Q1. Print the following statement using print function

"This is my first Python Test"

In [1]:
```
print("This is my first Python Test")
```

```
This is my first Python Test
```

Q2. What is 8 to the power of 6 using inbuilt function?

In [2]:
```
pow(8,6)
```

Out[2]:
```
262144
```

Q3. Split this string:

```
s = "All#the#best#for#test"
```

into a list.

In [3]:
```
s = "All#the#best#for#test"

s.split("#")
```

Out[3]:
```
['All', 'the', 'best', 'for', 'test']
```

Q4. Given the variables:

```
Flower = roses
Count = 20
```

Use .format() method to print the following statement:

```
I have 20 yellow color roses.
```

In [4]:
```
Flower = "roses"
Count = 20

print("I have {} yellow color {}".format(Count,Flower))
```

```
I have 20 yellow color roses
```

Q5. The nested list is given, use the indexing method to find the word "Am here".

lst = [1,5,[3,9],[2,3,[5,[100,200,['Am here']]],29,11],[1,7]]

In [12]:
```
lst = [1,5,[3,9],[2,3,[5,[100,200,['Am here']]],29,11],[1,7]]

lst[3][2][1][2][0]
```

Out[12]:
```
'Am here'
```

Q6. From nested dictionary find the word "hi".

d = {'k1':[1,2,3,{'nice':['oh','man','insane',{'achieve':[1,2,3,'hi']}]}]}

In [14]:
```
d = {'k1':[1,2,3,{'nice':['oh','man','insane',{'achieve':[1,2,3,'hi']}]}]}

d['k1'][3]['nice'][3]['achieve'][3]
```

Out[14]:
```
'hi'
```

Q7. What is the main difference between a tuple and a list? Write down in points

In [15]:
```
# Tuple is immutable and list mutable
# Tuple is faster than list
# Tuple uses paranthesis() and list uses square brackets []
```

Q8. Create a function GetDomain that finds the email website domain from a string in the form:

```
python@datascience.in
```

So for example, passing "python@datascience.in (mailto:python@datascience.in)" would return: datascience.in

In [16]:
```
def GetDomain(email):
    return email.split('@')[-1]

GetDomain("python@datascience.in")
```

Out[16]:
```
'datascience.in'
```

Q9. Write function that returns True if the word 'python' is contained in the input string by user

In [19]:
```
def SearchPython(st):
    return 'python' in st.lower().split()


SearchPython("I like subject Python")
```

Out[19]:
```
True
```

Q10. Use lambda expressions and the filter() function to filter out words from a list that ends with the letter 't'.

For example:

```
seq = ['mat','dog','pet','run','cat','pizza','great']
```

Output:

```
['mat','pet','cat','great']
```

In [18]:
```
seq = ['mat','dog','pet','run','cat','pizza','great']

list(filter(lambda w: w[-1]=='t',seq))
```

Out[18]:
```
['mat', 'pet', 'cat', 'great']
```

Q11.You are driving a little too fast, and a police officer stops you.

Write a function name CheckSpeed to return one of 3 possible results: "No Sorry", "Small Sorry", or "Big Sorry".

- Check the speed of the car, if your speed is 60 or less, the result is "No Sorry".
- If speed is between 61 and 80 inclusive, the result is "Small Sorry".
- If speed is 81 or more, the result is "Big Sorry".

Note:- Unless it is your birthday (take it as a boolean value in the parameters of the function) -- on your birthday, your speed can be 5 higher in all cases.

In [20]:
```
def CheckSpeed(speed, is_birthday):
    # check value of birthday is true or false and decrease speed with constan
    if is_birthday:
        s= speed - 5
    else:
        s = speed
    # Check the conditions
    if s > 80:
        return 'Big Sorry'
    elif s > 60:
        return 'Small Sorry'
    else:
        return 'No Sorry'

CheckSpeed(70,False)
```

Out[20]:
```
'Small Sorry'
```

Q12. Program to find the largest length element from a list.

```
colors=["brown", "cherry", "pink", "red"]
```

In [22]:
```
colors = ["brown", "cherry", "pink", "red"]

largest_element = max(colors, key=lambda x: len(x))

print("Largest length element:", largest_element)
```

```
Largest length element: cherry
```

In [25]:
```
#Another way

colors = ["brown", "cherry", "pink", "red"]

length = 0
largest_element = ""

for x in colors:
    if len(x) > length:
        length = len(x)
        largest_element = x

print("Largest length element:", largest_element)
```

```
Largest length element: cherry
```

Q13. Write a function to give the following output from input.

Input: aabbbcccdeeff

Output: a2b3c3d1e2f2

In [26]:
```
def charcterCount(s):
    # Take empty string and length of string
    r = ""
    l = len(s)

    # Check for length 0
    if l == 0:
        return ""

    # Check for length 1
    if l == 1:
        return s + "1"

    # Intialize Values
    cnt = 1
    i = 1

    while i < l:

        # Check to see if it is the same letter
        if s[i] == s[i - 1]:
            # Increment the cnt variable if same as previous
            cnt += 1
        else:
            # Otherwise store the previous data
            r = r + s[i - 1] + str(cnt)
            cnt = 1

        # Add count to i for termination of while loop
        i += 1

    # Put everything back into r
    r = r + s[i - 1] + str(cnt)

    return r

s = "aabbbcccdeeff"

print(charcterCount(s))
```

```
a2b3c3d1e2f2
```

Q 14. What is unique about a set?

In [27]:
```
#A set is a collection which is unordered, unchangeable, and unindexed.
```

Q 15. Write a function that asks for an integer and prints the square root of it. Use a `while` loop with a `try`, `except`, `else` block to account for incorrect inputs.

In [28]:
```
import math

def get_square_root():
    while True:
        try:
            num = int(input("Enter an integer: "))
            square_root = math.sqrt(num)
        except ValueError:
            print("Invalid input. Please enter an integer.")
        else:
            print("Square root:", square_root)
            break

get_square_root()
```

```
Enter an integer: 12
Square root: 3.4641016151377544
```

Q 16. Write a function GenerateListTuple that accepts a sequence of comma-separated numbers and generates a list and a tuple containing the numbers.

Example: User enter 1,2,3,4,5,6

Output: [1,2,3,4,5,6]

and (1,2,3,4,5,6)

In [30]:
```
def GenerateListTuple(numbers):
    nlist = numbers.split(",")
    ntuple = tuple(nlist)
    return nlist, ntuple


numbers = input("Enter a comma-separated numbers: ")
rlist, rtuple = GenerateListTuple(numbers)
print("List:", rlist)
print("Tuple:", rtuple)
```

```
Enter a comma-separated numbers: 1,2,3,4,5,6
List: ['1', '2', '3', '4', '5', '6']
Tuple: ('1', '2', '3', '4', '5', '6')
```

Q17. What is the correct way to call a function?

a) get_max_num([57, 99, 31, 18])

b) call.(get_max_num)

c) def get_max_num([57, 99, 31, 18])

d) call.get_max_num([57, 99, 31, 18])

In [ ]:
```
a)
```

Q18. Write a program using list comprehension to print a list of not the multiples of 3 and 6 and less then 100.

In [31]:
```
multiples = [num for num in range(100) if num % 3 != 0 and num % 6 != 0]

print(multiples)
```

```
[1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26, 28, 29, 3
1, 32, 34, 35, 37, 38, 40, 41, 43, 44, 46, 47, 49, 50, 52, 53, 55, 56, 58, 5
9, 61, 62, 64, 65, 67, 68, 70, 71, 73, 74, 76, 77, 79, 80, 82, 83, 85, 86, 8
8, 89, 91, 92, 94, 95, 97, 98]
```

Q19. Implement a function that checks whether a given numbers is a palindrome or not.

In [34]:
```
def palindrome(num):
    num_str = str(num)
    rev_str = num_str[::-1]
    return num_str == rev_str


n = int(input("Enter the number: "))

if palindrome(n):
    print(number, "is a palindrome.")
else:
    print(number, "is not a palindrome.")
```

```
Enter the number12345
12321 is not a palindrome.
```

Q20. Create a list of tuples matching these lists of name and age?

Name = ["Priya", "Maddy", "David"]

Age = [19, 25 , 22]

Output : [("Priya", 19), ("Maddy", 25), ("David", 22)]

In [ ]:
```
Name = ["Priya", "Maddy", "David"]

Age = [19, 25 , 22]

#zip() function is used to zip the data

list(zip(Name,Age))
```

Q21. Define Name Error with example

In [36]:
```
#A NameError is a type of exception that occurs when a local or global name is

print(p)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [36], in <cell line: 3>()
      1 #A NameError is a type of exception in Python that occurs when a loca
l or global name is not found.
----> 3 print(p)

NameError: name 'p' is not defined
```

Q22. Implement a class called Car with attributes make, model, and year. Include methods to start the car, stop the car, and display the car's details.

In [48]:
```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def start_car(self):
        print("The car has started.")

    def stop_car(self):
        print("The car has stopped.")

    def display_details(self):
        print("Car Details:")
        print("Make:", self.make)
        print("Model:", self.model)
        print("Year:", self.year)


my_car = Car("Volkswagon", "Ameo", 2020)
my_car.start_car()
my_car.display_details()
my_car.stop_car()
```

```
The car has started.
Car Details:
Make: Volkswagon
Model: Ameo
Year: 2020
The car has stopped.
```

Q23. Make a function called CountVowels that takes a string as input and counts the number of vowels in that string. Take 3 different cases perfom unit testing. (4 Marks)

In [43]:
```
%%writefile countvowels.py

def CountVowels(text):
    vowels = 'aeiouAEIOU'
    vowel_count = 0
    if not text:
        raise ValueError("Input string cannot be empty.")
    else:
        for char in text:
            if char in vowels:
                vowel_count += 1
    return vowel_count
```

```
Overwriting countvowels.py
```

In [46]:
```
%%writefile test_countvowels.py

import unittest
import countvowels

class PalindromeTest(unittest.TestCase):
    def test_empty_string(self):
        with self.assertRaises(ValueError):
            countvowels.CountVowels("")

    def test_one_word_string(self):
        self.assertEqual(countvowels.CountVowels("rhythm"),0)
        self.assertEqual(countvowels.CountVowels("m"),0)
        self.assertEqual(countvowels.CountVowels("python"),1)
        self.assertEqual(countvowels.CountVowels("data"),2)

    def test_multi_word_strings(self):
        self.assertEqual(countvowels.CountVowels("hello himani"),5)
        self.assertEqual(countvowels.CountVowels("python good"),3)


if __name__ == '__main__':
    unittest.main()
```

```
Overwriting test_countvowels.py
```

In [47]:
```
! python test_countvowels.py
```

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

In [ ]: