```
 1
1   2
1    2    3
1    2    3   4
1    2    3   4   5
```

Chapter 2 Python Revision Tour-II

1.**String in Python**

1. String: A string is a sequence of characters enclosed in single quotes, double quotes, or triple quotes.
2. Sequential Data Type: A string is a sequential data type in Python because its characters are arranged in a sequence.
3. Indexing: Each character in a string can be accessed using its index. Python supports forward (positive) indexing and backward (negative) indexing.

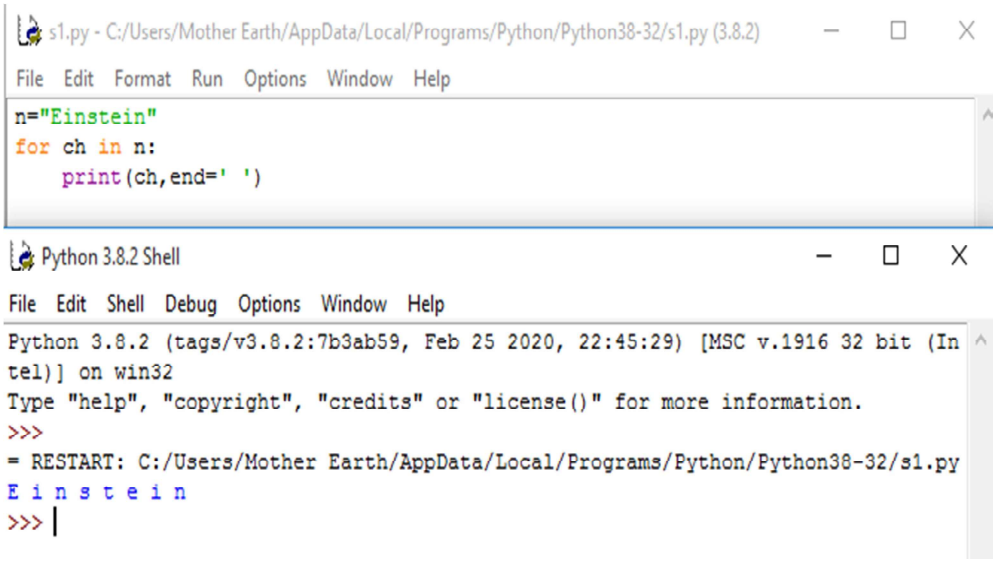
Example:

N=”EINSTEIN”

0	1	2	3	4	5	6	7
E	I	N	S	T	E	I	N
-8	-7	-6	-5	-4	-3	-2	-1

1. Traversing of string:

```
n=”Einstein”
for ch in n:
    print(ch, end=’ ‘)
```



2. String operators:

a. **+ Operator:** String can be concatenated.

```
n=”Einstein”
m=”school”
p=n+m
print(p)
output:
 Einsteinschool
```

c. String replication (*)

String Replication: String replication means repeating a string or character a specified number of times using the * operator. It generates a new string containing the repeated sequence.

```
n=”einstein”
print(n*3)
output: einsteineinsteineinstein
```

3. Membership operators

Membership Operators: Membership operators (`in` and `not in`) check whether a character or substring is present in a given string. They return a Boolean value (`True` or `False`).

a. in
b. not in

Eg.

```
>>a=”Einstein”
>>”Ein” in a
>>True
>>”Ein” not in a
>>False
```

c. Comparison operator ==, < , >, <= >=, !=

```
>>a=”Einstein”
>>b=”Lalaganj”
>>a==b
>>False
```

d. String Slices:

String slicing means extracting a part of a string by specifying a range of indexes inside square brackets `[]`.

Syntax of String Slicing in Python:

```
string_name[start : stop : step]
```

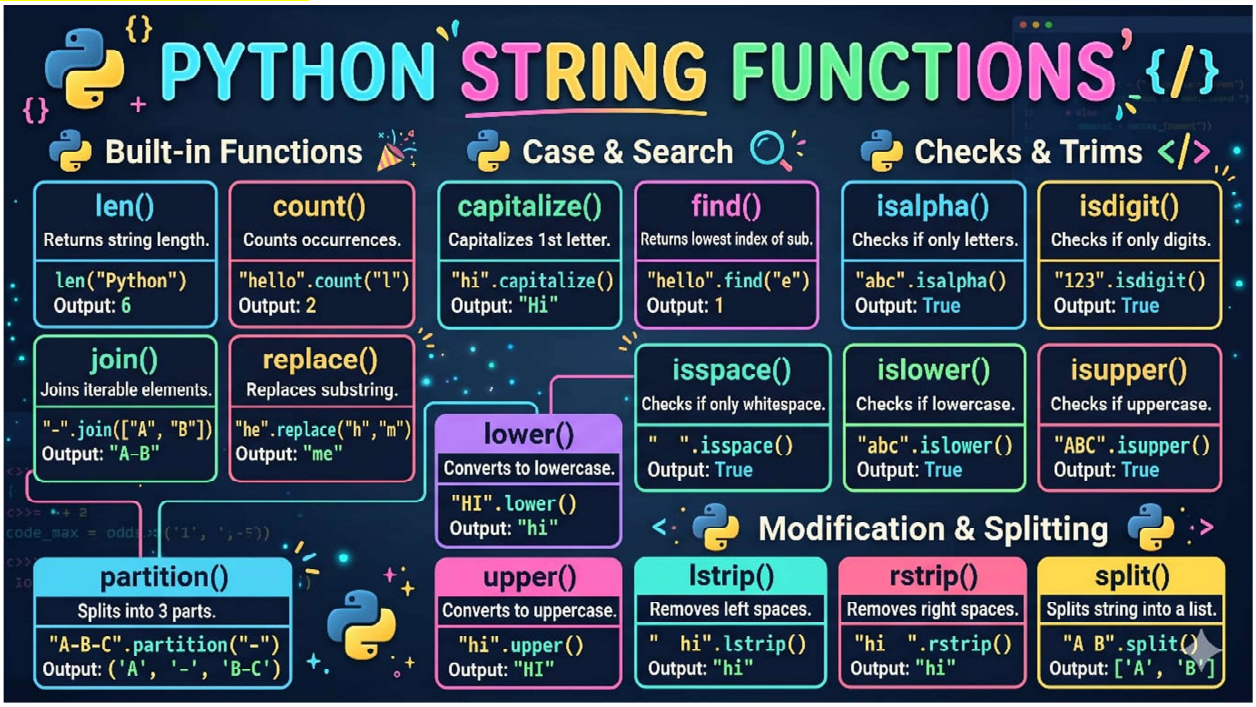
- start: Index from which slicing begins (included).
- stop: Index at which slicing ends (excluded).
- step: Number of positions to skip between characters (optional).

Syntax of String Slicing in Python:

```
string_name[start : stop : step]
```

- start: Index from which slicing begins (included).
- stop: Index at which slicing ends (excluded).
- step: Number of positions to skip between characters (optional).

e. String built in functions



A. string.capitalize(): Retrun a copy of string with its first character capitalized.

B. string.find(sub_string, start, end) : Returns the lowest index in the string where the substring is found within the slice range of start and end.
C. string.isalnum() : Returns True if the characters in the string are alphanumeric(alphabets or numbers).
D. string.isalpha(): Returns True if all characters in the string are alphabetic and there is at least one character, False otherwise.
E. string.isdigit()Returns Trueif string contains digits.
F. string.isspace():Returns True if all characters in the string are whitespaces
G. string.islower() : Returns True if all characters in the string are lower case.
H. string.isupper(): Returns True if all characters in the string are upper case.
I. string.lower(): Converts a string into lower case.
J. string.upper() : Converts a string into upper case.
K. string.lstrip() : Returns a left trim version of the string.
L. string.rstrip() : Returns a right trim version of the string.
M. string.split() : Splits the string at the specified separator, and returns a list

:List in Python:

A list is a standard data type of python that can store a sequence of values belonging to any type and defined inside square brackets.

Eg.

```
[]    # Empty list
[5,7,8,9]  # List of Integers
['a','b','c']   # List of characters
['a', 1, 'b', 'c', 6.9, 'zero']    # Mixed value types
['one','two', 'three']  # list of strings
```

Note Lists are mutable i.e. (modifiable) we can change elements of a list in place.

Creating lists

```
A=[]
a=[3,5,6,99]
```

Creating empty list:

```
A=list()   # It will generate an empty list and name that list as L.
```

List operations:

1. **Joining lists: We can join/concatenate two or more lists by using + operator.**