PYTHON MINOR ASSIGNMENT- 07

```
''' 1. Write a function that takes a string as a parameter and returns a string with every successive
repetitive character replaced with a star(*). For example, ’balloon’ is returned as ’bal*o*n’.'''

def replace(str):
    res=' '
    for i in range(len(str)):
        if i>0 and str[i]==str[i-1]:
            res+='*'
        else:
            res+=str[i]
    return res
str='balloon'
print(replace(str))
```

```
''' 2. Write a function that takes two strings and returns True if they are anagrams and False otherwise.
A pair of strings is anagram if the letters in one word can be arranged to form the second one.'''

def anagrams(str1,str2):
    if sorted(str1)==sorted(str2):
        return True #if not then 'None'
print(anagrams('silent','listen'))
```

```
''' 3. Write a function that takes a sentence as an input parameter and displays the number of words in
the sentence.'''

def count_words(sentence):
    return len(sentence.split())
print(count_words("This is my Python Minor Assignment. Done by Shubhangini"))  # Output: 9
```

```
''' 4. Create a program to count the number of occurrences of a specific character in a string.'''

str='Shubhangini'
c='n'#character i want to find in str
print(str.count(c)) #o/t: 2
```

```
''' 5. Write a Python program to find the length of the longest word in a sentence.'''

str='My name is Shubhangini'
words=str.split()
print(max(words ,key=len)) #key point is to return by len
```

```
''' 6. Write a Python function that takes a string and returns a new string where every vowel in the
input string is replaced by the next vowel in sequence (a → e, e → i, i → o, o → u, u → a).'''

def replace(s):
    vowels = "aeiou"
    result = ""
    for char in s:
        if char in vowels:
            result += vowels[(vowels.index(char) + 1) % len(vowels)]
        else:
            result += char
    return result
print(replace("Shubhangini"))  # Output:Shabhengono
```

```
'''7. Write a Python program that checks if a string is a ”rotational palindrome.” A rotational
palindrome is a string that can be rearranged cyclically to form a palindrome.'''

def is_rotational_palindrome(s):
    for i in range(len(s)):
        rotated = s[i:] + s[:i] #eg: aab , 1st rotate- aba , 2nd- baa
        if rotated == rotated[::-1]: #aab==baa
            return True
    return False
print(is_rotational_palindrome("aab"))  # Output:True
```

```
''' 8. Implement a program to check if a string is a valid URL.'''

import re #regular Expression
def validURL(url):
    pattern = r"^(https?://)?(www\.)?[a-z0-9]+(\.[a-z]+)+(/.*)?$"
    return bool(re.match(pattern, url))

print(validURL("google.com"))  # Output: True
```

```
''' 9. Create a program to find the number of vowels and consonants in a string.'''

def count(str):
    vowels = 'aeiouAEIOU'
    count_vowels = 0
    count_consonants = 0
    for char in str:
        if char in vowels:
            count_vowels += 1
        else:
            count_consonants+=1
    return count_consonants,count_vowels
print(count('Shubhangini'))
```

```
''' 10. Write a script that reads a line of text as a string, tokenizes the string with the split method
and outputs the tokens in reverse order. Use space characters as delimiters.'''

def reverse_tokens(s):
    return " ".join(s.split()[::-1]) #slicing

print(reverse_tokens("Hello world"))  # Output: world Hello
```

```
''' 11. Write a script that reads a line of text, tokenizes the line using space characters as delimiters
and outputs only those words beginning with the letter ’b’ and ending with the letter ’d’..'''

def filter_words(s):
    return [word for word in s.split() if word[0] == 'b' and word[-1] == 'd']

print(filter_words("bad bond bend bed"))
```

```
''' 12. Write a script that reads a five-letter word from the user and produces every possible three-
letter string, based on the word’s letters. For example, the three-letter words produced from the word
 “bathe” include “ate,” “bat,” “bet,” “tab,” “hat,” “the,” and “tea.” Challenge: Investigate the
functions from the itertools module, then use an appropriate function to automate this task'''

from itertools import permutations
def combinations(word):
    return ["".join(p) for p in permutations(word, 3)] #5C3
print(combinations("bathe"))
# Output: ['bat', 'bah', 'bae', ..., 'tha', 'the', ...]
```

```
''' 13. Check whether a sentence contains more than one space between words. If so, remove the extra
 spaces and display the results. For example, ’Hello   World’ should become ’Hello World’.'''

def space(str):
    return " ".join(str.split())
print(space("Kumari   Shubhangini"))
```

```
''' 14. Write a Python program to reverse the middle half of characters in a string.'''
def reverse_middle(s):
    n = len(s)
    mid = n // 2
    if n % 2 == 0:  # Even-length string
        mid_part = s[mid - 1:mid + 1]
        left = s[:mid - 1]
        right = s[mid + 1:]
    else:  # Odd-length string
        mid_part = s[mid]
        left = s[:mid]
        right = s[mid + 1:]
    return left + mid_part[::-1] + right

print(reverse_middle("abcdefgh"))  # Output: abfedcgh
```

```
''' 15. Write a Python program to print the substrings of a character having a particular frequency. For
’aabbbccccddddd’, you should print ’bbb’ if particular frequency is 3. 16. Write a code to extract unique
characters of a string in sorted order'''

from collections import Counter
def frequency(s, freq):
    result = ""
    counts = Counter(s)
    for char in s:
        if counts[char] == freq:
            result += char
    return "".join(set(result))
print(frequency("aabbbccccddddd", 3))  # Output: bbb
```

```
''' 16. Write a code to extract unique characters of a string in sorted order'''
def unique_sort(s):
    return "".join(sorted(set(s)))

print(unique_sort("banana"))  # Output: abn
```

```python
s = "how now brown cow"
print(s[s.find('o'):s.rfind('o')])
```

Answer: `"ow now brown c"`

2. Code:

```python
chr(ord('A') + 2) + chr(ord('Z') - 3)
```

Answer: `"CW"`

3. Code:

```python
s = "abc123def456ghi789"
indices = [i for i, c in enumerate(s) if c == '']
result = s[indices[1]+1:indices[2]] + s[indices[4]+1:]
print(result)
```

Answer: Error

4. Code:

```python
s = "abracadabra"
print(s.replace(s[s.find('a'):s.find('r')], "XYZ"))
```

Answer: `"XYZracadabra"`

5. Code:

```python
s = "hello"
shift = 2
print("".join(chr((ord(c) - 97 + shift) % 26 + 97) for c in s))
```

Answer: `"jgnnq"`

6. Code:

```python
s = "mississippi"
print("".join(sorted(set(s))))
```

Answer: `"imps"`

Q18: Output for the given statements with `quote = "The quick brown fox jumps over the lazy dog"`

- `quote.upper()`: `"THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"`
- `quote[::-1]`: `"god yzal eht revo spmuj xof nworb kciuq ehT"`
- `quote[4:19]`: `"quick brown fox"`
- `quote.replace('fox', 'cat')`: `"The quick brown cat jumps over the lazy dog"`
- `quote.count('o')`: `4`
- `quote.startswith('The')`: `True`
- `'brown' in quote`: `True`
- `quote.islower()`: `False`

Q19: Output for the given statements with `quote = "Knowledge is power. Power is gained through knowledge."`

- `quote.find('power')`: `14`
- `quote.rfind('knowledge')`: `46`
- `quote.title()`: `"Knowledge Is Power. Power Is Gained Through Knowledge."`
- `quote.lower()`: `"knowledge is power. power is gained through knowledge."`
- `quote.upper()`: `"KNOWLEDGE IS POWER. POWER IS GAINED THROUGH KNOWLEDGE."`
- `quote.endswith('knowledge.')`: `True`
- `quote.split(' ')`: `['Knowledge', 'is', 'power.', 'Power', 'is', 'gained', 'through', 'knowledge.']`
- `quote.partition('is')`: `('Knowledge ', 'is', ' power. Power is gained through knowledge.')`
- `quote.isalpha()`: `False` (because it contains spaces and punctuation)

Q20: Output for regular expressions with `string1 = 'Python Programming Language'`

- `re.search('. m?', string1).group()`: `"P"` (matches the first character followed by optional `m`)
- `re.search('.*Language$', string1).group()`: `"Python Programming Language"` (matches entire string ending with "Language")
- `re.search(' w* s w*', string1)`: `None` (no matching pattern in the string)
- `re.search('.*', string1).group()`: `"Python Programming Language"` (matches the entire string)

Q21: Output for operations with `string1 = 'Python Programming Language'`

- `re.fullmatch(r'[A-Za-z]*', string1).group()`: `None` (spaces are not matched by `[A-Za-z]*`)
- `re.sub(r'Programming', 'Coding', string1)`: `"Python Coding Language"`
- `re.split(r'+', string1)`: Error (incorrect regex pattern; needs escaping for `+`)
- `re.findall(r'+', string1)`: Error (incorrect regex pattern; needs escaping for `+`)

```python
''' 22. Write a python program to check if a string is symmetric or asymmetric.'''
def is_symmetric(string):
    if string == string[::-1]:
        return "The string is symmetric."
    else:
        return "The string is asymmetric."
string = input("Enter a string: ")
print(is_symmetric(string))
```

```python
''' 23.Given a string s and index i, write a python program to delete the i-th value from s.'''
def delete_char(string, i):
    if 0 <= i < len(string):  # Ensure the index is valid
        return string[:i] + string[i + 1:]  # Remove the i-th character
    else:
        return "Invalid index."
print(delete_char('Shubhangini',7))
```

```python
''' 24. Write a python program to find the character having maximum frequency in a string'''

from collections import Counter
def maxFreq(string):
    if not string:
        return "The string is empty."
    frequency = Counter(string)
    max_char = max(frequency, key=frequency.get)
    return max_char
print(maxFreq('Hellllllllllooooooooooooo'))
```

```python
'''25. Use regular expressions to validate secure passwords. Passwords must have a minimum of 8 characters and contain at least one each from uppercase characters, lowercase characters, digits, and punctuation characters, such as characters in '!@#$%&*?'.'''

import re
def password(password):
    pattern = r'^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)(?=.*[!@#$%&*?]).{8,}$'
    if re.fullmatch(pattern, password):
        return "The password is secure."
    else:
        return "The password is not secure."
print(password('Shubhangini@2003'))
```

```python
''' 26. Use regular expressions and the findall function to count the number of digits, non-digit characters, whitespace characters and words in a string.'''

import re
string ='Shubhangini@@2003'
digits = len(re.findall(r'\d', string))
non_digits = len(re.findall(r'\D', string))
whitespaces = len(re.findall(r'\s', string))
words = len(re.findall(r'\w+', string))

print(f"Digits: {digits}, Non-digits: {non_digits}, Whitespace: {whitespaces}, Words: {words}")
```