

## Class #02: Python Strings:

1. Introduction to strings in Python and how to declare them.
2. Accessing and modifying string values.
3. Using string formatting and escape sequences.
4. Exploring various string functions and operations.
5. Key built-in string methods. **Assignment #2:** String manipulation exercises.

### input() function:

Python-এ input() ফাংশন সব সময় স্ট্রিং আকারে ইনপুট নেয়, এবং এর পেছনে কিছু গুরুত্বপূর্ণ কারণ এবং সুবিধা রয়েছে। আসুন, এটি সহজ উদাহরণ দিয়ে বুঝে নেই।

#### কেনে Python-এ input() সব সময় স্ট্রিং হয়?

1. ইউজারের ইনপুট সাধারণত টেক্স্ট আকারে আসে: যখন আমরা কিছু ইনপুট দিই, তা আসলে কম্পিউটারের কাছে সাধারণত "টেক্স্ট" আকারে যায়, যেমন আমরা কিবোর্ডে যে কী চাপি, তা সরাসরি টেক্স্ট আকারে কম্পিউটারে পৌঁছায়। যেমন: "23", "Hello", "True" ইত্যাদি।
2. সব ধরনের ইনপুটকেই একীভূতভাবে নেওয়া সম্ভব: স্ট্রিং হল একটি সাধারণ ডেটা টাইপ, এবং এটি সব ধরনের ইনপুট ধারণ করতে পারে—সংখ্যা, অক্ষর, প্রতীক ইত্যাদি। এতে, যেকোনো ধরনের ইনপুট নেওয়া সহজ হয় এবং পরে যখন প্রয়োজন, তখন সেই ইনপুটটি অন্য ডেটা টাইপে রূপান্তর (যেমন, সংখ্যা বা দশমিক) করা সম্ভব হয়।

#### এর সুবিধা কি?

1. **নমনীয়তা (Flexibility):** যেহেতু input() সব সময় স্ট্রিং রিটার্ন করে, আপনি যেকোনো ধরনের ডেটা (যেমন সংখ্যা, অক্ষর, ফ্লোট, ইত্যাদি) একটি সাধারণ স্ট্রিং হিসেবে নিয়ে তারপর তা আপনার প্রয়োজন অনুযায়ী কনভার্ট করতে পারেন। অর্থাৎ, ইনপুট নেওয়ার পর আপনি চাইলে তা অক্ষ (যেমন, গাণিতিক হিসাব) বা অন্য কোনো ধরনের ডেটাতে রূপান্তর করতে পারবেন।
2. **ক্রটি নিয়ন্ত্রণে সুবিধা:** স্ট্রিং হিসেবে ইনপুট নেয়া হলে, আপনি চাইলে ইনপুটটি যাচাই (validate) করতে পারবেন। ধরুন, আপনি একটি সংখ্যা চান, কিন্তু ব্যবহারকারী ভুলবশত অক্ষর দিয়েছে। আপনি স্ট্রিংটিকে সহজেই পরীক্ষা করতে পারেন, এবং ক্রটির ক্ষেত্রে ব্যবহারকারীকে একটি সহায়ক বার্তা দিতে পারেন।
3. **ইনপুট প্রক্রিয়াকরণে সহজতা:** যখন ইনপুট স্ট্রিং আকারে থাকে, তখন আপনি স্ট্রিংয়ের বিভিন্ন অংশে সহজে কাজ করতে পারেন। যেমন, ব্যবহারকারী একটি নাম দিলে, আপনি সেই নামের প্রথম অক্ষর, শেষ অক্ষর বা কোন কিছু বের করতে পারবেন, বা যদি নামের মধ্যে কোনো বিশেষ চিহ্ন থাকে, সেগুলো সরিয়ে ফেলতে পারবেন।

#### সহজ উদাহরণ:

ধরা যাক, আপনি একটি সিঙ্গেল নাম ইনপুট নিচ্ছেন। যখন আপনি input() ফাংশন ব্যবহার করবেন, তখন আপনি আসলে একটি স্ট্রিং ইনপুট নেবেন, যেমন: "Alice" বা "Bob"।

কিন্তু, যদি আপনি চান যে ব্যবহারকারী সংখ্যাটি ইনপুট দিন, যেমন "25" (বয়স), তাহলে আপনাকে সেটা স্ট্রিং হিসেবে ইনপুট নিতে হবে। কিন্তু Python আপনাকে সেই "25"-এর মধ্যে কোনো সমস্যা দেখায় না কারণ এটি স্ট্রিং হিসেবে আছেই।

তারপর, আপনি চাইলে সেই স্ট্রিংকে int বা float ডেটা টাইপে রূপান্তর করতে পারেন, যেমন "25" কে একটি পূর্ণসংখ্যা (integer) বা দশমিক সংখ্যা (float) হিসেবে ব্যবহার করতে পারবেন।

এভাবে, Python-এ সব সময় স্ট্রিং ইনপুট নেয়া হয়, যাতে আপনি পরে সেই ইনপুটটি যে কোনো ধরনের ডেটা টাইপে রূপান্তর করে ব্যবহার করতে পারেন।

The screenshot shows the PyCharm IDE interface. The project tree on the left shows a folder named 'MinhazulKabir' containing a 'keras.py' file. The code editor window displays the following Python code:

```

1 a = input("Hi, Minhaz. Enter a Number:")
2 print(type(a))
3

```

The run output window at the bottom shows the execution of the script:

```

"C:\Users\Minhazul Kabir\AppData\Local\Programs\Python\Python313\python.exe" D:\PythonProject\MinhazulKabir\keras.py
Hi, Minhaz. Enter a Number: 15
<class 'str'>
Process finished with exit code 0

```

The status bar at the bottom right indicates Python 3.13, CRLF, UTF-8, 4 spaces, ENG US, and the date/time 12/9/2024 8:50:14 PM.

আপনার কোডে, `input()` ফাংশন ব্যবহার করে একটি ইনপুট নেয়া হচ্ছে। Python-এ `input()` ফাংশন সব সময় স্ট্রিং আকারে ইনপুট গ্রহণ করে, যা সুতরাং `"type(a)"` প্রিন্ট করলে স্ট্রিং টাইপ দেখাবে।

**কেনো আউটপুট স্ট্রিং টাইপ হবে?**

1. **`input()` ফাংশন সবসময় স্ট্রিং আকারে ইনপুট নেয়:**

`input()` ফাংশনটি ব্যবহারকারীর ইনপুটকে টেক্স্ট হিসেবে নেয়, যতই ইনপুট হিসেবে আপনি সংখ্যা দিন না কেন। উদাহরণস্বরূপ, যদি আপনি 5 (সংখ্যা) দেন, তখন Python সেটি একটি স্ট্রিং হিসেবে গ্রহণ করে। এটি একটি টেক্স্ট স্ট্রিং হিসেবে মনে করে, যেমন "5"। এটা একটি সংখ্যা নয়, বরং একটি টেক্স্টের অংশ হিসেবে দেখা হয়।

2. **`type()` ফাংশন:**

`type()` ফাংশনটি একটি ভেরিয়েবলের ডেটা টাইপ বের করে। যেহেতু `a` একটি স্ট্রিং হয়ে ইনপুট নিয়েছে, `type(a)` আউটপুট হিসেবে `str` (স্ট্রিং) দেখাবে।

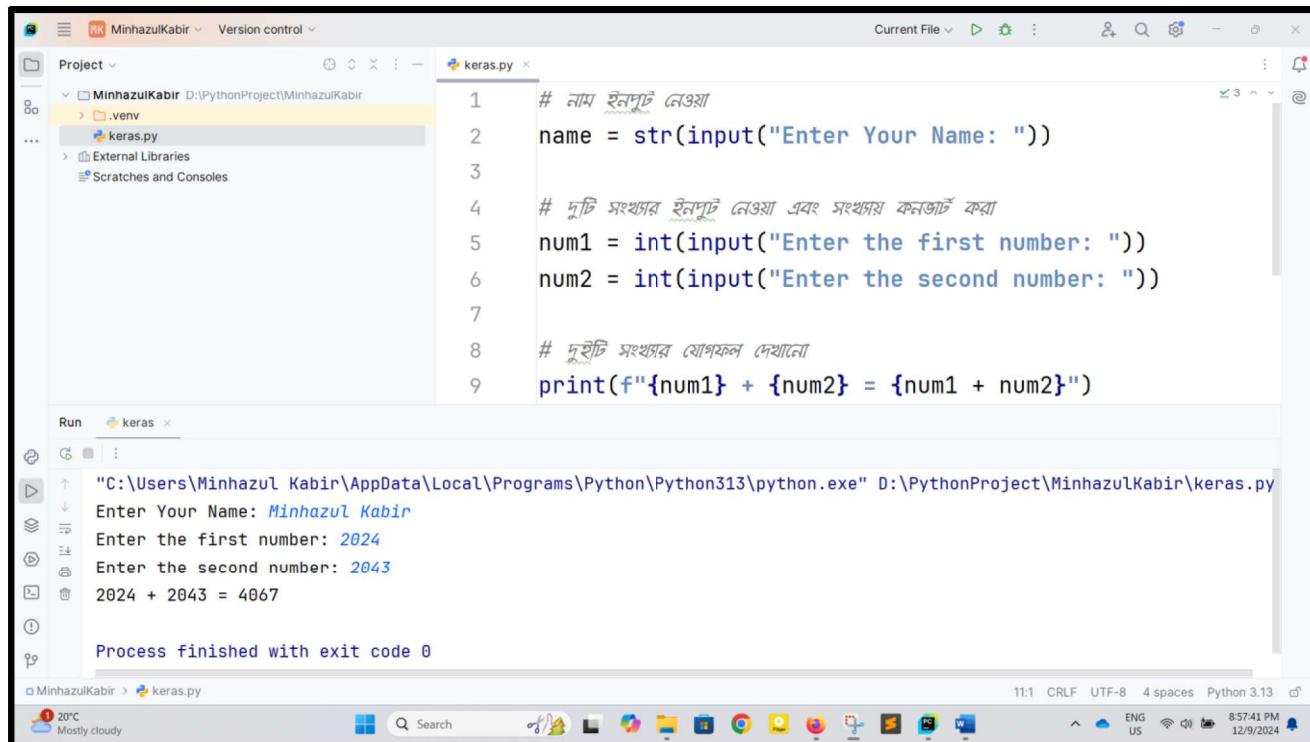
**উদাহরণ:**

ধরা যাক, আপনি ইনপুট হিসেবে 5 দিয়েছেন। তবে Python সেটি স্ট্রিং হিসেবে মনে করবে, এর মানে "5" (কোটেশন চিহ্ন)।

**এটি স্ট্রিং কেন?**

- Python স্ট্রিং হিসেবে ইনপুট নেয় কারণ কিবোর্ড থেকে যে কোনো তথ্য আসবে, তা সবসময় টেক্সট হিসেবে পাঠানো হয়। এমনকি আপনি সংখ্যাও ইনপুট দিন, তা স্ট্রিং আকারে স্টোর হবে, পরে প্রয়োজনে আপনি সেটি ইন্টিজার (integer) বা ফ্লোট (float) ইত্যাদিতে কনভার্ট করতে পারবেন।

## Program and explanation:



```

1 # নাম ইনপুট নেওয়া
2 name = str(input("Enter Your Name: "))
3
4 # দুটি সংখ্যার ইনপুট নেওয়া এবং সংখ্যার কনভার্ট করা
5 num1 = int(input("Enter the first number: "))
6 num2 = int(input("Enter the second number: "))
7
8 # দুইটি সংখ্যার যোগফল দেখানো
9 print(f"{num1} + {num2} = {num1 + num2}")

```

"C:\Users\Minhzul Kabir\AppData\Local\Programs\Python\Python313\python.exe" D:\PythonProject\MinhzulKabir\keras.py

Enter Your Name: *Minhzul Kabir*

Enter the first number: *2024*

Enter the second number: *2043*

*2024 + 2043 = 4067*

Process finished with exit code 0

### প্রোগ্রাম ব্যাখ্যা:

- str() ব্যবহার না করা:** `input()` ফাংশন নিজেই স্ট্রিং আকারে ইনপুট নেয়, তাই `str()` দিয়ে আবার ইনপুট নেওয়ার প্রয়োজন নেই।
- ভেরিয়েবল নাম:** কোডে `num1` ও `num2` যথেষ্ট স্পষ্ট, তাই কোনও পরিবর্তন করা হয়নি। তবে, আপনার কোডটি আরও পরিষ্কার করার জন্য সাধারণত স্পষ্ট ও বোধগম্য নাম ব্যবহার করা হয় (যেমন, `first_number`, `second_number` ইত্যাদি)। তবে, এই ক্ষেত্রে `num1` এবং `num2` যথেষ্ট।
- মন্তব্য যোগ করা:** কোডের প্রতিটি অংশের জন্য মন্তব্য যোগ করা হয়েছে যাতে কোডটি বুবলে সহজ হয়।

### Output ব্যাখ্যা:

- প্রথম সংখ্যা:**
  - আপনি প্রথম ইনপুট হিসেবে `2024` দিয়েছেন। এই ইনপুটটি `input()` ফাংশনের মাধ্যমে স্ট্রিং আকারে নেয়া হয়, কিন্তু পরে এটি `int()` ফাংশন দিয়ে পূর্ণসংখ্যায় রূপান্তরিত হয়েছে।
  - তাই, `num1 = 2024`।
- দ্বিতীয় সংখ্যা:**
  - আপনি দ্বিতীয় ইনপুট হিসেবে `2043` দিয়েছেন। এটি ঠিক একইভাবে প্রথম সংখ্যার মতো স্ট্রিং থেকে `int()` ফাংশন ব্যবহার করে পূর্ণসংখ্যায় রূপান্তরিত হয়েছে।

- ০ তাই, num2 = 2043।

### 3. যোগফল:

- ০ `print(f"{num1} + {num2} = {num1 + num2}")` এই লাইনে দুটি সংখ্যার যোগফল দেখানো হয়েছে।
- ০ এটি আসলে: `2024 + 2043 = 4067`।
- ০ এখানে, `num1 + num2` হলো `2024 + 2043`, যার মান `4067`।

অতএব, আউটপুট "`2024 + 2043 = 4067`" দেখানো হয়েছে, যা দুটি সংখ্যার যোগফল।

[https://www.w3schools.com/python/python\\_strings.asp](https://www.w3schools.com/python/python_strings.asp) পাইথনের স্ট্রিং শিখার জন্য গুরুত্বপূর্ণ resource. এখান, থেকে আমি পাইথনের string এর ব্যাপারে অনেক জ্ঞান লাভ করি। আশাকরি, সবার কাজে আসবে।

## Introduction to strings in Python and how to declare them.

Python এ string নিয়ে কাজ করা খুব গুরুত্বপূর্ণ এবং সাধারণ কারণ, string প্রায় সব প্রোগ্রামে ব্যবহৃত হয়। এটি একটি ডাটা টাইপ যা অক্ষরের সিকোয়েন্স (sequence) প্রকাশ করে। Python এ string এর সাহায্যে বিভিন্ন ধরণের কাজ করা হয়। Single quote এবং double quote উভয়ই একইভাবে কাজ করে। তবে, যদি আপনার string-এর মধ্যে একটি quotation mark ব্যবহার করতে হয়, তাহলে অন্যটি ব্যবহার করলে আপনি escape character (\) ব্যবহার না করেও এটি করতে পারবেন।

### উদাহরণ:

```
# Single quote inside double quotes
message = "It's a beautiful day."
print(message)
```

```
# Double quote inside single quotes
message = 'He said, "Hello!"'
print(message)
```

```
It's a beautiful day.
He said, "Hello!"
```

## Accessing and modifying string values.

**String Value Access:** পাইথনে, strings একটি **immutable** (অপরিবর্তনীয়) data type, যার মানে হলো একবার string তৈরি হলে, তার মান পরিবর্তন করা যায় না। তবে, আপনি string এর একটি নির্দিষ্ট character বা অংশ access করতে পারেন। এটি **indexing** বা **slicing** ব্যবহার করে করা হয়। **প্রতিটি character এর একটি index থাকে, যা ০ থেকে শুরু হয়।**

- **Indexing:** string এর প্রতিটি character এর নির্দিষ্ট অবস্থান (index) দিয়ে আপনি সেই character টি access করতে পারেন।

```

1 b = 'bangladesh'
2 print(len(b))
3 print(b[0])
4 print(b[10])

```

"C:\Users\Minhzul Kabir\AppData\Local\Programs\Python\Python313\python.exe" D:\PythonProject\MinhzulKabir\keras.py

10  
b

Traceback (most recent call last): Explain with AI  
File "D:\PythonProject\MinhzulKabir\keras.py", line 4, in <module>  
print(b[10])  
~~~~~~  
IndexError: string index out of range

Process finished with exit code 1

2. len(b) ফাংশনটি স্ট্রিং b এর দৈর্ঘ্য বের করে, অর্থাৎ স্ট্রিংটিতে মোট কতটি অক্ষর আছে। উদাহরণস্বরূপ, যদি b = 'bangladesh' হয়, তাহলে এর দৈর্ঘ্য হবে 10, কারণ "bangladesh" স্ট্রিংয়ে মোট 10টি অক্ষর রয়েছে। ফলে len(b) এর আউটপুট হবে 10।
3. স্ট্রিংয়ের প্রথম অক্ষরটি পাইথনে index 0 তে থাকে, কারণ পাইথনের ইনডেক্সিং 0 থেকে শুরু হয়। তাই b[0] স্ট্রিং b এর প্রথম অক্ষর 'b' রিটার্ন করবে।
4. তবে, স্ট্রিং b এর ইনডেক্সগুলি 0 থেকে 9 পর্যন্ত থাকে, কারণ এর দৈর্ঘ্য 10, অর্থাৎ b তে মোট 10টি অক্ষর আছে। তাই b[10] চেষ্টা করলে এটি স্ট্রিংয়ের বাইরে (out of range) গিয়ে একটি **IndexError** ঘটাবে, কারণ index 10 তে কোনো অক্ষর নেই।

- **Slicing:** একাধিক character বা string এর একটি অংশ একসাথে নিয়ে আসা যায় slicing এর মাধ্যমে।

পাইথনে slicing ব্যবহার করে আপনি স্ট্রিংয়ের একটি অংশ (substring) বের করতে পারেন। এর জন্য স্ট্রিংয়ের শুরু এবং শেষ ইনডেক্স নির্দিষ্ট করা হয়।

## উদাহরণ:

ধরা যাক, আমাদের স্ট্রিং `b = 'bangladesh'` এবং আমরা এটি থেকে কিছু অংশ বের করতে চাই।

The screenshot shows a PyCharm IDE window with a project named 'MinhazulKabir'. The code editor contains a file named 'keras.py' with the following content:

```

1 b = 'bangladesh'
2
3 # স্ট্রিংয়ের প্রথম 5টি অক্ষর বের করা
4 print(b[:5]) # আউটপুট: 'bang'
5
6 # স্ট্রিংয়ের 3য় ইনডেক্স থেকে শেষ দর্শক অংশ বের করা
7 print(b[3:]) # আউটপুট: 'gladesh'
8
9 # স্ট্রিংয়ের 2য় ইনডেক্স থেকে 7ম ইনডেক্স মর্ত্তি অংশ বের করা
10 print(b[2:7]) # আউটপুট: 'nglad'
11
12 # স্ট্রিংয়ের শেষ 5টি অক্ষর বাদে বাকি অক্ষর (-1 হচ্ছে h এবং -5 হচ্ছে a)
13 print(b[-5:-1]) # আউটপুট: 'ades'

```

The run tab shows the output of the code:

```

"python.exe" D:\PythonProject\MinhazulKabir\keras.py
bangl
gladesh
nglad
ades
Process finished with exit code 0

```

At the bottom right, it says: 1:17 CRLF UTF-8 4 spaces Python 3.13

## ব্যাখ্যা:

1. `b[:5]`: স্ট্রিংয়ের প্রথম 5টি অক্ষর (ইনডেক্স 0 থেকে 4 পর্যন্ত)। আউটপুট হবে 'bangl'।
2. `b[3:]`: স্ট্রিংয়ের 3য় ইনডেক্স থেকে শুরু করে শেষ পর্যন্ত সব অক্ষর বের করা। আউটপুট হবে 'gladesh'।
3. `b[2:7]`: স্ট্রিংয়ের 2য় ইনডেক্স থেকে 7ম ইনডেক্সের আগ পর্যন্ত অংশ বের করা। আউটপুট হবে 'nglad'।
4. `b[-5:-1]`: স্ট্রিংয়ের শেষ 5টি অক্ষর বাদে বাকি অক্ষর বের করা। আউটপুট হবে 'ades'।

Index: 0 1 2 3 4 5 6 7 8 9 10

`b = b a n g l a d e s h`

**String Value Modification:** পাইথনে string পরিবর্তন করা যায় না কারণ string গুলি **immutable**। তবে, আপনি string এর নতুন মান তৈরি করতে পারেন, যেমন নতুন character যোগ করে বা পুরোনো character সরিয়ে। এভাবে, string এর মানকে সরাসরি পরিবর্তন না করতে পারলেও, নতুন string তৈরি করে তার মান পরিবর্তন করা সম্ভব।

## String Methods (স্ট্রিং মেথডস) ব্যাখ্যা

### 1. `lower()`:

- ০ **ব্যাখ্যা:** এই মেথডটি স্ট্রিংয়ের সব অক্ষরকে ছোট হাতের অক্ষরে (lowercase) রূপান্তরিত করে।
- ০ **উদাহরণ:** "Hello".lower() রিটার্ন করবে "hello"।

## 2. `upper()`:

- **ব্যাখ্যা:** এই মেথডটি স্ট্রিংয়ের সব অক্ষরকে বড় হাতের অক্ষরে (uppercase) রূপান্তরিত করে।
- **উদাহরণ:** "hello".upper() রিটার্ন করবে "HELLO"।

## 3. `capitalize()`:

- **ব্যাখ্যা:** এই মেথডটি স্ট্রিংয়ের প্রথম অক্ষরকে বড় হাতের অক্ষরে রূপান্তরিত করে এবং বাকি সব অক্ষরকে ছোট হাতের অক্ষরে রূপান্তরিত করে।
- **উদাহরণ:** "hello world".capitalize() রিটার্ন করবে "Hello world"।

## 4. `replace(old, new)`:

- **ব্যাখ্যা:** এই মেথডটি স্ট্রিংয়ের মধ্যে থাকা পুরানো substring (old) কে নতুন substring (new) দিয়ে প্রতিস্থাপন করে।
- **উদাহরণ:** "hello world".replace("world", "Python") রিটার্ন করবে "hello Python"।

## 5. `split()`:

- **ব্যাখ্যা:** এই মেথডটি স্ট্রিংটিকে একটি তালিকায় (list) ভাগ করে, যেটি একটি delimiter (ডিফল্টভাবে space) দ্বারা বিভক্ত হয়।
- **উদাহরণ:** "hello world".split() রিটার্ন করবে ['hello', 'world']।

## 6. `strip()`:

- **ব্যাখ্যা:** এই মেথডটি স্ট্রিংয়ের শুরু এবং শেষের অতিরিক্ত (leading and trailing) স্পেস মুছে ফেলে।
- **উদাহরণ:** " hello world ".strip() রিটার্ন করবে "hello world"।

## 7. `find(substring)`:

- **ব্যাখ্যা:** এই মেথডটি স্ট্রিংয়ের মধ্যে একটি নির্দিষ্ট substring খুঁজে তার প্রথম occurrence এর ইনডেক্স রিটার্ন করে। যদি substring না পাওয়া যায়, তবে -1 রিটার্ন হয়।
- **উদাহরণ:** "hello world".find("world") রিটার্ন করবে 6 (কারণ "world" স্ট্রিংয়ের 6 নম্বর ইনডেক্সে শুরু হয়)।

## 8. `count(substring)`:

- **ব্যাখ্যা:** এই মেথডটি স্ট্রিংয়ের মধ্যে একটি নির্দিষ্ট substring এর occurrences (গণনা) রিটার্ন করে।
- **উদাহরণ:** "hello hello world".count("hello") রিটার্ন করবে 2 (যেহেতু "hello" দুইবার এসেছে)।

## 9. `join(iterable)`:

- **ব্যাখ্যা:** এই মেথডটি একটি iterable (যেমন list) এর সব elements কে স্ট্রিং এর মাধ্যমে যুক্ত করে একটি নতুন স্ট্রিং তৈরি করে।

- উদাহরণ: `"-".join(["hello", "world"])` রিটার্ন করবে "hello-world"। এখানে `"-"` স্ট্রিংটি separator হিসেবে ব্যবহার হচ্ছে।

#### 10. `startswith(prefix)`:

- ব্যাখ্যা:** এই মেথডটি চেক করে যে স্ট্রিংটি নির্দিষ্ট prefix দিয়ে শুরু হয় কিনা। যদি হয়, তাহলে True রিটার্ন হয়, অন্যথায় False।
- উদাহরণ: `"hello world".startswith("hello")` রিটার্ন করবে True।

#### 11. `endswith(suffix)`:

- ব্যাখ্যা:** এই মেথডটি চেক করে যে স্ট্রিংটি নির্দিষ্ট suffix দিয়ে শেষ হয় কিনা। যদি হয়, তাহলে True রিটার্ন হয়, অন্যথায় False।
- উদাহরণ: `"hello world".endswith("world")` রিটার্ন করবে True।

এই মেথডগুলো স্ট্রিংয়ের সাথে বিভিন্ন ধরনের অপারেশন করতে সাহায্য করে, যেমন স্ট্রিংয়ের কনভার্সন, অংশ বের করা, অংশ প্রতিস্থাপন, বা খোঁজা ইত্যাদি।

```
name = ' minhazuL kabiR'
```

```
# lower() - সব অক্ষরকে lowercase (ছোট হাতের অক্ষর) তে রূপান্তর
print(name.lower()) # আউটপুট: ' minhazul kabir'
```

```
# upper() - সব অক্ষরকে uppercase (বড় হাতের অক্ষর) তে রূপান্তর
print(name.upper()) # আউটপুট: ' MINHAZUL KABIR'
```

```
# capitalize() - প্রথম অক্ষরকে বড় হাতের অক্ষরে রূপান্তর এবং বাকি অক্ষর ছোট হাতের অক্ষরে রূপান্তর
print(name.capitalize()) # আউটপুট: ' minhazul kabir'
```

```
# replace(old, new) - "old" substring কে "new" substring দিয়ে প্রতিস্থাপন
print(name.replace("L", "l")) # আউটপুট: ' minhazul kabir'
```

```
# split() - স্ট্রিংটি স্পেস দিয়ে ভাগ করা (ডিফল্ট delimiter হিসেবে space)
print(name.split()) # আউটপুট: ['minhazuL', 'kabiR']
```

```
# strip() - স্ট্রিংয়ের শুরু এবং শেষের স্পেস মুছে ফেলা
print(name.strip()) # আউটপুট: 'minhazuL kabiR'
```

```
# find(substring) - substring এর প্রথম occurrence এর ইনডেক্স খোঁজা
print(name.find("L")) # আউটপুট: 8 (কারণ "L" প্রথমবার 8 নম্বর ইনডেক্সে এসেছে)
```

```
# count(substring) - substring এর occurrences (গণনা) বের করা।
print(name.count("a")) # আউটপুট: 3 (কারণ "a" তিনটি বার এসেছে)
```

```
minhazul kabir
MINHAZUL KABIR
minhazul kabir
minhazul kabiR
['minhazuL', 'kabiR']
minhazuL kabiR
10
2
```

এখানে name = ' minhazuL kabiR' স্ট্রিংয়ের সাথে প্রোগ্রামের মাধ্যমে প্রতিটি স্ট্রিং মেথডের আউটপুটের ব্যাখ্যা দেওয়া হলো:

#### 1. lower():

- আউটপুট: ' minhazul kabir'
- **ব্যাখ্যা:** lower() মেথড স্ট্রিংটির সব অক্ষরকে ছোট হাতের অক্ষরে রূপান্তরিত করে। তবে, স্ট্রিংয়ের শুরু এবং শেষে থাকা অতিরিক্ত স্পেস পরিবর্তিত হয়নি, কারণ strip() মেথড ব্যবহার করা হয়নি।

#### 2. upper():

- আউটপুট: ' MINHAZUL KABIR'
- **ব্যাখ্যা:** upper() মেথড স্ট্রিংটির সব অক্ষরকে বড় হাতের অক্ষরে রূপান্তরিত করে। এই পরিবর্তনেও স্ট্রিংয়ের শুরু এবং শেষে থাকা অতিরিক্ত স্পেস অপরিবর্তিত থাকে।

#### 3. capitalize():

- আউটপুট: ' minhazul kabir'
- **ব্যাখ্যা:** capitalize() মেথড স্ট্রিংয়ের প্রথম অক্ষরকে বড় হাতের অক্ষরে এবং বাকি অক্ষরগুলোকে ছোট হাতের অক্ষরে রূপান্তরিত করে। তবে, এই মেথড স্ট্রিংয়ের শুরু এবং শেষে থাকা স্পেস পরিবর্তন করে না।

#### 4. replace(old, new):

- আউটপুট: ' minhazul kabir'
- **ব্যাখ্যা:** এখানে "L" কে "l" দিয়ে প্রতিস্থাপন করা হয়েছে। স্ট্রিংয়ের "L" এর পরিবর্তে "l" এসেছে, কিন্তু এটি স্ট্রিংয়ের পরিবর্তনযোগ্য অক্ষর হওয়ায় এই পরিবর্তন কার্যকর হয়।

#### 5. split():

- আউটপুট: ['minhazuL', 'kabiR']
- **ব্যাখ্যা:** split() মেথডটি ডিফল্টভাবে স্ট্রিংটিকে স্পেস দিয়ে বিভক্ত করে একটি লিস্টে রূপান্তরিত করে। স্ট্রিংয়ের মধ্যে স্পেস থাকার কারণে এটি দুটি আলাদা উপাদানে বিভক্ত হয়ে ['minhazuL', 'kabiR'] লিস্টে পরিণত হয়।

#### 6. strip():

- আউটপুট: 'minhazuL kabiR'
- **ব্যাখ্যা:** strip() মেথডটি স্ট্রিংয়ের শুরু এবং শেষে থাকা স্পেস সরিয়ে দেয়। তবে, স্ট্রিংয়ের মধ্যে স্পেস অপরিবর্তিত থাকে।

#### 7. find(substring):

- আউটপুট: 10
- **ব্যাখ্যা:** find() মেথডটি প্রথম যে ইনডেক্সে "L" চরিত্রটি এসেছে তা খুঁজে পায়। "L" প্রথমবার 10 নম্বর ইনডেক্সে পাওয়া যায়, তাই আউটপুট 10।

#### 8. count(substring):

- আউটপুট: 2 (minhAzul Kabir)
- **ব্যাখ্যা:** count() মেথডটি স্ট্রিংয়ে "a" চরিত্রটি কতবার এসেছে তা গননা করে। "a" চরিত্রটি স্ট্রিংয়ে 2 বার এসেছে, তাই আউটপুট 2।

## Using string formatting and escape sequences

ফর্ম্যাট স্ট্রিং (f-string) হলো পাইথনে একটি আধুনিক এবং সহজ পদ্ধতি যা স্ট্রিংয়ের মধ্যে সরাসরি ভেরিয়েবল বা এক্সপ্রেশন ইনপুট করার সুযোগ দেয়। এটি পাইথনের 3.6 ভার্সন থেকে চালু করা হয়েছে এবং এটি কোড লেখার সময় অনেক সুবিধা প্রদান করে।

কেনো f-string ব্যবহার করা হয়?

1. **পরিষ্কার কোড:** f-string ব্যবহারে কোড খুবই পরিষ্কার এবং সহজ হয়ে যায়। এতে স্ট্রিং এবং ভেরিয়েবলকে একসাথে মিশিয়ে দেওয়া যায়, এবং এই কারণে অতিরিক্ত অপারেটর (যেমন, +) ব্যবহার করার প্রয়োজন পড়ে না। ফলে কোড সংক্ষিপ্ত ও আরও পাঠ্যোগ্য হয়ে ওঠে।
2. **সহজ ভেরিয়েবল ইনপুট:** f-string এর মাধ্যমে সরাসরি {} ৱ্রেসেসের মধ্যে ভেরিয়েবল বা এক্সপ্রেশন ব্যবহার করা যায়। এতে কোডের মধ্যে কোনো জটিলতা থাকে না, এবং এটি খুব দ্রুত কাজ করে।
3. **ইনলাইন এক্সপ্রেশন ব্যবহার:** f-string এর মাধ্যমে আপনি স্ট্রিংয়ের মধ্যে শুধু ভেরিয়েবল নয়, বরং কোনো এক্সপ্রেশনও ব্যবহার করতে পারেন। যেমন, আপনি সরাসরি কোনো গাণিতিক হিসাব বা ফাংশন কলও সেখানে করতে পারেন।
4. **দ্রুত কার্যকরী:** অন্যান্য পদ্ধতির তুলনায় f-string দ্রুত এবং বেশি কার্যকরী। কারণ এটি স্ট্রিং তৈরির সময় সরাসরি ভেরিয়েবলগুলোর মান সন্নিবেশ করে, যা সময় সাধারণ।

উপকারিতা:

- কমপ্যাট কোড:** f-string ব্যবহারের মাধ্যমে কোড অনেক সংক্ষিপ্ত হয়, কারণ এতে স্ট্রিং যোগ করার জন্য আলাদা করে প্লাস (+) ব্যবহার করতে হয় না।
- সহজ পড়া:** একসাথে স্ট্রিং এবং ভেরিয়েবল থাকতে পারে, তাই কোডটি পড়তে সহজ হয়। বিশেষ করে বড় প্রোগ্রামে এটি খুব কার্যকরী।
- বিভিন্ন ধরনের এক্সপ্রেশন সাপোর্ট:** f-string এর মাধ্যমে আপনি কেবল ভেরিয়েবলই নয়, বরং গাণিতিক হিসাব, ফাংশন কল, বা কোনো কভিশনও স্ট্রিংয়ের মধ্যে সম্মিলিত করতে পারেন।

### উদাহরণ:

ধরা যাক, আপনি আপনার নাম এবং বয়স একটি স্ট্রিংয়ের মধ্যে একসাথে দেখতে চান, তাহলে f-string এর মাধ্যমে তা সহজেই করা সম্ভব, যেমন:

- পুরনো পদ্ধতিতে আপনাকে স্ট্রিং যোগ করতে প্লাস সাইন (+) ব্যবহার করতে হতো, কিন্তু f-string এ আপনি সরাসরি {} এর মধ্যে ভেরিয়েবল রাখতে পারেন।

### উপসংহার:

ফরম্যাট স্ট্রিং (f-string) পাইথনে কোড লেখার এক সহজ, পরিষ্কার এবং কার্যকরী পদ্ধতি। এটি স্ট্রিংয়ের মধ্যে ভেরিয়েবল এবং এক্সপ্রেশন সহজে যুক্ত করার সুযোগ দেয় এবং কোডে কমপ্যাটনেস, পাঠ্যোগ্যতা এবং গতি নিশ্চিত করে।

```
# নাম এবং বয়সের পরিবর্তনযোগ্য মান
name = "Minhazul Kabir"
age = 29.4

# কমা ব্যবহার করে নাম এবং বয়স আক্রমণিক করা
# কমা ব্যবহার করলে স্বয়ংক্রিয়ভাবে স্পেস যোগ হয়
print("My name is", name, "& my age is", age)

# প্লাস সাইন (+) ব্যবহার করে নাম এবং বয়স যোগ করা
# এখানে স্পেস যোগ করা হয় না, সরাসরি স্ট্রিং যোগ হয়
print("My name is" + name)

# ফরম্যাট স্ট্রিং (f-string) ব্যবহার করা
# এটি প্রস্তুতিমূলকভাবে সহজ এবং কোডে আরও পরিকল্পনা আক্রমণিক দেয়
print(f"My name is {name} & my age is {age}")

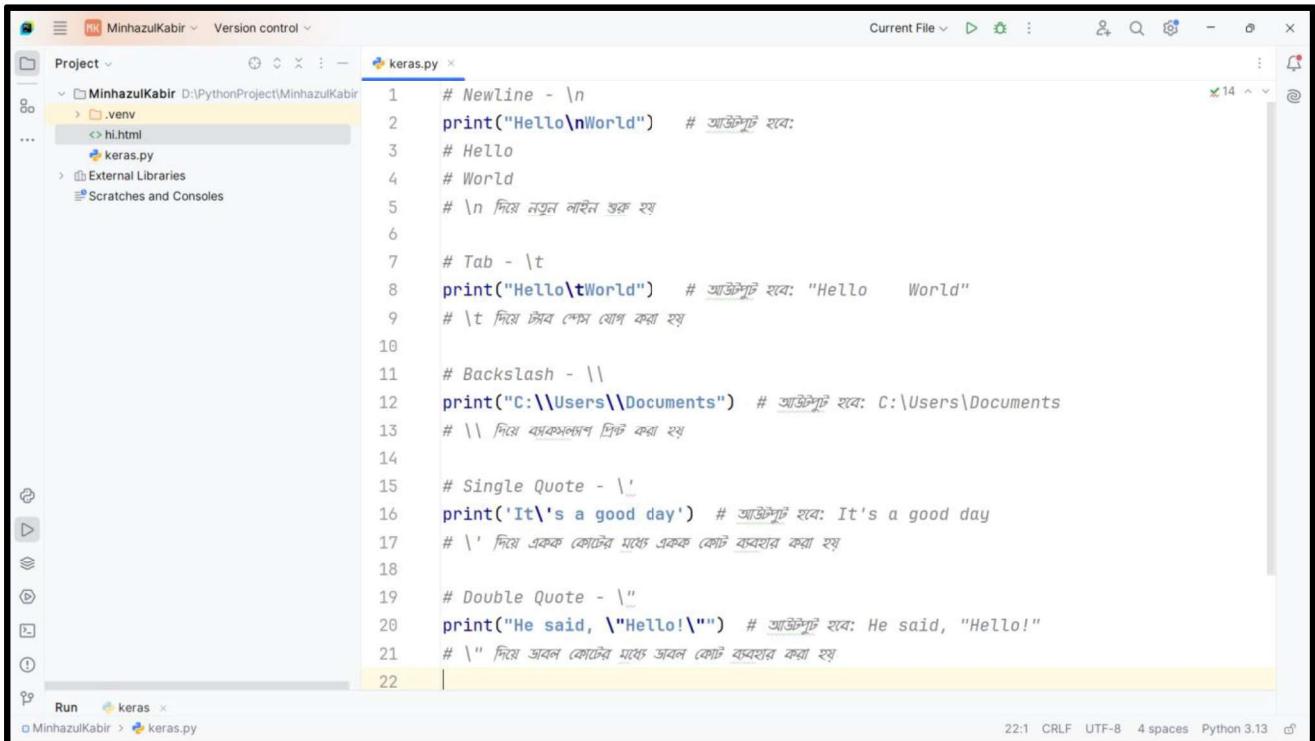

```

- কমা ব্যবহার করে নাম এবং বয়স আউটপুট করা:** যখন আমরা পাইথনে print ফাংশনে কমা ব্যবহার করি, তখন কমা দ্বারা আলাদা করা প্রতিটি উপাদান (যেমন, স্ট্রিং, ভেরিয়েবল, সংখ্যা) স্বয়ংক্রিয়ভাবে একটি স্পেস দ্বারা আলাদা হয়ে আউটপুটে প্রদর্শিত হয়। এর মানে হলো, কোডে আলাদা আলাদা অংশ যোগ করলেও তারা একে অপর থেকে স্পেস দিয়ে আলাদা হয়ে যায়। উদাহরণস্বরূপ, print("My name is", name, "& my age is", age) এই কোডে "My name is" এবং name ভেরিয়েবল, এবং "& my age is" এবং age ভেরিয়েবলটি আলাদা করে প্রদর্শিত হয় এবং তাদের মধ্যে একটি স্পেস যুক্ত হয়।

2. **প্লাস সাইন (+) ব্যবহার করে স্ট্রিং যোগ করা:** প্লাস সাইন ব্যবহার করলে দুটি স্ট্রিং একসাথে যোগ করা হয়। তবে, এর মধ্যে কোনো স্পেস স্বয়ংক্রিয়ভাবে যুক্ত হয় না। আপনি যদি স্পেস চান, তবে আপনাকে তা নিজেই যুক্ত করতে হবে। উদাহরণস্বরূপ, `print("My name is" + name)` এই কোডটি সরাসরি "My name is" এবং name ভেরিয়েবলটি একসাথে যোগ করে, কিন্তু এখনে কোনো স্পেস থাকবে না। তাই, যদি আপনি চাইছেন যে এই স্ট্রিংটি পরিপূর্ণভাবে সঠিকভাবে দেখাক, তবে আপনাকে স্পেসটি নিজে যোগ করতে হবে, যেমন "My name is " + name।

3. **ফরম্যাট স্ট্রিং (f-string) ব্যবহার করা:** ফরম্যাট স্ট্রিং (f-string) একটি নতুন এবং আধুনিক পদ্ধতি, যা স্ট্রিংয়ের মধ্যে সরাসরি ভেরিয়েবল বা এক্সপ্রেশন রাখার সুবিধা দেয়। এতে কোড খুবই পরিষ্কার এবং সহজ হয়ে যায়। উদাহরণস্বরূপ, `print(f"My name is {name} & my age is {age}")` এই কোডটি সরাসরি {} ৰেসের মধ্যে name এবং age ভেরিয়েবলটির মান সন্তুষ্ট করে, ফলে এটি দ্রুত এবং সহজভাবে স্ট্রিং আউটপুট তৈরি করে। f-string ব্যবহারে আপনি অনেক দ্রুত এবং কমপ্যাক্টভাবে আউটপুট পেতে পারেন, যা কোডকে আরও পরিষ্কার করে তোলে।

**Escape sequences** হল বিশেষ অক্ষরের সমষ্টি, যা স্ট্রিংয়ের মধ্যে নির্দিষ্ট কাজ করতে ব্যবহৃত হয়। পাইথনে, এই escape sequences \ (ব্যাকসল্যাশ) চিহ্ন দিয়ে শুরু হয় এবং এটি সাধারণত স্ট্রিংয়ের মধ্যে বিশেষ অক্ষর তৈরি বা আচরণ পরিবর্তন করতে ব্যবহৃত হয়। নিচে কিছু সাধারণ escape sequences এর ব্যবহার এবং ব্যাখ্যা দেওয়া হলো:



```

1 # Newline - \n
2 print("Hello\nWorld") # আউটপুট হবে:
3 # Hello
4 # World
5 # \n সিলে নতুন লাইন ড্রপ হয়
6
7 # Tab - \t
8 print("Hello\tWorld") # আউটপুট হবে: Hello      World
9 # \t সিলে টেবল স্পেস যোগ করা হয়
10
11 # Backslash - \\
12 print("C:\\Users\\\\Documents") # আউটপুট হবে: C:\\Users\\Documents
13 # \\ সিলে ব্যাকসল্যাশ স্পিষ্ট করা হয়
14
15 # Single Quote - \
16 print('It\'s a good day') # আউটপুট হবে: It's a good day
17 # \' সিলে একক কোটের মধ্যে একক কোট ব্যবহার করা হয়
18
19 # Double Quote - \
20 print("He said, \"Hello!\"") # আউটপুট হবে: He said, "Hello!"
21 # \" সিলে অবল কোটের মধ্যে অবল কোট ব্যবহার করা হয়
22

```

Hello

World

Hello World

C:\\Users\\Documents

It's a good day

He said, "Hello!"

### ব্যাখ্যা:

1. **\n (Newline):** এটি একটি নতুন লাইন শুরু করে। যখন এটি স্ট্রিংয়ের মধ্যে ব্যবহৃত হয়, তখন যে অংশটি এর পরে থাকে তা নতুন লাইনে চলে আসে।
2. **\t (Tab):** এটি ট্যাব স্পেস যোগ করে। এটি সাধারণত ফর্ম্যাটিং এবং টেবুলার ডেটা সাজানোর জন্য ব্যবহৃত হয়।
3. **\\" (Backslash):** ব্যাকসল্যাশ স্ট্রিংয়ে প্রিন্ট করতে হলে \। ব্যবহার করতে হয়, কারণ একক ব্যাকসল্যাশটি escape character হিসেবে ব্যবহৃত হয়।
4. **\' (Single Quote):** যখন আপনি স্ট্রিংয়ের মধ্যে একক কোটি ব্যবহার করতে চান, তখন \' ব্যবহার করতে হয়, যাতে একক কোটিটি স্ট্রিংয়ের অংশ হিসেবে দেখা যায়।
5. **\\" (Double Quote):** একইভাবে, ডাবল কোটি স্ট্রিংয়ের মধ্যে ব্যবহারের জন্য \" ব্যবহার করা হয়।

এই escape sequences গুলি আপনাকে স্ট্রিংয়ের মধ্যে বিশেষ অক্ষর বা কার্যকারিতা সঠিকভাবে ব্যবহারের সুবিধা দেয়।

**String Concatenation** হলো দুটি বা ততোধিক স্ট্রিং একত্রিত করার প্রক্রিয়া। এটি স্ট্রিংয়ের শেষে অন্য একটি স্ট্রিং যোগ করার মাধ্যমে সম্পন্ন হয়। Python এ, স্ট্রিং কনক্যাটেনেট করার জন্য সাধারণত + অপারেটর ব্যবহার করা হয়।

```

# String Concatenation using + operator
# Defining two strings
first_name = "Minhazul"
last_name = "Kabir"
# Concatenating two strings with a space in between
full_name = first_name + " " + last_name
# Printing the result
print(full_name) # Output: Minhazul Kabir

# Concatenating numbers as strings
num1 = "2024"
num2 = "2043"
# Concatenating numbers
result = num1 + num2
# Printing the result
print(result) # Output: 20242043

```

যেমন:

- যদি আপনার দুটি স্ট্রিং থাকে, "Hello" এবং "World", এবং আপনি এগুলিকে একত্রিত করতে চান, তাহলে ফলাফল হবে "HelloWorld"।

**উদাহরণ:**

1. String Concatenation using + operator:

- first\_name = "Minhazul"
- last\_name = "Kabir"
- full\_name = first\_name + " " + last\_name
- আউটপুট হবে: "Minhazul Kabir"

2. Concatenation of numbers as strings:

- num1 = "2024"
- num2 = "2043"
- result = num1 + num2
- আউটপুট হবে: "20242043"

এছাড়া, স্ট্রিং কনক্যাটেনেশনের জন্য join() মেথডও ব্যবহার করা যেতে পারে, যা একটি iterable (যেমন একটি লিস্ট) থেকে স্ট্রিংগুলো যোগ করার জন্য ব্যবহৃত হয়।

**উপকারিতা:**

- স্ট্রিং কনক্যাটেনেশন দিয়ে আপনি সহজেই একাধিক টুকরো স্ট্রিং একত্রিত করতে পারেন।
- এটি বিশেষ করে ব্যবহারী ইন্টারফেস বা বার্তা প্রেরণের সময় সাহায্যকারী।