can only be decoded by another entity if they have access to a decryption key. Encryption is one of the most important methods for providing data security, especially for end-to-end protection of data transmitted across networks.

Encryption is widely used on the internet to protect user information being sent between a browser and a server, including passwords, payment information and other personal information that should be considered private. Organizations and individuals also commonly use encryption to protect sensitive data stored on computers, servers and mobile devices like phones or tablets.

***How encryption works:***

Unencrypted data, often referred to as [plaintext](http://searchsecurity.techtarget.com/definition/plaintext), is encrypted using an encryption [algorithm](http://whatis.techtarget.com/definition/algorithm) and an encryption [key](http://searchsecurity.techtarget.com/definition/key). This process generates [cipher text](http://whatis.techtarget.com/definition/ciphertext) that can only be viewed in its original form if decrypted with the correct key. Decryption is simply the inverse of encryption, following the same steps but reversing the order in which the keys are applied. Today's most widely used encryption algorithms fall into two categories:



1. Symmetric/Private Key: In [symmetric-key](https://en.wikipedia.org/wiki/Symmetric-key_algorithm) schemes, the encryption and decryption keys are the same. Communicating parties must have the same key in order to achieve secure communication.
2. Asymmetric Key: In [public-key encryption](https://en.wikipedia.org/wiki/Public-key_encryption) schemes, the encryption key is published for anyone to use and encrypt messages.

**About**

# What is Python? Executive Summary

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python History

* Python laid its foundation in the late 1980s.
* The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
* In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
* In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
* Python 2.0 added new features like: list comprehensions, garbage collection system.
* On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
* *ABC programming language* is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
* Python is influenced by following programming languages:
  + ABC language.
  + Modula-3

# Python Applications Area

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development. Here, we are specifying applications areas where python can be applied.

*1) Web Applications:* We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and delelop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

#### 2) Desktop GUI Applications: Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

#### 3) Software Development: Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

#### 4) Scientific and Numeric: Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

#### 5) Business Applications: Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.

#### 6) Console Based Application: We can use Python to develop console based applications. For example: IPython.

#### 7) Audio or Video based Applications: Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

#### 8) 3D CAD Applications: To create CAD application Fandango is a real application which provides full features of CAD.

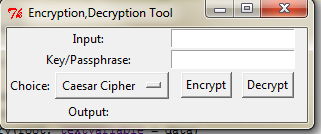
#### 9) Enterprise Applications: Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

#### 10) Applications for Images: Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc. There are several such applications which can be developed using Python

**Project Description**

In this specific project, we decided to make an Encryption/Decryption Tool using Python (version 2.7.14) and it’s pre-defined modules. We used the python module Tkinter to make the User Interface. In this encryption/decryption tool we provide the user with 3 choices of encryption techniques, namely:

1. Caesar Cipher (a substitution cipher)
2. Base64 Encryption
3. Advanced Encryption Standards (A.E.S.)



This is the User Interface for the Encryption/Decryption Tool that we designed. As can be seen, the Input text field accepts the original text to be encrypted or decrypted. The Key/Passphrase field takes in the Key (for Caesar Cipher), and Passphrase (for AES Encryption). The Drop-Down List gives the choice to select any one of the three techniques. And finally, the end result is displayed alongside the Output label.

For the encryption techniques we designed three Python files, one for each technique, which contained a class and two methods, namely: encrypt() and decrypt(). Then another file was created to contain the code for the UI and the choice menu. It also provides the function of calling the appropriate functions according to User’s choice.

The necessary algorithms of the three Encryption techniques are as follows:

**1.Caesar Cipher:** It’s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.  
Thus to cipher a given text we need an integer value, known as shift which indicates the number of position each letter of the text has been moved down.  
The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as:

E(x) = ( x + n ) mod 26 (Encryption Phase with shift n)

E(x) = ( x - n ) mod 26 (Decryption Phase with shift n)

**Algorithm:**

**Input:**

1. A String of lower case letters, called Text.
2. An Integer between 0-25 denoting the required shift.

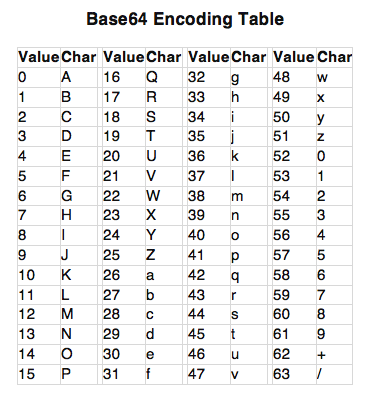
**Procedure:**

* Traverse the given text one character at a time .
* For each character, transform the given character as per the rule, depending on whether we’re encrypting or decrypting the text.
* Return the new string generated.

**2.Base64:** Base64, also known as MIME encoding, translates binary into safe text. It is used to send attachments in email and to change small bits of unsafe high-character data into stuff that is a lot nicer for text-based system.

Base64 encoding takes the original binary data and operates on it by dividing it into tokens of three bytes. A byte consists of eight bits, so Base64 takes 24bits in total. These 3 bytes are then converted into four printable characters from the ASCII standard.

The first step is to take the three bytes (24bit) of binary data and split it into four numbers of six bits. Because the ASCII standard defines the use of seven bits, Base64 only uses 6 bits (corresponding to 2^6 = 64 characters) to ensure the encoded data is printable and none of the special characters available in ASCII are used. The algorithm's name Base64 comes from the use of these 64 ASCII characters. The ASCII characters used for Base64 are the numbers 0-9, the alphabets 26 lowercase and 26 uppercase characters plus two extra characters '+' and '/'.



**Our Algorithm:** We made use of the base64 in-built Python module to solve the task. To encrypt we used the base64.encodestring(String) method, and the base64.decodestring(String) method to decrypt the texts.

**3.Advanced Encryption Standards:** The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

## Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below −



### Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

### Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that ‘fall off’ are re-inserted on the right side of row.

### MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

### Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

## Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse

**Our Algorithm:** Since Python already has a pre-defined module for these advanced encryption processes, we made use of it. We used the ***PyCrypto*** Python module and its python file AES.py to encrypt the given data.

**(i)**First, we passed the passphrase and a randomized initialization vector as the arguments to create an ***AES*** object.

**(ii)**Then we passed the original text as an argument to the .encrypt(String) method to encrypt and .decrypt(String) of the ***Crypto.Cipher.AES*** file to decrypt the given Data.

**Source Codes**

**File name: CaesarCipher.py**

**class** CaesarCipher:  
 **def** encrypt(self, key, data):  
 newData = **""  
  
 for** char **in** data:  
 ascii = ord(char)  
 **if** ascii == 32:  
 newData += **" "  
 continue  
 elif** ascii <= 90:  
 newData += unichr(((ascii - 65) + key) % 26 + 65)  
 **else**:  
 newData += unichr(((ascii - 97) + key) % 26 + 97)  
  
 **return** newData  
  
 **def** decrypt(self, key, data):  
 newData = **""  
 for** char **in** data:  
 ascii = ord(char)  
 **if** ascii == 32:  
 newData += **" "  
 continue  
 elif** ascii <= 90:  
 newData += unichr(((ascii - 65) - key) % 26 + 65)  
 **else**:  
 newData += unichr(((ascii - 97) - key) % 26 + 97)  
  
 **return** newData

**Variable Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Data Type** | **Scope** | **Use** |
| Key | Int | encrypt(), decrypt() | To store the Key |
| Data | String | encrypt(), decrypt() | To store the input String |
| newData | String | encrypt(), decrypt() | To store the output String |
| Char | String | encrypt(), decrypt()  for loop | To store each character of the data |
| Ascii | Int | encrypt(), decrypt()  for loop | To store the ascii code of each char |

**Method Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Parameters** | **Return Value** | **Function** |
| encrypt() | key, data | String | To encrypt the plain Text |
| decrypt() | key, data | String | To decrypt the cipher Text |

**File Name: Base64Cipher.py**

**import** base64  
**class** Base64Cipher:  
 **def** encrypt(self, data):  
 newData = base64.encodestring(data)  
 **return** newData  
  
 **def** decrypt(self, data):  
 newData = base64.decodestring(data)  
 **return** newData

**Variable Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Data Type** | **Scope** | **Use** |
| data | Int | encrypt(), decrypt() | To store the input String |
| newData | String | encrypt(), decrypt() | To store the output String |

**Method Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Parameters** | **Return Value** | **Function** |
| encrypt() | data | String | To encrypt the plain Text |
| decrypt() | data | String | To decrypt the cipher Text |

**File Name:** **AESCipher.py**

**from** Crypto.Cipher **import** AES  
**import** os, binascii  
  
**class** AESCipher:  
  
 **def** encrypt(self, message, passphrase):  
 iv = os.urandom(16)  
 aes = AES.new(passphrase, AES.MODE\_CBC, iv)  
 cipherText = aes.encrypt(message)  
 **return** cipherText  
  
 **def** decrypt(self, cipherText, passphrase):  
 iv = os.urandom(16)  
 aes = AES.new(passphrase, AES.MODE\_CBC, iv)  
 message = aes.decrypt(cipherText)  
 **return** message

**Variable Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Data Type** | **Scope** | **Use** |
| message | String | encrypt(), decrypt() | To store the original message |
| passphrase | String | encrypt(), decrypt() | To store the passcode for encryption/decryption |
| iv | String | encrypt(), decrypt() | To store the initialization Vector |
| aes | Object of class AES | encrypt(), decrypt() | To call the methods of AES class |
| cipherText | String | encrypt(), decrypt() | To store cipher text(encrypted) |

**Method Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Parameters** | **Return value** | **Function** |
| encrypt() | message, passphrase | String | To encrypt the message |
| decrypt() | cipherText, passphrase | String | To decrypt the Cipher Text |

**File Name: ToolGUI.py**

**from** Tkinter **import** \*  
**from** CaesarCipher **import** \*  
**from** Base64Cipher **import** \*  
**from** AESCipher **import** \*  
  
*#objects*cc = CaesarCipher()  
b64 = Base64Cipher()  
aes = AESCipher()  
  
root = Tk(className=**"/Encryption,Decryption Tool"**) *# main window  
#variables*data = StringVar()  
key = StringVar()  
var = StringVar()  
*#input data*inputLabel = Label(text = **'Input:'**)  
inputLabel.grid(row = 0, column = 0)  
inputBox = Entry(root, textvariable = data)  
inputBox.grid(row = 0, column = 1)  
  
*#key/passphrase input*keyLabel = Label(text = **'Key/Passphrase:'**)  
keyLabel.grid(row = 1, column = 0)  
keyBox = Entry(root, textvariable = key)  
keyBox.grid(row = 1, column = 1)  
  
*#output data*outputLabel = Label(text = **'Output:'**)  
outputLabel.grid(row = 3, column = 0)  
outputBox = Label(root)  
outputBox.grid(row = 3, column = 1)  
  
*#User choice*fch = Frame(root)  
chlb = Label(fch, text = **'Choice:'**)  
chlb.grid(row = 0, column = 0)  
choices = [**'Caesar Cipher'**, **'Base64 Cipher'**, **'AES Encryption'**]  
menu = OptionMenu(fch, var, \*choices)  
menu.grid(row = 0, column = 1)  
fch.grid(row = 2, column = 0)  
  
**def** encrypt():  
 cipherText = **''** text = data.get()  
 option = var.get()  
 **if** option == **'Caesar Cipher'**:  
 k = int(key.get())  
 cipherText = cc.encrypt(k, text)  
 **print 'Caesar'**,  
 **elif** option == **'Base64 Cipher'**:  
 cipherText = b64.encrypt(text)  
 **print 'B64'**,  
 **elif** option == **'AES Encryption'**:  
 passphrase = key.get()  
 cipherText = aes.encrypt(text, passphrase)  
 **print 'AES'**,  
 **print "Encrypting"** outputBox.config(text = cipherText)  
  
**def** decrypt():  
 message = **''** text = data.get()  
 option = var.get()  
 **if** option == **'Caesar Cipher'**:  
 k = int(key.get())  
 message = cc.decrypt(k, text)  
 **print 'Caesar'**,  
 **elif** option == **'Base64 Cipher'**:  
 message = b64.decrypt(text)  
 **print 'B64'**,  
 **elif** option == **'AES Encryption'**:  
 passphrase = key.get()  
 message = aes.decrypt(text, passphrase)  
 **print 'AES'**,  
 **print "Decrypting"** outputBox.config(text = message)

*#buttons*fb = Frame(root)  
encrypt = Button(fb, text = **'Encrypt'**, command = encrypt)  
encrypt.grid(row = 0, column = 0,padx = 10)  
decrypt = Button(fb, text = **'Decrypt'**, command = decrypt)  
decrypt.grid(row = 0, column = 1)  
fb.grid(row = 2, column = 1)  
root.mainloop()

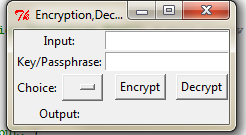
**Variable Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Data Type** | **Scope** | **Use** |
| cc | Object(CaesarCipher) | Global | To store the Key |
| aes | Object(AESCipher) | Global | To store the input String |
| b64 | Object(Base64Cipher) | Global | To store the output String |
| data | String | Global | To store input String |
| key | String | Global | To store key of encryption |
| root | Object(Tkinter) | Global | To create main window |
| inputLabel | Object(Tkinter.Label) | Global | To create input Label |
| inputBox | Object(Tkinter.Entry) | Global | To create input entry field |
| keyLabel | Object(Tkinter.Label) | Global | To create key Label |
| keyBox | Object(Tkinter.Entry) | Global | To create key input field |
| outputLabel | Object(Tkinter.Label) | Global | To create output Label |
| outputBox | Object(Tkinter.Label) | Global | To create output field |
| chlb | Object(Tkinter.Label) | Global | To create user choice label |
| menu | Object(Tkinter.OptionMenu) | Global | To create Drop Down List |
| choices | String(list) | Global | To store menu data |
| var | String | Global | To store menu choice result |
| fch | Object(Tkinter.Frame) | Global | To hold the user choice menu |
| encrypt | Object(Tkinter.Button) | Global | To create the encrypt button |
| decrypt | Object(Tkinter.Button) | Global | To create the Decrypt button |
| fb | Object(Tkinter.Frame) | Global | To hold the two buttons |
| text | String | Global |  |
| option | String | encrypt(), decrypt() | To store user choice |
| k | Int | encrypt(), decrypt() | To store key |
| passphrase | String | encrypt(), decrypt() | To store passcode |
| message | String | decrypt() | To store original text |
| cipherText | String | encrypt() | To store encrypted text |

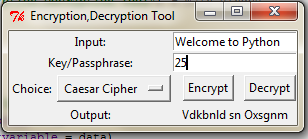
**Method Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Parameters** | **Return Type** | **Function** |
| encrypt() | None | None | To encrypt given data according to user’s Choice |
| decrypt() | None | None | To Decrypt cipher Text |

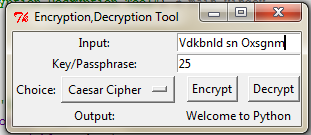
**Output Screenshots**



(a)User Interface on Startup



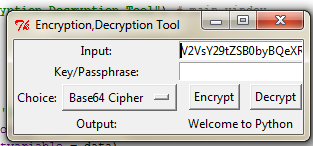
(b)UI for Caesar Cipher (i)Encryption



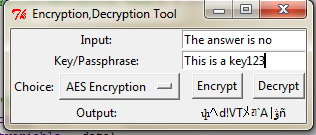
(ii)Decryption



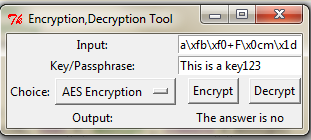
(c)UI for Base64 Cipher (i)Encryption



(ii)Decryption



(d)UI for AES Encryption (i)Encryption



(ii)Decryption(given input is hex code)(passphrase is kept same as before)

**Conclusion**

**Bibliography**

1. <https://www.python.org/doc/> : Python Docs
2. <https://stackoverflow.com/>: Stackoverflow
3. <https://github.com/>: Github
4. <https://en.wikipedia.org/wiki/Encryption>
5. <https://en.wikipedia.org/wiki/Base64>
6. <https://en.wikipedia.org/wiki/Advanced_Encryption_Standard>