Homework Assignment 3

- Responsible teaching assistant: Assaf Y. Livne.
- To get quick answers regarding this assignment please ask in Piazza under label HW3.
- The solution should be submitted by 2.6 23:30

String Encoding

General Instructions:

In this assignment, we will expand the default python string abilities. We will focus mostly on different encoding and decoding techniques. We will create a new string class that will have all the default abilities of the default python string class but will also have some encoding and decoding features. Because of copyright issues, we cannot use any built-in packages, therefore most of the code should be written from scratch. The solution has to be in one file, please see the attached prototype. However, don't limit yourself to only one class. Feel free to use classes or functions if you find them useful.
For the rest of this article, we will use ***str*** to call the default python string and the new class will be called **string**. **Literal strings are marked with Apostrophe** (').

Tasks:
A successful solution will include the features mentioned below. Don't feel obligated to code in this specific order. This section covers all the different tasks however a lot more details are explained ahead.

1. Build a class that behaves the same as str. You can limit yourself to only the following behaviors: slicing ([::]), indexing ([3]), comparison (==), concatenation of strings (+), repetition of strings (*), iterating over a string, obtaining the length of a string, count, isupper, islower.
2. Construct all the relevant exceptions.
3. Add a base64 encoding method. This method will return a string with the base64 representation of the original string.
4. Add a base64 decoding method. This method will return a string with the original string.
5. Add a cyclic chars encoding method. This method receives as input an integer that declares the value of the cyclic chars encoding. The method will return a string with the cyclic chars encoding of the original string.
6. Add a cyclic bits encoding method. This method receives as input an integer number that declares the value of the cyclic bits encoding. The method will return a string with the cyclic bits encoding of the original string.
7. Add a byte pair encoding method. This method will return a string with the byte pair encoding representation of the original string.
8. Add a histogram of chars feature. This method will return a dictionary.

Mandatory property:
you must implement a property called "rules". "rules" will be used for decoding the byte pair encoding, and it must be initialized as an empty list. You may use other properties if you like.

String Basics:
A string is a sequence of characters. There are many ways to store characters, but for this assignment, we will store characters as 8 bits binary numbers. The first 128 numbers (0-127) follow the ASCII coding table. For higher values of char (128-255) python will use the Unicode compression, but from your point of view you can leave it in its default form. For example, the decimal value of 97 is considered the 'a' char, while the decimal value of 167 is considered the '§' char.
We will only use the first 256 (2^8) chars. Pay attention that the *printable chars* are coded between 32- 126 while others with value lower then 128 are considered *control code chars*. **Our input will always be a string of the printable chars, but the output could have some control code chars or even chars with decimal values higher than 127.**

base64 Encoding - def base64(self)-> 'String'
Base64 is an encoding system that encodes a binary sequence into a string in a specific way. The binary sequence is first split into consecutive 6-bit pieces. Each piece is then matched to a predefined char according to a predefined table. The mapping between the values and the chars can be found in this link. Your task is to convert the string into a sequence of bits according to the ASCII table (example is given the above link as well). Then, split the binary sequence into consecutive 6-bit pieces, and convert them into chars using the base64 table. The base64 has several versions that differ about using padding chars (=). **In this work it is mandatory to not return the padding chars (=).**
Example:
For the string: '*Hello World*' the method will return the string '*SGVsbG8gV29ybGQ*'.

base64 Decoding - def decode_base64(self)-> 'String'
a decoding for a base64 string. If the string cannot be decoded this method will **raise a Base64DecodeError**
Example:
For the string: '*SGVsbG8gV29ybGQ*' the method will return the string '*Hello World*'.

Cyclic Chars Encoding - def cyclic_chars(self, num:int) -> 'String'
In this encoding, we Use a circular (or cyclical) shift for each char in order to get the result. In our case, we are **only considering the printable chars.** The code should shift between the printable chars *num* times. In this work, a shift is an increase of one in the decimal value of the char. Therefore 'a' with decimal value of 97 will be shifted once to 'b' with the decimal value of 98. If we are considering a shift to num=10, then '}' with decimal value of 125 will be shifted ten times to '(' with decimal value of 40. Only strings with printable chars can be encoded. When trying to encode a problematic string the code should **raise a CyclicCharsError**.
Example:
For 'Hello World' and num = 15 the method will return a string 'Wt{{~/f~"{s'.

Cyclic Chars Decoding - decode_cyclic_chars(self, num:int) -> 'String'

Decoding for a Cyclic Chars string. If the string cannot be decoded this method will **raise a CyclicCharsDecodeError.**

Example:

For the string: 'Wt{{~/f~"{s' and num = 15 the method will return a string 'Hello World'.

Cyclic Bits Encoding - def cyclic_bits(self, num:int) -> 'String'

Using a circular bit shift for the string will produce the result. In this case shift is the bit circular shift operator. You should convert the string into bits and shift them num times. The char 'õ' has decimal value of 245, therefore its binary value is 11110101. Shifting it once will produce the char 'ë' with the decimal value of 235.

Example:

For the string: 'Hello World' and num = 8 the method will return a string *'ello WorldH'*.

Cyclic Bits Decoding - decode_cyclic_bits(self, num:int) -> 'String'

Decoding for a Cyclic Bits string.

Example:

For the string *'ello WorldH'* and num = 8 the method will return a string 'Hello World'.

Byte Pair Encoding - def byte_pair_encoding(self) -> 'String'

Repeatedly replace the most common pair of consecutive characters in the string with a new, unused char. If, at an iteration step in the algorithm, there is more than one pair with the highest frequency, replace **the leftmost pair**. String should keep track of the replacement "rules" so that the original string can be reconstructed. The output will be a string with a property named rules filled with a list of sorted replacement rules. Four replacement chars groups can be used according to the group priority. If one of the chars in a certain group is used in the original string**, do not use the relevant group at all.** Please see the table below for details on the groups. If all the groups cannot be used the function should **raise a BytePairError**. Inside each group, you should use the chars according to thier decimal value, from lower to higher. The encoding stops when all pairs appear only once.

Group Name	Decimal values in ASCII	priority
Other	33-47, 58-64, 91-96, 124-126	1
Digits	48-57	2
Upper Case	65-90	3
Lower Case	97-122	4

Try first to use the "other" group. if one of the chars in the original string is in the group the code shouldn't use any of the chars group. Test again for the second group in priority, the

"digits". Again, if one of the chars in the original string is in the group the code should't use any of the chars group. And so on.
After deciding which groups, you can use, order all the chars according to priority and decimal value.

Example:
For the string 'aaabdaaabac' we can use the groups "other", "digits", and "upper case". We cannot use the "lower case" group.

1. Scanning the string we found that:

pair	appearance
'aa'	2
'ab'	2
'bd'	1
'da'	1
'ba'	1
'ac'	1

2. We will replace the pair 'aa' (the leftmost pair) with the first character in the *other* group '!' (decimal value 33). We will get a new string '*!abd!abac'* and the first rule '! = aa'
3. Scanning the new string we found that:

pair	appearance
'!a'	2
'ab'	2
'bd'	1
'd!'	1
'ba'	1
'ac'	1

4. We will replace the pair '!a' with the second character in the other group '"' (decimal value 34). We will get a new string '*"bd"bac'* and the second rule '" = !a'
5. Repeating this will produce the string '*#d#ac'* and the third rule '# = "b'
6. The process stops because all the pairs have only one appearance
7. The method will return a string '*#d#ac'* with rules as list of str ['! = aa', '" = !a', '# = "b']

<u>Byte Pair Decoding – decode_byte_pair(self) -> 'String'</u>
Decoding for a Byte Pair string. This method uses the replacement rules. If rules property is undefined or an empty list, the method should raise **BytePairDecodeError.**
Example:
For the string '*#d#ac'* with rules=['! = aa', '" = !a', '# = "b'] return a string 'aaabdaaabac'.

<u>Histogram of chars - def histogram_of_chars(self) -> dict</u>
Count the chars from each bin. The output should be a dictionary with bins as keys, and chars' count as values. This is the bins:

- "*control code*" - see string basics.
- "digits"
- "upper"
- "lower"

- "other printable"
- "higher than 128"

For more information please see the string basics paragraph.

Tips:

Do not use any import in your coder. If you use the import statement your solution is invalid – **your grade will be 0**.
If your code crashes for any reason – **your grade will be 0**.
Pay attention to our suggested prototype, changing the class methods will probably decrease your grade.

Some Codes that may help:

- Testing will be something like this:

```
h = String("Hello World")
h_base64 = h.base64()
if h_base64 == "SGVsbG8gV29ybGQ":
   grade ++
```

- Base64 package – to validate your work you can compare your results to the base64 package as follows:

```
import base64
encodedBytes = base64.b64encode(string.encode("utf-8"))
print(str(encodedBytes, "utf-8"))
```

- Random string function with length num – to generate_random_string you can use this function:

```
import random
def generate_random_string(num:int) -> str:
   letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789/*-+ ?><:.,/';"
   return ''.join(random.choice(letters) for i in range(num))
```

- A script that will print all the relevant decimal values and their correlated chars according Unicode.

```
for i in range(0,256):
     print(str(i) + " is the char " + chr(i))
```