4_ASSIGNMENT PYTHON UTKARSH

In [ ]:
```
1. Write a function translate() that will translate a. That is, double
every consonant and place an occurrence of "o" in between. For
example, translate("this is fun") should return the string "tothohisos
isos fofunon".
```

In [4]:
```
def translate(input_string):
    consonants = 'bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ'
    output_string = ''

    for char in input_string:
        if char in consonants:
            output_string += char + 'o' + char
        else:
            output_string += char

    return output_string

print(translate("this is fun"))
```

```
tothohisos isos fofunon
```

In [ ]:
```
2. Write a program that contains a function that has one parameter,
n, representing an integer greater than 0. The function should return
n! (n factorial). Then write a main function that calls this function
with the values 1 through 20, one at a time, printing the returned
results. This is what your output should look like:
1 1
2 2
3 6 etc
```

In [6]:
```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)

for n in range(1, 21):
    print(n, factorial(n))
```

```
1 1
2 2
3 6
4 24
5 120
6 720
7 5040
8 40320
9 362880
10 3628800
11 39916800
12 479001600
13 6227020800
14 87178291200
15 1307674368000
16 20922789888000
17 355687428096000
18 6402373705728000
19 121645100408832000
20 2432902008176640000
```

In [ ]:
```
3. Write a function find_longest_word() that takes a list of words and
returns the length of the longest one.
```

In [25]:
```python
def find_longest_word(a):
    length = 0
    w = ''
    for wo in a:
        if len(wo) > length:
            length = len(wo)
            w = wo
    return length

words = ["apple", "banana", "cherryeeee", "date"]
print(find_longest_word(words))
```

```
10
```

In [ ]:
```
4. Define a simple "spelling correction" function correct () that takes a
string and sees to it that
a)two or more occurrences of the space character is compressed into one,
and
b)inserts an extra space after a period if the period is directly followed
by a letter. e.g. correct ("This is very funny and cool.Indeed!") should
return "This is very funny and cool. Indeed!"
```

In [37]:
```python
import re

def correct(text):
    c = re.sub(' +', ' ', text)
    c = re.sub(r'\.(?=[A-Za-z])', '. ', c)
    return c

text = "This is very funny and cool.Indeed!"
print(correct(text))
```

```
This is very funny and cool. Indeed!
```

In [ ]:

```
5. In English, the present participle is formed by adding the suffix -ing
to the infinite form: go -> going. A simple set of heuristic rules can be
given as follows:
a) If the verb ends in e, drop the e and add ing (if not exception: be,
see, flee, knee, etc.)
b) If the verb ends in ie, change ie to y and add ing
c) For words consisting of consonant-vowel-consonant, double the final
letter before adding ing
d) By default just add ing
Your task in this exercise is to define a function make_ing_form() which,
given a verb in infinitive form, returns its present participle form. Test
your function with words such as lie, see, move and hug. However, you
must not expect such simple rules to work for all cases.
```

In [55]:

```python
def make_ing_form(verb):
    if verb.endswith('ie'):
        return verb[:-2] + 'ying'
    elif verb.endswith('e') and not verb.endswith('ee'):
        return verb[:-1] + 'ing'
    elif len(verb) >= 2 and verb[-1] not in 'aeiou' and verb[-2] in 'aeiou':
        return verb + verb[-1] + 'ing'
    else:
        return verb + 'ing'

verbs = ["lie", "see", "move", "hug"]

for verb in verbs:
    print(make_ing_form(verb))
```

```
lying
seeing
moving
hugging
```

In [ ]:

```
6. Make the program of network error codes done yesterday with match-case.
```

In [77]:

```python
def get_error_message(code):
    match code:
        case 400:
            print("error code:", code)
            print("400 Bad Request. The 400 status code, or Bad Request error, mean
        case 401:
            print("error code:", code)
            print("The 401 status code, or an Unauthorized error, means that the us
        case 502:
            print("error code:", code)
            print("The 502 Bad Gateway error means that the server is a gateway or
        case 404:
            print("error code:", code)
            print("The 404 status code, or a Not Found error, means that the reques
        case 500:
            print("error code:", code)
            print("The 500 status code, or Internal Server Error, means that the se
        case _:
            return "Different error code, don't have information"

print("1.404 2.401 3.502 4.400 5.500")
num = int(input("Choose error code: "))
get_error_message(num)
```

```
1.404 2.401 3.502 4.400 5.500
error code: 404
The 404 status code, or a Not Found error, means that the requested resource was not
found on the server.
```

In [ ]:

```
7. Define a class Student with data members rollno, name, mark1, mark2,
mark3, total, avg. Use appropriate method for entering the details and
displaying the details. Also define a method for calculating the total mark
and average. Create an object for the class and invoke all the methods.
```

In [92]:

```
class Student:
    def __init__(self, r, n, m1, m2, m3):
        self.rollno = r
        self.name = n
        self.mark1 = m1
        self.mark2 = m2
        self.mark3 = m3

    def ctotal(self):
        self.total = self.mark1 + self.mark2 + self.mark3

    def cavg(self):
        self.avg = self.total / 3

    def display(self):
        print(f"Roll No: {self.rollno}")
        print(f"Name: {self.name}")
        print(f"Mark 1: {self.mark1}")
        print(f"Mark 2: {self.mark2}")
        print(f"Mark 3: {self.mark3}")
        print(f"Total: {self.total}")
        print(f"Average: {self.avg}")

student1 = Student(14, "utkarsh", 85, 80, 78)
student1.ctotal()
student1.cavg()
student1.display()
```

```
Roll No: 14
Name: utkarsh
Mark 1: 85
Mark 2: 80
Mark 3: 78
Total: 243
Average: 81.0
```

In [ ]:

```
8. Create a class corresponding to BankAccount with the data members accno, custnam
balamt. Use two methods for entering the details and displaying the details. Define
more method for checking whether the balamt is greater than 20,000 and display a
corresponding message. Create an object for the class and invoke all the methods.
```

```
In [71]: class BankAccount:
             def __init__(self, accno, Custname, balanamt):
                 self.accno = accno
                 self.Custname = Custname
                 self.balanamt = balanamt

             def display(self):
                 print(f"Account No: {self.accno}")
                 print(f"Customer Name: {self.Custname}")
                 print(f"Balance Amount: {self.balanamt}")

             def balance(self):
                 if self.balanamt > 20000:
                     print("Balance amount is greater than 20,000")
                 else:
                     print("Balance amount is not greater than 20,000")

         acc = BankAccount(56572910193, "vikas yadav", 35000)
         acc.display()
         acc.balance()
```

```
Account No: 56572910193
Customer Name: vikas yadav
Balance Amount: 35000
Balance amount is greater than 20,000
```

```
In [69]: 9. Write a python program to demonstrate method overloading.
```

```
  Cell In[69], line 1
    9. Write a python program to demonstrate method overloading.
       ^
SyntaxError: invalid syntax
```

```
In [72]: class Overloading:
             def methOver(self, a, b, c=None):
                 if c is None:
                     return a - b
                 else:
                     return a - b - c

         # Create an instance of the Overloading class
         obj = Overloading()

         # Call the methOver method with two different sets of arguments
         result1 = obj.methOver(60, 10)
         result2 = obj.methOver(80, 20, 31)

         # Print the results
         print(result1)  # Output: 50 (60 - 10)
         print(result2)  # Output: 29 (80 - 20 - 31)
```

```
50
29
```

```
In [73]: 10. Write a program to demonstrate inheritance in python.
```

```
  Cell In[73], line 1
    10. Write a program to demonstrate inheritance in python.
        ^
SyntaxError: invalid syntax
```

In [103…

```
class Animal:
    def __init__(self, name):
        self.name = name

    def eat(self):
        print(f"{self.name} is eating")

class Dog(Animal):
    def bark(self):
        print(f"{self.name} is barking")

# Create an instance of the Dog class
d = Dog("Tommy")

# Call methods of the Dog class
d.eat()
d.bark()
```

```
Tommy is eating
Tommy is barking
```

In [88]:

```
11. Write a Python program to check that a string contains only a certain set of ch
(in this case a-z, A-Z and 0-9).
```

```
  Cell In[88], line 1
    11. Write a Python program to check that a string contains only a certain set of
characters
        ^
SyntaxError: invalid syntax
```

In [113…

```
import re
def check_string(s):
 match = re.fullmatch(r'[a-zA-Z0-9]*', s)
 return match is not None
print(check_string("niti09092u"))
print(check_string("Ram and shayam"))
```

```
True
False
```

In [77]:

```
12. Write a Python program that matches a string that has an a followed by zero or
b's
```

```
  Cell In[77], line 2
    b's
    ^
SyntaxError: unterminated string literal (detected at line 2)
```

In [125…

```
import re
if re.match(r'ab*', "aaaabbbbbbbb"):
 print("found")
```

```
else:
 print("Not Found")
```

```
found
```

In [79]:
```
13. Write a Python program that matches a string that has an a followed by one or m
```

```
  Cell In[79], line 1
    13. Write a Python program that matches a string that has an a followed by one o
r more b's

^
SyntaxError: unterminated string literal (detected at line 1)
```

In [139…
```
import re
if re.match(r'ab*', "aaaabbbbbbbb"):
 print("found")
else:
 print("Not Found")
```

```
found
```

In [81]:
```
14. Write a Python program that matches a string that has an a followed by zero or
```

```
  Cell In[81], line 1
    14. Write a Python program that matches a string that has an a followed by zero
or one 'b'
        ^
SyntaxError: invalid syntax
```

In [156…
```
import re

def text_match(text):
    pattern = 'ab?'
    if re.search(pattern, text):
        return 'matched'
    else:
        return 'Not matched'

print(text_match("ab"))
```

```
matched
```

In [83]:
```
15. Write a Python program that matches a string that has an a followed by three 'b
```

```
  Cell In[83], line 1
    15. Write a Python program that matches a string that has an a followed by three
'b'
        ^
SyntaxError: invalid syntax
```

In [171…
```
import re

def text_match(text):
    pattern = 'ab{3}'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match("abbb"))
print(text_match("aabbbbbc"))
```

```
Found a match!
Found a match!
```

In [85]:
```
16. Write a Python program that matches a string that has an a followed by two to t
```

```
  Cell In[85], line 1
    16. Write a Python program that matches a string that has an a followed by two t
o three 'b'.
        ^
SyntaxError: invalid syntax
```

In [192…
```
import re

def text_match(text):
    pattern = 'ab{3}'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match("ab"))
print(text_match("aabbbbbc"))
```

```
Not matched!
Found a match!
```

In [ ]:
```
17. Write a Python program to find sequences of lowercase letters joined with a und
```

In [210…
```
import re

def text_match(text):
    pattern = '^[a-z]+_[a-z]+$'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match("aab_cbbbc"))
print(text_match("aab_Abbbc"))
print(text_match("Aaab_abbbc"))
```

```
Found a match!
Not matched!
Not matched!
```

In [ ]:
```
18. Write a Python program to find the sequences of one upper case letter followed
lower case letters.
```

In [220…
```
import re

def text_match(text):
    pattern = '[A-Z]+[a-z]+$'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match("AaBbGg"))
print(text_match("Python"))
print(text_match("python"))
print(text_match("PYTHON"))
print(text_match("aA"))
print(text_match("Aa"))
```

```
Found a match!
Found a match!
Not matched!
Not matched!
Not matched!
Found a match!
```

In [ ]:

```
19. Write a Python program that matches a string that has an 'a' followed by anythi
ending in 'b'.
```

In [232…

```
import re

def text_match(text):
    pattern = 'a.*?b$'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match("aabbbbd"))
print(text_match("aabAbbbc"))
print(text_match("accddbbjjjb"))
```

```
Not matched!
Not matched!
Found a match!
```

In [ ]:

```
20. Write a Python program that matches a word at the beginning of a string.
```

In [241…

```
import re

def text_match(text):
    pattern = r'^\w+'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match(" a lazy dog."))
print(text_match(" lazy dog."))
```

```
Not matched!
Not matched!
```

In [ ]:
```
21. Write a Python program that matches a word at the end of a string, with optiona
punctuation.
```

In [251…
```
import re

def text_match(text):
    pattern = r'\w+[.,!?]*$'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match("The quick brown fox jumps over the lazy dog."))
print(text_match("The quick brown fox jumps over the lazy dog. "))
print(text_match("The quick brown fox jumps over the lazy dog "))
```

```
Found a match!
Not matched!
Not matched!
```

In [ ]:
```
22. Write a Python program that matches a word containing 'z'
```

In [268…
```
import re

def text_match(text):
    pattern = r'\w*z.\w*'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match("The quick brown fox jumps over the lazy dog."))
print(text_match("Python Exercises."))
```

```
Found a match!
Not matched!
```

In [ ]:
```
23. Write a Python program that matches a word containing 'z', not at the start or
the word.
```

In [281…
```
import re

def text_match(text):
    pattern = r'\Bz\B'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match("The quick brown fox jumps over the lazy dog."))
print(text_match("Python Exercises."))
```

```
Found a match!
Not matched!
```

In [ ]:
```
24. Write a Python program to match a string that contains only upper and lowercase
letters, numbers, and underscores.
```

In [299…
```
import re

def text_match(text):
    pattern = r'^[a-zA-Z0-9_]*$'
    if re.search(pattern, text):
        return 'Found a match!'
    else:
        return 'Not matched!'

print(text_match("The quick brown fox jumps over the lazy dog."))
print(text_match("Python_Exercises_1"))
```

```
Not matched!
Found a match!
```

In [ ]:
```
25. Write a Python program where a string will start with a specific number
```

In [306…
```
import re

def match_num(string):
    pattern = re.compile(r"^3")
    if pattern.match(string):
        return True
    else:
        return False

print(match_num('5-2345861'))  # False
print(match_num('3-2345861'))   # True
```

```
False
True
```

In [ ]:
```
26. Write a Python program to remove leading zeros from an IP address
```

In [316…
```
import re

ip = "216.08.094.096"
string = re.sub(r'\.[0]*', '.', ip)
print(string)
```

```
216.8.94.96
```

In [ ]:
```
27. Write a Python program to check for a number at the end of a string.
```

In [321…
```
import re

def end_num(string):
    pattern = re.compile(r".*[0-9]$")
    if pattern.match(string):
        return "End with Numbered"
    else:
        return "Not End with Numbered"
```

```
print(end_num('abcdef42'))  # End with Numbered
print(end_num('abcdef'))   # Not End with Numbered
```

```
End with Numbered
Not End with Numbered
```

In [ ]:
```
28. Write a Python program to search the numbers (0-9) of length between 1 to 3 in
string.
```

In [327…
```
import re

results = re.finditer(r"([0-9]{1,3})", "Exercises number 1, 12, 134, and 345 are im
print("Number of length 1 to 3")
for n in results:
    print(n.group(0))
```

```
Number of length 1 to 3
1
12
134
345
```

In [ ]:
```
29. Write a Python program to replace whitespaces with an underscore and vice versa
```

In [333…
```
import re
text = 'mind should be full of buisness'
text =text.replace (" ", "_")
print(text)
text =text.replace ("_", " ")
print(text)
```

```
mind_should_be_full_of_buisness
mind should be full of buisness
```

In [ ]:
```
30. Write a Python program to convert a date of yyyy-mm-dd format to dd-mm-yyyy
format.
```

In [339…
```
import re
def change_date_format(dt):
     return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', dt)
dt1 = "2023-05-11"
print("Original date in YYY-MM-DD Format: ",dt1)
print("New date in DD-MM-YYYY Format: ",change_date_format(dt1))
```

```
Original date in YYY-MM-DD Format:  2023-05-11
New date in DD-MM-YYYY Format:  11-05-2023
```

In [ ]:
```
31. Write a Python program to separate and print the numbers of a given string.
```

In [346…
```
import re
text = "One 1, Two 2, Three 3"
result = re.split(r"\D+", text)
for element in result:
 print(element)
```

```
1
2
3
```

In [ ]:
```
32. Write a Python program to find all words starting with 'a' or 'e' in a given st
```

In [357…
```
import re
t = " An apple a day keeps doctor away and everything is working for my highest pot
l = re.findall(r'[ae]\w+', t)
print(l)
```

```
['apple', 'ay', 'eeps', 'away', 'and', 'everything', 'est', 'ential']
```

In [ ]:
```
33. Write a Python program to abbreviate 'Road' as 'Rd.' in a given string.
```

In [361…
```
import re
s = 'Great britain Road'
print(re.sub('Road$', 'Rd.', s))
```

```
Great britain Rd.
```

In [ ]: