Day - 8

**Function Parameters**

**&**

**Caesar Cipher**

Function with inputs

Function with arguments and parameters

**Encryption Project:** Caesar Cipher.

Shift a letter w.r.t given number. Eg. "a" is shifted to 3 becomes "d"

**8.1 Functions with parameters**

In below ***naMe*** is the parameter and ***"Bijo"*** is the argument

**def** **greet**(naMe):

**print**("Hi ", naMe)

**print**("How")

**print**("Are You")

**greet**("Bijo")

#*python func\_input.py*

* Function with multiple parameters: Following example illustrates the idea.

#*function - multipe params*

**def** **greet**(naMe, cTy):

**print**("Hi ", naMe)

**print**(f"What is like in {cTy}")

**greet**("Bijo", "Dhaka")

**8.2 *POSITIONAL* arguments and *KEYWORD* arguments**

* Positional arguments: In the " POSITIONAL arguments " arguments must match to the position of the parameters it’s the classic case. For example, if we interchange the position

**greet**("Bijo", "Dhaka") and **greet**("Dhaka", "Bijo")

Above function calls give DIFFERENT results.

* Keyword arguments: In the case of "Keyword arguments", a ***key-value pair*** is used. Hence the position of arguments doesn’t matter, it automatically matches to the key of the parameters. For example, if we interchange the position the ***key-value*** pair:

**greet**(naMe="Silvi", cTy="NY")

**greet**(cTy="NY", naMe="Silvi")

above function calls give SAME results.

* Exercise 8.1: Painting wall.

You are painting a wall. The instructions on the paint can says that 1 can of paint can cover 5 square meters of wall. Given a random ***height*** and ***width*** of wall, calculate how many cans of paint you'll need to buy.

**import** math

test\_h = **int**(input("Height of wall: "))

test\_w = **int**(input("Width of wall: "))

coverage = 5

**def** **paint\_calc**(height, width, cover) :

    caNumber = math**.ceil**((height\*width)/cover)

**print**(f"\n\t To paint your wall you need {caNumber} can paint")

**paint\_calc**(height=test\_h, width=test\_w, cover=coverage)

#*python wall\_paint.py*

* You need also use the ceiling method ***math.ceil()*** and ***import math***
* Exercise 8.2: Prime numbers in a given range.

number = **int**(input("Enter the number: "))

**def** **prMchk**(num) :

    is\_prm = **True**

    fc = 0

**for** i **in** **range**(2, **int**(num/2)):

**if** (num % i) **==** 0 :

            is\_prm = **False**

            fc = i

**if** is\_prm **==** **True** :

**print**(f"\t{num} is a prime number")

**else** :

**print**(f"\t{num} isn't prime. Divisible by {fc}")

**for** k **in** **range**(2, number+1):

**prMchk**(k)

#*python prime.py*

* Exercise 8.3: Caesar Cipher project.

Step 1: ENCODE:

#TODO-1: Create a function called 'encrypt' that takes the 'text' and 'shift' as inputs.

#TODO-2: Inside the 'encrypt' function, shift each letter of the 'text' forwards in the alphabet by the shift amount and print the encrypted text.

#e.g.

#plain\_text = "hello"

#shift = 5

#cipher\_text = "mjqqt"

#print output: "The encoded text is mjqqt"

|  |
| --- |
| Instructor |
| alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']  direction = **input**("Type 'encode' to encrypt, type 'decode' to decrypt:\n")  text = **input**("Type your message:\n")**.lower**()  shift = **int**(input("Type the shift number:\n"))  **def** **encrypt**(plain\_text, shift\_amount):      cipher\_text = ''  **for** letter **in** plain\_text :          position = alphabet**.index**(letter)          new\_position = position + shift\_amount          new\_letter = alphabet[new\_position]          cipher\_text += new\_letter    **print**(f"The Encoded text is {cipher\_text}")  **encrypt**(plain\_text = text, shift\_amount = shift)  #*python caeser\_cipher.py* |

Practice

**def** **encrypt**(text, shift):

    cipher\_text = ''

**for** ch **in** text :

        ch\_indx = **int**(alphabet**.index**(ch))

        cipher\_text += alphabet[ch\_indx + shift]

**print**(f"The Encoded text is {cipher\_text}")

**encrypt**(text, shift)

* Index of an element:

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

**print**(alphabet**.index**('g'))

**def** **index**(object: \_T, start: int=..., stop: int=...)

* Return ***first*** ***index*** of value.
* Raises ***ValueError*** if the value is not present.
* What if index is greater than 25?

2 Solutions:

**if** new\_position **>** 25 :

            new\_position -= 25

or

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

Extending "***alphabet***" is good idea because it reduces complexity when we ***decode***.

Step 2: DECODE:

#T0D0-1: Create a different function called '***decryp***t' that takes the '***text***' and '***shift***' as inputs.

#T0D0-2: Inside the '***decrypt***' function, shift each letter of the '***text***'\*backwards\* in the alphabet by the shift amount and print the decrypted text.

#e.g.

#cipher\_text = "mjqqt"

#shift = 5

#plain\_text = "hello"

#print output: "The decoded text is hello"

#T0D0-3: Check if the user wanted to ***encrypt*** or ***decrypt*** the message by checking the ***'direction'*** variable. Then call the correct function based on that ***'drection'*** variable. You should be able to test the code to encrypt.

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

direction = **input**("Type 'encode' to encrypt, type 'decode' to decrypt:\n")**.lower**()

text = **input**("Type your message:\n")**.lower**()

shift = **int**(input("Type the shift number:\n"))

#*--------------- Encode ------------*

**def** **encrypt**(plain\_text, shift\_amount):

    cipher\_text = ''

**for** letter **in** plain\_text :

        position = alphabet**.index**(letter)

        new\_position = position + shift\_amount

        new\_letter = alphabet[new\_position]

        cipher\_text += new\_letter

**print**(f"The Encoded text is {cipher\_text}")

#*------------ Decode ------------*

**def** **decrypt**(cipher\_text, shift\_amount):

    plain\_text = ''

**for** letter **in** cipher\_text :

        position = alphabet**.index**(letter)

        new\_position = position - shift\_amount

**if** new\_position **<** 0:

            new\_position += 26

        plain\_text += alphabet[new\_position]

**print**(f"The Decoded text is {plain\_text}")

**if** direction **==** "encode":

**encrypt**(plain\_text = text, shift\_amount = shift)

**elif** direction **==** "decode":

**decrypt**(cipher\_text = text, shift\_amount = shift)

#*python caeser\_cipher.py*

Step 3: Re-FACTORING:

#T0D0-1: Combine the ***encrypt()*** and ***decrypt()*** functions into a single function called ***caesar()***.

#T0D0-2: Call the ***caesar()*** function, passing over the 'text', 'shift' and 'direction' values.

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

direction = **input**("Type 'encode' to encrypt, type 'decode' to decrypt:\n")**.lower**()

text = **input**("Type your message:\n")**.lower**()

shift = **int**(input("Type the shift number:\n"))

|  |  |
| --- | --- |
| Practice | Instructor |
| #*--------------- Encode/Decode ------------*  **def** **caeser**(plain\_text, shift\_amount, dirCtn):      new\_text = ''  **for** letter **in** plain\_text :          position = alphabet**.index**(letter)  **if** dirCtn **==** "encode":              new\_position = position + shift\_amount  **if** new\_position **>** 25:                  new\_position -= 26  **elif** dirCtn **==** "decode":              new\_position = position - shift\_amount  **if** new\_position **<** 0:                  new\_position += 26          new\_text += alphabet[new\_position]    **print**(f"The {dirCtn}d text is {new\_text}") | #*--------------- Encode/Decode ------------*  **def** **caeser**(plain\_text, shift\_amount, dirCtn):      new\_text = ''  **if** dirCtn **==** "decode":          shift\_amount \*= -1  **for** letter **in** plain\_text :          position = alphabet**.index**(letter)          new\_position = position + shift\_amount  **if** new\_position **>** 25:              new\_position -= 26  **elif** new\_position **<** 0:              new\_position += 26          new\_text += alphabet[new\_position]    **print**(f"The {dirCtn}d text is {new\_text}") |

**caeser**(plain\_text = text, shift\_amount = shift, dirCtn = direction)

#*python caeser\_cipher.py*

Step 4: User-Experience:

#T0D0-1: Import and print the logo from ***caesar\_cipher\_art.py*** when the program starts.

#T0D0-2: What if the user enters a shift that is greater than the number of letters in the alphabet?

#Try running the program and entering a shift number of 45.

#Hint: Think about how you can use the modulus (%).

#T0D0-3: What happens if the user enters a number/symbol/space?

#Can you fix the code to keep the number/symbol/space when the text is encoded/decoded?

#e.g. start\_text = "meet me at 3"

#end text = "... .... ... 3 "

#T0D0-4: Can you figure out a way to ask the user if they want to restart the cipher program? #e.g. Type 'yes' if you want to go again. Otherwise type 'no'.|

#If they type 'yes' then ask them for the direction/text/shift again and call the caesar() function again?

#Hint: Try creating a new function that calls itself if they type 'yes'.

Practice

**import** caesar\_cipher\_art

**print**(caesar\_cipher\_art**.**logo)

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

direction = **input**("Type 'encode' to encrypt, type 'decode' to decrypt:\n")**.lower**()

text = **input**("Type your message:\n")**.lower**()

shift = **int**(input("Type the shift number:\n"))

#*--------------- Encode/Decode ------------*

**def** **caeser**(plain\_text, shift\_amount, dirCtn):

    new\_text = ''

**if** shift\_amount **>** 25:

        shift\_amount %= 26

**if** dirCtn **==** "decode":

        shift\_amount \*= -1

**for** letter **in** plain\_text :

**if** letter **in** alphabet:

            position = alphabet**.index**(letter)

            new\_position = position + shift\_amount

**if** new\_position **>** 25:

                new\_position -= 26

**elif** new\_position **<** 0:

                new\_position += 26

            new\_text += alphabet[new\_position]

**else**:

            new\_text += letter

**print**(f"The {dirCtn}d text is {new\_text}")

**caeser**(plain\_text = text, shift\_amount = shift, dirCtn = direction)

**while** **input**("Type 'yes' to continiue. To teminate enter any key : ")**.lower**() **==** "yes":

    direction = **input**("Type 'encode' to encrypt, type 'decode' to decrypt:\n")**.lower**()

    text = **input**("Type your message:\n")**.lower**()

    shift = **int**(input("Type the shift number:\n"))

**caeser**(plain\_text = text, shift\_amount = shift, dirCtn = direction)

**print**("Goodbye")

#*python caeser\_cipher.py*

Instructor

caesar\_cipher\_art.py

logo = """

 ,adPPYba, ,adPPYYba,  ,adPPYba, ,adPPYba, ,adPPYYba, 8b,dPPYba,

a8"     "" ""     `Y8 a8P\_\_\_\_\_88 I8[    "" ""     `Y8 88P'   "Y8

8b         ,adPPPPP88 8PP"""""""  `"Y8ba,  ,adPPPPP88 88

"8a,   ,aa 88,    ,88 "8b,   ,aa aa    ]8I 88,    ,88 88

 `"Ybbd8"' `"8bbdP"Y8  `"Ybbd8"' `"YbbdP"' `"8bbdP"Y8 88

            88             88

           ""             88

                          88

 ,adPPYba, 88 8b,dPPYba,  88,dPPYba,   ,adPPYba, 8b,dPPYba,

a8"     "" 88 88P'    "8a 88P'    "8a a8P\_\_\_\_\_88 88P'   "Y8

8b         88 88       d8 88       88 8PP""""""" 88

"8a,   ,aa 88 88b,   ,a8" 88       88 "8b,   ,aa 88

 `"Ybbd8"' 88 88`YbbdP"'  88       88  `"Ybbd8"' 88

              88

              88

"""

main.py

**import** caesar\_cipher\_art

**print**(caesar\_cipher\_art**.**logo)

alphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

#*--------------- Encode/Decode ------------*

**def** **caeser**(plain\_text, shift\_amount, dirCtn):

    new\_text = ''

**if** dirCtn **==** "decode":

        shift\_amount \*= -1

**for** letter **in** plain\_text :

**if** letter **in** alphabet:

            position = alphabet**.index**(letter)

            new\_position = position + shift\_amount

**if** new\_position **>** 25:

                new\_position -= 26

**elif** new\_position **<** 0:

                new\_position += 26

            new\_text += alphabet[new\_position]

**else**:

            new\_text += letter

**print**(f"The {dirCtn}d text is {new\_text}")

cont\_state = **True**

**while** cont\_state **==** **True**:

    direction = **input**("Type 'encode' to encrypt, type 'decode' to decrypt:\n")**.lower**()

    text = **input**("Type your message:\n")**.lower**()

    shift = **int**(input("Type the shift number:\n"))

**if** shift **>** 25:

        shift %= 26

**caeser**(plain\_text = text, shift\_amount = shift, dirCtn = direction)

    user\_ask = **input**("Type 'yes' to continiue. To teminate enter any key : ")**.lower**()

**if** user\_ask **==** "no":

        cont\_state = **False**

**print**("Goodbye")

#*python caeser\_cipher.py*