CS101. A7 - Isogram

Determine if a word or phrase is an isogram.

An isogram (also known as a "non-pattern word") is a word or phrase without a repeating letter, however spaces and hyphens are allowed to appear multiple times. Examples of isograms: lumberjacks, background, downstream, six-year-old, isogram.

The word isograms, however, is not an isogram, because the s repeats.

Input: A word or a phrase (String)
Output – True / False
    True (if the supplied input is an Isogram)
    False (if the supplied input is NOT an Isogram)

Examples:

- “” (Empty String)
- uncopyrightable (longest isogram)
- First# Clan! (string contains punctuation marks)
- “BackGround” (String contains upper- and lowercase letter)
- Non-isograms containing single letter repetition to several letters repeating.

Code Template (isogram.py)

```
"""
Isogram
"""

def is_isogram(string):
     pass
```

Sample output

```
>>> print(is_isogram(“”))
False
>>> print(square(4))
8
```

Test Cases

Function	**Inputs**	**Output**	**Remarks**
isogram	“”	True	Empty String
isogram	“isogram”	True	
isogram	“eleven”	False	
isogram	“zzyzx”	False	
isogram	“subdermatoglyphic”	True	
isogram	“Alphabet”	False	

isogram	“alphAbet”	False	
isogram	“thumbscrew-japingly”	True	
isogram	“Dictionary”	False	
isogram	“thumbscrew-jappingly”	False	
isogram	“six-year-old”	True	
isogram	“Emily Jung Schwartzkopf”	True	
isogram	“Accenture”	False	
isogram	“BackGround”	True	
isogram	“First# Clan!”	True	
isogram	“uncopyrightable”	True	

****

Write a function to return the sum of the digits of an integer. Raise a ValueError if any other datatype is input. Ignore the sign.

Examples

- sum_of_digits(10), sum_of_digits(10000000), sum_of_digits(010) will all return 1
- sum_of_digits(10.0) will `raise ValueError "Invalid input type. Expecting an integer."`

Tasks in the Assignment

- Implement one function: `sum_of_digits(number)`
- Input: an Integer.
- Output: Sum of the digits.

Code Template (sum_of_digits.py)

```
'''
Sum of digits
'''

def sum_of_digits(number):
     pass
```

Test Cases

Function	Inputs	Output	Remarks
sum_of_digits	-1.0	ValueError	
sum_of_digits	10	1	
sum_of_digits	34	7	
sum_of_digits	100	1	
sum_of_digits	10000	1	

****

CS101. A9 - Rain drops

Ref: https://en.wikipedia.org/wiki/Fizz_buzz

Your task is to convert a number into a string that contains raindrop sounds corresponding to certain potential factors. A factor is a number that evenly divides into another number, leaving no remainder. The simplest way to test if a one number is a factor of another is to use the modulo operation.

The rules of raindrops are that if a given number:

- has 2 as a factor, add ‘rim jhim’ to the result.
- has 3 as a factor, add ‘jal tarang’ to the result.
- has 5 as a factor, add ‘baadal’ to the result.
- has 7 as a factor, add ‘chalte hain' to the result.
- *does not* have any of 2, 3, 5, or 7 as a factor, the result should be the digits of the number.

Examples

- 7 has 7 as a factor and not 2, 3 or 5. Result: “chalte hain”
- 34 has 2 as a factor and is not factored by 3, 5, or 7, so the result would be "rim jhim".
- 15 has 3 and 5 as factors and not 2 or 7. Result: “jal tarang baadal”.
- 28 has 2 and 7 as factors, but not 3 or 5, so the result would be "rim jhim chalte hain".
- 30 has 2, 3 and 5 as factors, but not 7, so the result would be "rim jhim jal tarang chalte hain".
- 13 no factors in 2, 3, 5 or 7. Result “13”.
- Input is valid if it is a positive number >=2. Any other number given as input: `raise ValueError “Invalid input.”`

Tasks in the Assignment

- Implement one function: `convert(number)`
- Input: Integer
- Output: String.

Code Template (raindrops.py)

```
'''
Raindrops
'''

def convert(number):
     pass
```

Sample output

```
print(convert(34))
rim jhim

print(convert(210))
rim jhim jal tarang baadal chalte hain
```

Test Cases

Function	Inputs	Output	Remarks

convert	-1	ValueError	
convert	7	chalte hain	
convert	13	13	
convert	15	jal tarang baadal	
convert	28	rim jhim chalte hain	
convert	30	rim jhim jal tarang baadal	
convert	34	rim jhim	
convert	210	rim jhim jal tarang baadal chalte hain	

****

CS101. A10a - Fixed Substitution

Write a function to return the 5th english letter after the input.

Tasks to do
Implement the function f_substitute(letter)
letter (string): single letter from the english alphabet.
Return value: 5th letter from the input

Examples
- f_substitute(‘a’) returns ‘f’
- f_substitute(‘f’) returns ‘k’
- f_substitute(‘u’) returns ‘z’
- f_substitute(‘z’) returns ‘e’

Code Template (f_substitute.py)

```
'''
Fixed Substitute
'''

def f_substitute(letter):
     pass
```

Test Cases

Function	Inputs	Output	Remarks
f_substitute	‘a’	‘f’	
f_substitute	‘f’	‘k’	
f_substitute	‘u’	‘z’	
f_substitute	‘z’	‘e’	

****

CS101. A10b - Text Substitution

Write a function to return the nth english letter after the input.

Tasks to do
Implement the function substitute(letter, n)
letter (string): single letter from the english alphabet.
n (int): integer (can be negative)
Return value: nth letter from the input

Examples
- substitute(‘a’ , 0) returns ‘a’
- substitute(‘a’ , 1) returns ‘b’

- substitute(‘a’ , 25) returns ‘z’
- substitute(‘a’ , 26) returns ‘a’
- substitute(‘a’ , -1) returns ‘z’
- substitute(‘a’ , -25) returns ‘b’
- substitute(‘a’ , -26) returns ‘a’

Code Template (substitute.py)

```
'''
Substitute letter
'''

def substitute(letter, n):
     pass
```

Test Cases

Function	Inputs	Output	Remarks
substitute	‘a’ , 1	‘b’	
substitute	‘a’ 25	‘z’	
substitute	‘a’ , 26	‘a’	
substitute	‘a’ , -1	‘z’	
substitute	‘a’ , -25	‘b’	
substitute	‘a’ , -26	‘a’	
substitute	‘z’ , 1	‘a’	
substitute	‘z’ 25	‘y’	
substitute	‘z’ , 26	‘z’	
substitute	‘z’ , -1	‘y’	
substitute	‘z’ , -25	‘a’	
decrypt	‘z’ , -26	‘z’	

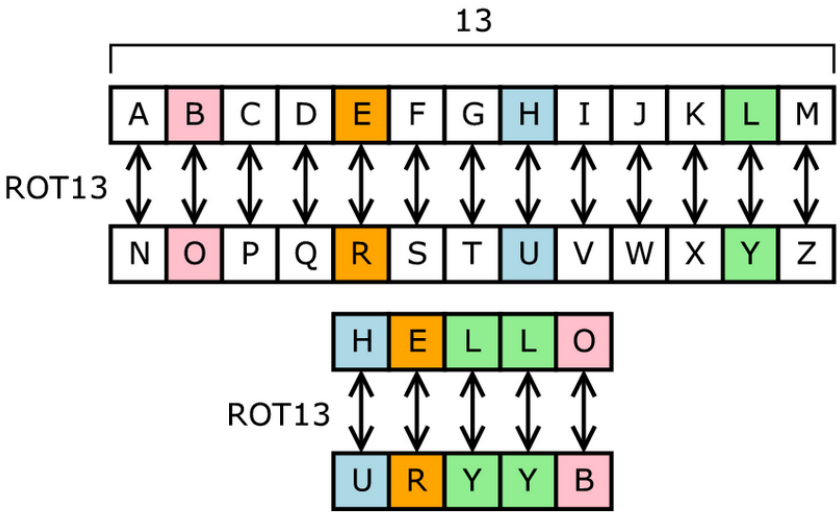
****

CS101. A10c - Encryption, Decryption through Substitution Cipher

Ref: http://en.wikipedia.org/wiki/Substitution_cipher

Encryption converts a string in plain english (known as *plaintext*), into another form known as *ciphertext*. Typically, *plaintext* is in plain readable English. The *ciphertext* is unintelligible and cannot be read without first decrypting it.

Substitution cipher is a method of encrypting in which units of plaintext are replaced with the ciphertext, in a well defined manner. Substitution can be done using a **key**. The encrypted message (*ciphertext*) can be read by the receiver after performing the inverse substitution process. The inverse substitution process is the decryption process here. Example process is shown in the picture below.



In the above example, plaintext is “HELLO” and the ciphertext is “URYYB”. The key here is ROT13. Details about ROT13 are below. The receiver receives “URYYB” and the key. Using the key, the receiver can decrypt the message and obtain the original message “HELLO”.

The Key
The ROT13 key can be written as “nnnnnnnnnnnnn”. Each position in the Key is the letter to be substituted with the character ‘a’ in that position. Examples.
Key is “aaaaa”. “hello” ==> “hello”
Key is “ddddd”. “hello” ==> “khoor”
Key is “nnnnn”. “hello” ==> “uryyb”
Key is “abcde”. “hello” ==> “hfnos”
Key is “abc”. “hello” ==> “hfnlp”. Key should be interpreted as “abcabcabc…” (long enough to encrypt the plaintext)

Tasks in the Assignment

- Implement two functions to achieve Substitution Cipher
    - encrypt() and decrypt()
- encrypt(plaintext, key)
    - Creates the ciphertext using key
    - Returns the ciphertext
- decrypt(ciphertext, key)
    - Reverses the ciphertext into the plaintext using key
    - Returns the plaintext

- Assume that the inputs do not have any characters apart from a-z (no upper case, no numerals, no punctuation marks)

Examples

- encrypt(plaintext, key="ddddddddddddddddddddd"):
  - plaintext: plain text to be encrypted
  - Key: key to use to encrypt. Keep default as ROT3.
- encrypt() implements Simple Shift Cipher using the key (Eg. Caesar Cipher)
- encrypt("dontlookup") should return "grqworrnxs"
- encrypt("dontlookup", "abcdefghij") should return "dppwpturcy"
- decrypt(ciphertext, key="ddddddddddddddddddddd"):
  - ciphertext: cipher text to be decrypted to plaintext
  - Key: key to use to encrypt. Default is ROT3.
- decrypt() implements Simple Shift Cipher using the key (Eg. Caesar Cipher)
- decrypt("grqworrnxs") should return "dontlookup"
- decrypt("dppwpturcy", "abcdefghij") should return "dontlookup"

Code Template (encrypt_decrypt.py)

```
'''
Substitution Cipher
Encrypt / Decrypt
'''

def encrypt(text, key="ddddddddddddddddddddd"):
     pass

def decrypt(text, key="ddddddddddddddddddddd"):
     pass
```

Sample output

```
>>> text="jurassicpark"
>>>key="ankylosaur"
>>>print(encrypt(text,key))
jhbydgacjrrx
>>>print("jhbydgacjrrx",key))
jurassicpark
```

Test Cases

Function	Inputs	Output	Remarks
encrypt	text="aaaaaaaaaa" key="aaaaaaaaaa"	aaaaaaaaaa	Empty String
decrypt	text="aaaaaaaaaa" key="aaaaaaaaaa"	aaaaaaaaaa	
encrypt	text="jurassicpark" key="ankylosaur"	jhbydgacjrrx	
decrypt	text="jhbydgacjrrx" key="ankylosaur"	jurassicpark	

encrypt	text="abcdefghij" key="abcdefghij"	acegikmoqs	
decrypt	text="acegikmoqs" key="abcdefghij"	abcdefghij	
encrypt	text="aaaaaaaaaa" key="abcdefghij"	abcdefghij	
decrypt	text="abcdefghij" key="abcdefghij"	aaaaaaaaaa	
encrypt	text="zzzzzzzzzz" key="abcdefghij"	zabcdefghi	
decrypt	text="zabcdefghi" key="abcdefghij"	zzzzzzzzzz	
encrypt	text="rainbowcolors" key="vibgyor"	mijtzcnxwmupg	
decrypt	text="mijtzcnxwmupg" key="vibgyor"	vibgyor	
encrypt	text="panthera" key="leo"	aebelsce	
decrypt	text="aebelsce" key="leo"	panthera	
encrypt	“hello”, “abc”	hfnlp	
decrypt	“hfnlp”, “abc”	hello	
encrypt	text=* (any string) key=""	raise ValueError(“Key cannot be empty.”)	
decrypt	text=* (any string) key=""	raise ValueError(“Key cannot be empty.”)	

****