NAME

Caeser - Get a caeser cipher

SYNOPSIS

```
Ceaser.py -m "Hello" -k 12

Ceaser.py --message "Hello" --key 12

Ceaser.py -h
```

DESCRIPTION

Caesar cipher is to replace each plaintext letter with a different one a fixed number of places down the alphabet. This program Has a right shift default I.E. A becomes B with a fixed key of 1, you shift left with a negative key I.E. -1 B then becomes A

SEE ALSO

- Wiki (Caesar cipher)
- Wiki (Rot cipher)
- Github (apscandy)

BUGS

Beware of entering `@#` in the Cli as bash, terminal and powershell try to interpret `@#` and will brake the script

---

convert_to_ceasar

Converts a message to upper case and removes non alpha-characters

Pseudo code

- Convert text to uppercase
- Replaces fullstops with X
- Removes spaces
- Removes numbers
- Removes special characters

How to use

```
>>> message = "Hello"
>>> convert_to_ceasar(message)
'HELLO'
```

Parameters

message (str): message to be converted

Returns

message (str): converted string

```
def convert_to_ceasar(message:str)->str:
    message = re.sub("\.", "X", message)
    message = re.sub("[^A-Z]", "", message.upper())
    return str(message)
```

---

encrypt_caesar

Shifts message:str by using key:int to cipher message:str

Pseudo code

- Takes key and message parameter
- Shifts letters in message by the key number
- Makes translation table and uses key for shift value
- Translation table looks up new value then message is saved to new value
- Returns ciphered message

How to use

```
>>> message = "Hello"
>>> message = convert_to_ceasar(message)
>>> key = 13
>>> encrypt_caesar(key, message)
'URYYB'
```

Parameters

- key (int): A number between -26 and 26
- message (str): The message to be encoded/decoded

Returns

message (str): encoded/decoded message

```
def encrypt_caesar(key:int, message:str)->str:
  translation = str.maketrans(string.ascii_uppercase,
string.ascii_uppercase[key:] + string.ascii_uppercase[:key])
```

```
    message = message.translate(translation)
    return str(message)
```

---

main

Takes args from the CLI

how to use

```
Ceaser.py -m "Hello" -k 12
Message is: TQXXA
```

```
Ceaser.py --message "Hello" --key 12
Message is: TQXXA
```

```
Ceaser.py -h
usage: Ceaser.py [-h] -k KEY -m MESSAGE

Andrew's Cipher system

optional arguments:
  -h, --help            show this help message and exit
  -k KEY, --key KEY     int: Enter a number between -26 and 26 to
Encrypt/Decrypt
  -m MESSAGE, --message MESSAGE
                        str: Enter a message to be Encrypt/Decrypt
```

Parameters

- key (int): A number between -26 and 26
- message (str): message to be converted

Returns

message (str): converted string

```
def main()->str:
    parser = argparse.ArgumentParser(description="Andrew's Cipher system")
    parser.add_argument("-k", "--key",required=True ,type=int, help="int:
Enter a number between -26 and 26 to Encrypt/Decrypt")
    parser.add_argument("-m", "--message",required=True, type=str,
help="str: Enter a message to be Encrypt/Decrypt")
```

```
    args = parser.parse_args()
    key:int = args.key
    message:str = args.message
    key_check(key)
    message = convert_to_ceasar(message)
    message = encrypt_caesar(key, message)
    return message
```

Other functions

clear_terminal

Clears the terminal for a clean output

```
def clear_terminal()->None:
    if os.name == 'nt': os.system("cls")
    elif os.name == 'posix': os.system("clear")
```

key_check

check if key is in range -26 to 26

Parameters

key (int): A number between -26 and 26

```
def key_check(key:int)->None:
    if not -26 <= int(key) < 27:
        print("Please enter a key between -26 and 26\n")
        exit()
```

main_ut

Testing function for unittest module calls convert_to_ceasar and encrypt_caesar

Parameters

- key (int): A number between -26 and 26
- message (str): message to be converted

```
def main_ut(key:int, messsage:str)->str:
    messsage = convert_to_ceasar(messsage)
```

```
    messsage = encrypt_caesar(key, messsage)
    return messsage
```

Module

Only runs when `Ceaser.py` is called in the terminal

```
if __name__=="__main__":
    clear_terminal()
    output = main()
    print(f"Message is: {output}\n")
```