



০০১-০০৭ পাইথনের কিছু বেসিক কোড (পর্ব-১)

Statistics কে কিছুটা বিরতি রেখে আমরা এখন পাইথন ধরছি, পরে আবার আমরা *Statistics continue* করবো।

পাইথনে ইন্সটলের সময় আমরা দেখেছি যে VS Code এ কিভাবে পাইথন ফাইল তৈরি করতে হয়, এবং কিভাবে terminal এর CMD দিয়ে পাইথন ফাইলের কোড রান করতে হয়।

তাহলে শুরু করি বেসিক কোড লার্নিং।

ভেরিয়েবল আর প্রিন্টঃ

পাইথনে ভেরিয়েবল ডিক্লারেশনের সময় ডেটাটাইপ লেখা লাগে না। যেমনঃ C Programming এ ভেরিয়েবল ডিক্লারেশনের সময় এভাবে লেখা লাগতঃ- `int a = 50;` অর্থাৎ ভেরিয়েবল ডিক্লারেশনের সময় ডেটাটাইপ লেখা লাগতো। কিন্তু Python এ সেই জিনিসটা এভাবে লেখা লাগে `a = 50`। এর মানে পাইথনে ভেরিয়েবল ডিক্লারেশনের সময় ডেটাটাইপ কষ্ট করে লেখা লাগে না।

```
int a = 50;
```

C Program

```
a = 50
```

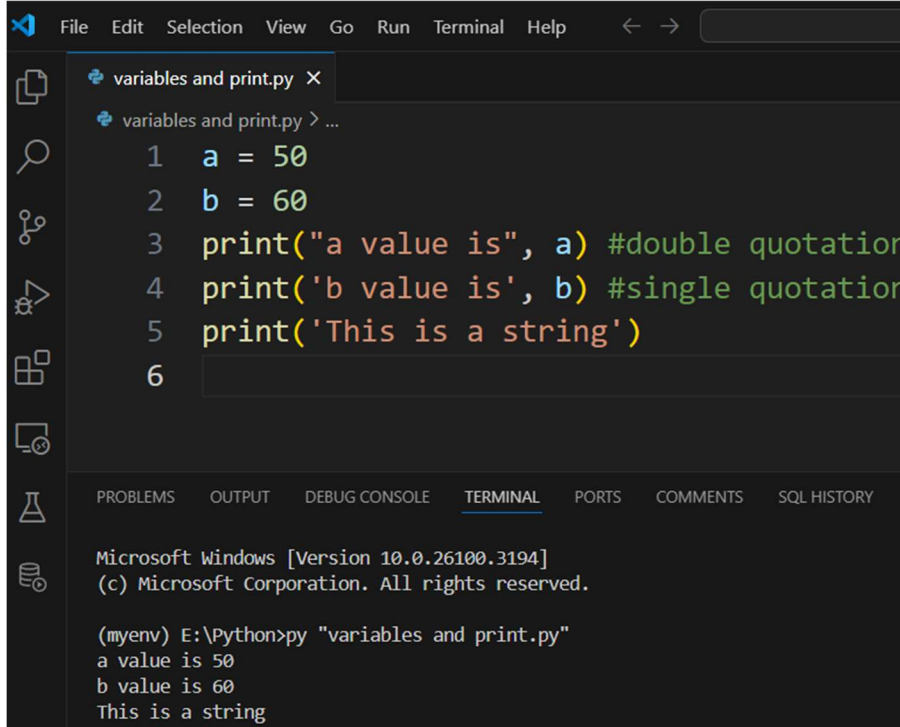
Python Program

পাইথনে বারবার শেষে সেমিকোলন (;) দেওয়া লাগে না।

প্রিন্টের ক্ষেত্রে String প্রিন্ট করার জন্য সিংগেল কোটেশন (‘ ’) বা ডাবল কোটেশন (“ ”) এই দুইটার যেকোনো একটা ব্যবহার করা যায়। এক্ষেত্রে কোনো সমস্যা নাই। প্রত্যেকটা প্রিন্ট ফাংশন

এক্সিকিউট হলে নতুন লাইনে চলে যায়, অর্থাৎ নতুন লাইনের জন্য আলাদা করে \n দেওয়া লাগে না।

উদাহরণঃ



```
File Edit Selection View Go Run Terminal Help
variables and print.py X
variables and print.py > ...
1 a = 50
2 b = 60
3 print("a value is", a) #double quotation
4 print('b value is', b) #single quotation
5 print('This is a string')
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS SQL HISTORY
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

(myenv) E:\Python>py "variables and print.py"
a value is 50
b value is 60
This is a string
```

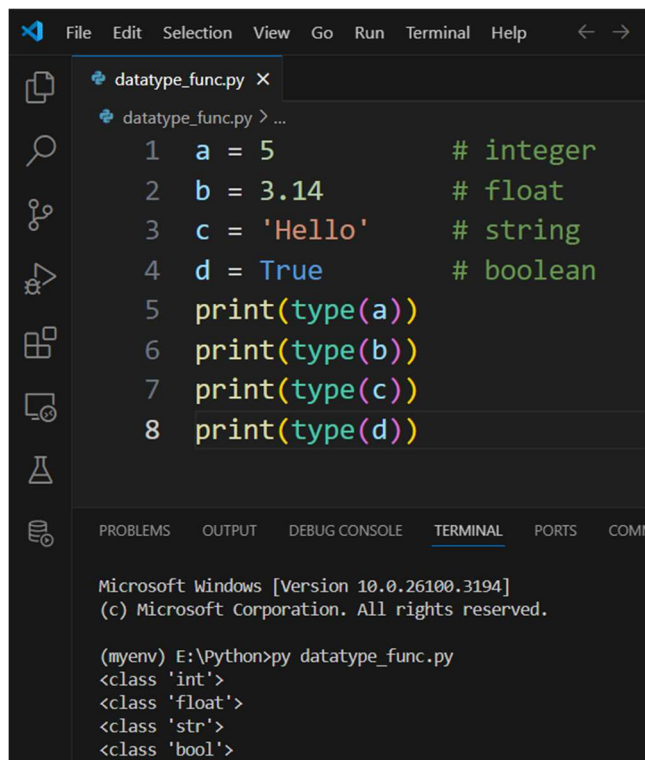
আরো একটা জিনিস আমাদের জানা লাগবে যে, যদি পাইথন ফাইলের নাম স্পেস বহন করে, তাহলে ঐ ফাইলকে Terminal CMD দ্বারা এক্সিকিউট করার সময় ফাইলের নামকে ডাবল কোটেশনের ভিতরে লিখতে হবে। যেমন এখানে পাইথন ফাইলের নাম variables and print.py, এখানে দুইটা স্পেস আছে। এইজন্যে এক্সিকিউশনের সময় Terminal CMD তে এভাবে `py "variables and print.py"` command দিতে হয়েছে যেখানে “ ” দিতে হয়েছে।

পাইথনে কমেন্ট লেখার নিয়ম হলো # দিয়ে কমেন্ট লেখা। কমেন্ট কখনই এক্সিকিউট হয় না। যেমনঃ এখানে “#single quotation” আর “#double quotation” হলো কমেন্ট। এরা

শুধুমাত্র কमेंটস হিসেবে রয়েছে এবং এরা এক্সিকিউটের জন্য কোনো ভূমিকা রাখেনি।

ডেটাইপ এর ফাংশনঃ

কোনো ভেরিয়েবলের ডেটাইপ চেক করার জন্য `type()` ফাংশন ব্যবহার করা হয়।



The screenshot shows a code editor with a file named `datatype_func.py`. The code defines four variables: `a = 5` (integer), `b = 3.14` (float), `c = 'Hello'` (string), and `d = True` (boolean). It then prints the type of each variable using `print(type(a))`, `print(type(b))`, `print(type(c))`, and `print(type(d))`. The terminal output shows the execution of the script, displaying the types: `<class 'int'>`, `<class 'float'>`, `<class 'str'>`, and `<class 'bool'>`.

```
1 a = 5 # integer
2 b = 3.14 # float
3 c = 'Hello' # string
4 d = True # boolean
5 print(type(a))
6 print(type(b))
7 print(type(c))
8 print(type(d))
```

Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

(myenv) E:\Python>py datatype_func.py
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>

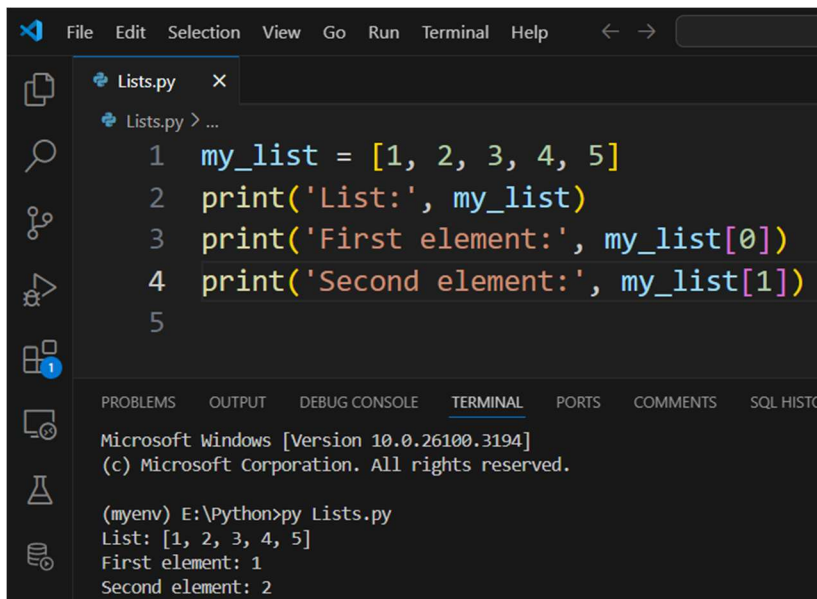
এখানে টাইপ ফাংশন প্রথমে ঐ ভেরিয়েবলের ডেটাইপ নির্ণয় করে। যেমনঃ `type(a)` মূলতঃ `a` ভেরিয়েবলের ডেটাইপ নির্ণয় করেছে, কিন্তু তার একমাত্র কাজ হলো ডেটাইপ কি, সেটাই নির্ণয় করা (সেই নির্ণয় করা ডেটাইপ টা প্রদর্শন করা `type()` এর কাজ নয়)। তাই সেইটা দেখার জন্য আমাদের `print()` ফাংশন ব্যবহার করা লেগেছে। তাই `print(type(a))` দ্বারা আমরা `a` এর ডেটাইপ দেখতে পেরেছি।

লিস্টঃ

লিস্ট (List) হলো পাইথনের array এর আরেক নাম। লিস্ট হলো collection of elements। এই লিস্টে নানা ধরনের ডেটাইপের elements রাখা যায়। লিস্ট ডুপ্লিকেট ভ্যালু সাপোর্ট করে।

লিস্ট তৈরি করতে হয় [] দিয়ে এবং elements গুলো কমা দিয়ে সেপারেট করতে হয়।

উদাহরণঃ

A screenshot of a Python IDE window titled 'Lists.py'. The code in the editor is:

```
1 my_list = [1, 2, 3, 4, 5]
2 print('List:', my_list)
3 print('First element:', my_list[0])
4 print('Second element:', my_list[1])
5
```

The bottom panel shows the 'TERMINAL' output:

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

(myenv) E:\Python>py Lists.py
List: [1, 2, 3, 4, 5]
First element: 1
Second element: 2
```

লক্ষ্য করুন, এখানে নির্দিষ্ট element প্রিন্ট করার জন্য ইন্ডেক্সিং করা হয়েছে, যেমনঃ প্রথম element এর জন্য my_list[0], দ্বিতীয় element এর জন্য my_list[1]। এভাবে তৃতীয় element এর জন্য হবে my_list[2], চতুর্থ element এর জন্য হবে my_list[3]।

১) লিস্টে নতুন element যোগ করাঃ

এক্ষেত্রে, `append()` মিথড ব্যবহার করে নতুন element যোগ করতে হবে। তখন `append()` মিথড এর ভিতর সেই নতুন element থাকতে হবে। লিস্টের নাম এর পরে ডট (.) চিহ্ন দেওয়ার পর `append()` মিথড লিখতে হয়, কারণ `append()` কোনো ফাংশন নয়, বরং এটা একটা মিথড (আপাততঃ এটা মাথায় রাখেন যে ডট চিহ্ন দেওয়ার পর যেসব লিখতে হয়, সেগুলো মিথড)

নিচে উদাহরণস্বরূপ দেখানো হলো।

```
File Edit Selection View Go Run Terminal Help
Lists.py x
Lists.py > ...
1 my_list = [1, 2, 3, 4, 5]
2 print('Old List:', my_list)
3 my_list.append(6)
4 print('New List:', my_list)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMM
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

(myenv) E:\Python>py Lists.py
Old List: [1, 2, 3, 4, 5]
New List: [1, 2, 3, 4, 5, 6]
```

এখানে `my_list` ছিলো `[1, 2, 3, 4, 5]`। `append(6)` করে `my_list` হয়েছে `[1, 2, 3, 4, 5, 6]`

২) লিস্টের elements অ্যাক্সেস করা

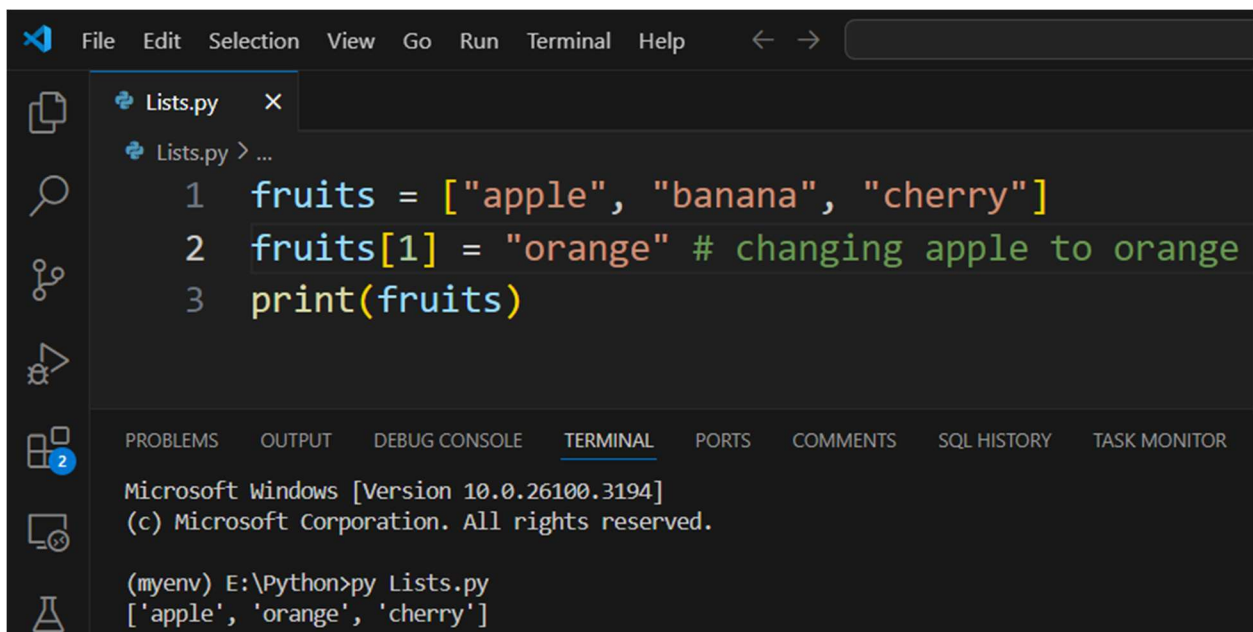
লিস্টের element গুলো তাদের সূচক (index) ব্যবহার করে অ্যাক্সেস করা হয়, যেখানে প্রথম উপাদানের সূচক হয় 0। এটা আগেই দেখানো হয়েছে যেমনঃ- কিছুক্ষণ আগেই বলেছিলাম যে প্রথম element এর জন্য `my_list[0]`, দ্বিতীয় element এর জন্য `my_list[1]`। এভাবে

তৃতীয় element এর জন্য হবে `my_list[2]`, চতুর্থ element এর জন্য হবে `my_list[3]`।
এটাই হলো লিস্টের elements অ্যাক্সেস করার পদ্ধতি।

৩) লিস্টের উপাদান পরিবর্তন করা

লিস্ট হলো mutable, অর্থাৎ এর নির্দিষ্ট উপাদান পরিবর্তন করা যায়।

যেমনঃ এই fruits লিস্টের apple কে orange হিসেবে রিপ্লেস করা।



The screenshot shows a code editor with a file named 'Lists.py'. The code contains three lines: 1. `fruits = ["apple", "banana", "cherry"]`, 2. `fruits[1] = "orange" # changing apple to orange`, and 3. `print(fruits)`. Below the code, the terminal output shows the command `(myenv) E:\Python>py Lists.py` and the resulting list `['apple', 'orange', 'cherry']`.

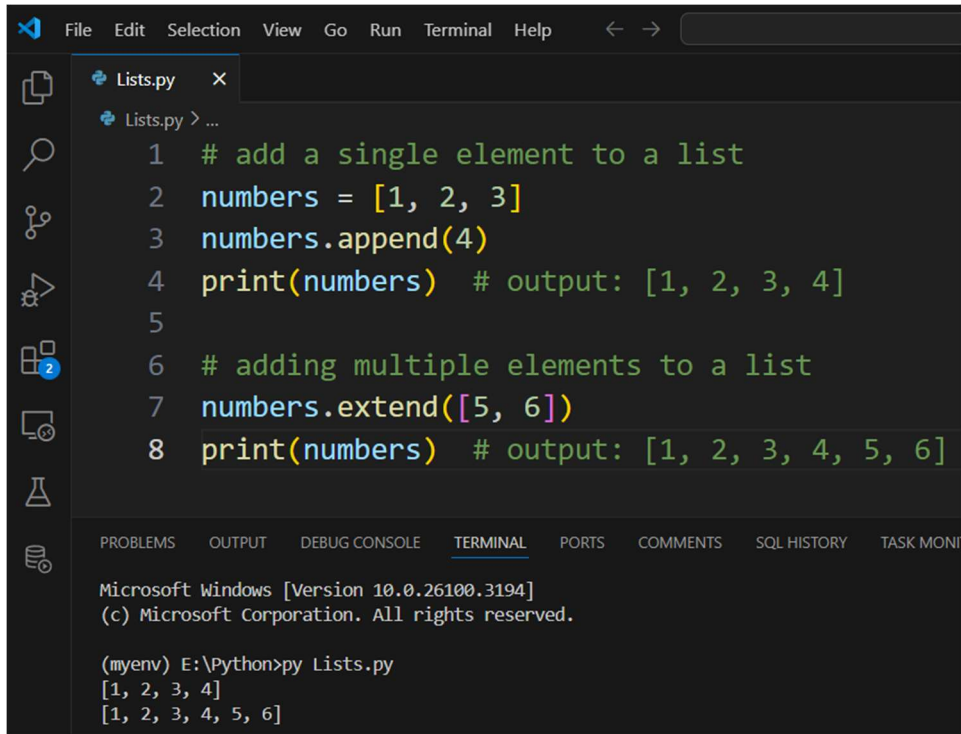
```
File Edit Selection View Go Run Terminal Help
Lists.py x
Lists.py > ...
1 fruits = ["apple", "banana", "cherry"]
2 fruits[1] = "orange" # changing apple to orange
3 print(fruits)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS SQL HISTORY TASK MONITOR
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

(myenv) E:\Python>py Lists.py
['apple', 'orange', 'cherry']
```

৪) লিস্টের উপাদান যোগ করা

`append()` মিথড ব্যবহার করে একটি উপাদান যোগ করা যায় এবং `extend()` ব্যবহার করে একাধিক উপাদান যোগ করা যায়।



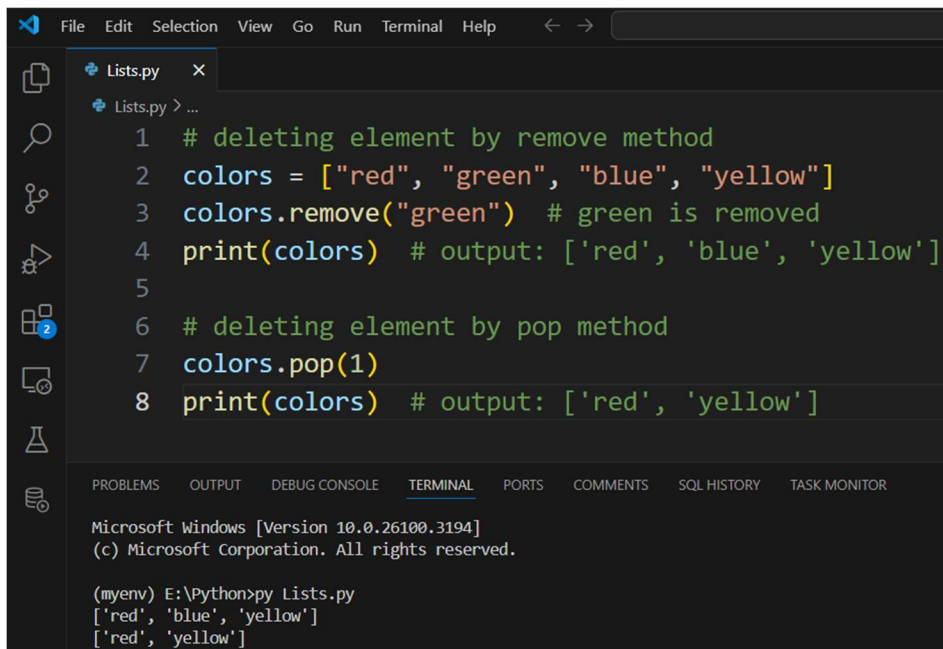
```
File Edit Selection View Go Run Terminal Help
Lists.py x
Lists.py > ...
1 # add a single element to a list
2 numbers = [1, 2, 3]
3 numbers.append(4)
4 print(numbers) # output: [1, 2, 3, 4]
5
6 # adding multiple elements to a list
7 numbers.extend([5, 6])
8 print(numbers) # output: [1, 2, 3, 4, 5, 6]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS SQL HISTORY TASK MONITOR
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

(myenv) E:\Python>py Lists.py
[1, 2, 3, 4]
[1, 2, 3, 4, 5, 6]
```

৫) লিস্ট থেকে উপাদান সরানোঃ

remove(), pop(), বা del ব্যবহার করে লিস্ট থেকে উপাদান সরানো যায়।



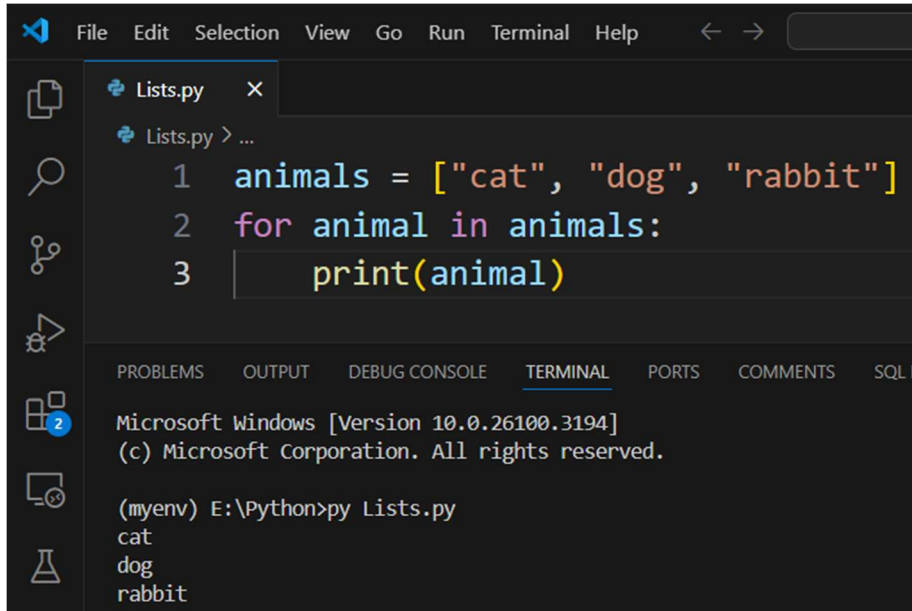
```
File Edit Selection View Go Run Terminal Help
Lists.py x
Lists.py > ...
1 # deleting element by remove method
2 colors = ["red", "green", "blue", "yellow"]
3 colors.remove("green") # green is removed
4 print(colors) # output: ['red', 'blue', 'yellow']
5
6 # deleting element by pop method
7 colors.pop(1)
8 print(colors) # output: ['red', 'yellow']

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS SQL HISTORY TASK MONITOR
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

(myenv) E:\Python>py Lists.py
['red', 'blue', 'yellow']
['red', 'yellow']
```


৬) লিস্টের উপর লুপ চালানোঃ

for লুপ ব্যবহার করে লিস্টের সকল উপাদান একে একে পড়া যায়।



```
File Edit Selection View Go Run Terminal Help
Lists.py x
Lists.py > ...
1 animals = ["cat", "dog", "rabbit"]
2 for animal in animals:
3     print(animal)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS SQL H
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

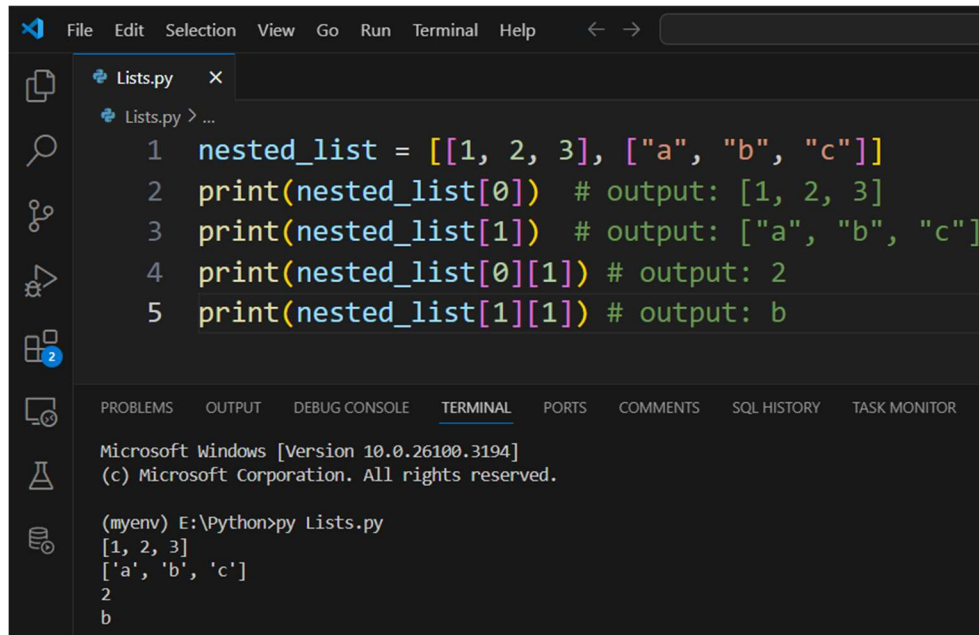
(myenv) E:\Python>py Lists.py
cat
dog
rabbit
```

৭) নেস্টেড লিস্ট (Nested Lists)

লিস্টের ভিতরে আরেকটি লিস্ট রাখা যায়, যা নেস্টেড লিস্ট তৈরি করে।

এক্ষেত্রে `list_name[index1][index2]` এ `index1` ভিতরের লিস্টকে রিপ্রেজেন্ট করে আর `index2` ভিতরের লিস্টের element কে রিপ্রেজেন্ট করে। শুধু `list_name[index1]` প্রিন্ট করলে ভিতরের লিস্ট প্রিন্ট করে। `list_name[index1][index2]` প্রিন্ট করলে ভিতরের লিস্টের element প্রিন্ট করবে। `index1` যদি 0 হয়, তাহলে প্রথম ভিতরের লিস্ট কে বুঝাবে, 1 হলে দ্বিতীয় ভিতরের লিস্ট বুঝাবে, 2 হলে তৃতীয় ভিতরের লিস্ট বুঝাবে। `index1` যদি 0 হয় আর `index2` যদি 0 হয়, তাহলে প্রথম ভিতরের লিস্টের প্রথম element কে বুঝাবে। `index1` যদি 0 হয় আর `index2` যদি 1 হয়, তাহলে প্রথম ভিতরের লিস্টের দ্বিতীয় element কে বুঝাবে।

উদাহরণঃ



```
File Edit Selection View Go Run Terminal Help
Lists.py
Lists.py > ...
1 nested_list = [[1, 2, 3], ["a", "b", "c"]]
2 print(nested_list[0]) # output: [1, 2, 3]
3 print(nested_list[1]) # output: ["a", "b", "c"]
4 print(nested_list[0][1]) # output: 2
5 print(nested_list[1][1]) # output: b

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS SQL HISTORY TASK MONITOR
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

(myenv) E:\Python>py Lists.py
[1, 2, 3]
['a', 'b', 'c']
2
b
```

এখানে `nested_list[0]` দ্বারা `nested_list` এর প্রথম ভিতরের লিস্ট `[1, 2, 3]` কে রিপ্রেজেন্ট করে আর `nested_list[1]` দ্বারা `nested_list` এর দ্বিতীয় ভিতরের লিস্ট `["a", "b", "c"]` কে রিপ্রেজেন্ট করে। `nested_list[0][0]` দ্বারা `nested_list` এর প্রথম ভিতরের লিস্ট `[1, 2, 3]` এর প্রথম element `1` কে রিপ্রেজেন্ট করে। `nested_list[0][1]` দ্বারা `nested_list` এর প্রথম ভিতরের লিস্ট `[1, 2, 3]` এর দ্বিতীয় element `2` কে রিপ্রেজেন্ট করে। একইভাবে `nested_list[1][0]` দ্বারা `nested_list` এর দ্বিতীয় ভিতরের লিস্ট `["a", "b", "c"]` এর প্রথম element `"a"` কে রিপ্রেজেন্ট করে। `nested_list[1][1]` দ্বারা `nested_list` এর দ্বিতীয় ভিতরের লিস্ট `["a", "b", "c"]` এর দ্বিতীয় element `"b"` কে রিপ্রেজেন্ট করে।