বাংলায় পাইথন

howtocode.com.bd

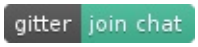
Published
with GitBook

সূচিপত্র

howtocode.com.bd

কোর্স এর মুল পাতা | HowToCode মুল সাইট | সবার জন্য প্রোগ্রামিং ব্লগ | পিডিএফ ডাউনলোড

বাংলায় পাইথন



Unknown error!

প্রারম্ভিকা

পাইথন একটি ডায়নামিক প্রোগ্রামিং ল্যাঙ্গুয়েজ যেটি জয় করেছে বহু ডেভেলপারের হৃদয় । এর মধ্যে আছে গুগল, ড্রপবক্স, ইন্সটাগ্রাম, মোজিলা সহ অনেক বড় বড় প্রতিষ্ঠানের হাজারো প্রকৌশলী । পাইথন এমন একটি ভাষা যার গঠন শৈলী অনন্য এবং প্রকাশভঙ্গি অসাধারণ । চমৎকার এই ল্যাঙ্গুয়েজটি তাই আজ ছড়িয়ে পড়েছে নানা দিকে - ওয়েব, ডেস্কটপ, মোবাইল, সিস্টেম এ্যাডমিনিস্ট্রেশন, সাইন্টিফিক কম্পিউটিং কিংবা মেশিন লার্নিং - সবত্রই পাইথনের দৃপ্ত পদচারণা । বলাই বাহুল্য, আমিও একজন পাইথন ফ্যান এবং “পাইথনিয়ার” । বাংলাদেশের ডেভেলপারদের মধ্যে এই ভাষাটি ছড়িয়ে দিতে আমার এই ক্ষুদ্র প্রয়াস ।

বাংলাদেশী পাইথন ইউজার গ্রুপ

বাংলাদেশী পাইথন ডেভেলপারদের মিলনকেন্দ্র এই ফেইসবুক গ্রুপটি । এটি বাংলাদেশের সবচেয়ে বড় পাইথন ইউজার গ্রুপ । এই গ্রুপের সদস্যরা বাংলাদেশে পাইথন প্রসারে প্রতিনিয়ত অবদান রেখে চলেছেন ।

পাইথন বাংলাদেশের জন্ম হয় ফেইসবুকের বাইরে । মূল ওয়েবসাইটের সাথে ফেইসবুক গ্রুপটির নাম নিয়ে যাতে কনফিউশন তৈরি না হয় তাই ফেইসবুক গ্রুপটির নাম পরবর্তীকালে পরিবর্তন করে রাখা হয় - "পাইচার্মার্স" ।

আমাদের ওয়েব সাইটের ঠিকানা: http://pybd.org

ওপেন সোর্স

এই বইটি মূলত স্বেচ্ছাশ্রমে লেখা এবং বইটি সম্পূর্ন ওপেন সোর্স । এখানে তাই আপনিও অবদান রাখতে পারেন লেখক হিসেবে । আপনার কন্ট্রিবিউশান গৃহীত হলে অবদানকারীদের তালিকায় আপনার নাম যোগ করে দেওয়া হবে ।

এটি মূলত একটি গিটহাব রিপোজিটোরি যেখানে এই বইয়ের আর্টিকেল গুলো মার্কডাউন ফরম্যাটে লেখা হচ্ছে । রিপোজটরিটি ফর্ক করে পুল রিকুয়েস্ট পাঠানোর মাধ্যমে আপনারাও অবদান রাখতে পারেন ।

ইন্সটলেশন

আপনি যদি লিনাক্স বা ম্যাক ব্যবহারকারী হন তবে আপনার পিসি বা ল্যাপটপে পাইথন দেওয়াই আছে । আপনাকে আর বাড়তি কিছু করতে হবে না । টার্মিনালে টাইপ করুন:

```
$ python
```

তাহলেই চলবে । ব্যক্তিগতভাবে আমি সব সময় ডেভেলপমেন্টের জন্য উইন্ডোজ এড়িয়ে চলি । আমি রিকমেন্ড করবো ওস এক্স বা লিনাক্স এ অভ্যস্ত হতে, ভবিষ্যতে কাজে আসবে ।

উইন্ডোজ ব্যবহারকারীরা পাইথনের অফিসিয়াল ওয়েব সাইট থেকে পাইথন 2.x লেটেস্ট ভার্সনটি ডাউনলোড করে ইন্সটল করে নিন । ওয়েব সাইটের ঠিকানা খোঁজা ও ইন্সটলেশন আপনাকে করে নিতে হবে । গুগলের সহায়তা নিতে পারেন ।

পাইথন কোড লেখার জন্যে উইন্ডোজে Notepad++ এবং লিনাক্সে gedit ব্যবহার করতে পারেন । তবে সাবলাইম টেক্সট সব অপারেটিং সিস্টমের জন্যই চমৎকার । এবং তুলনামূলকভাবে ভালো । আইডিই ব্যবহার করতে চাইলে পাইচার্ম কমিউনিটি এডিশন ব্যবহার করতে পারেন । এটি ওদের ওয়েব সাইট থেকে বিনামূল্যে ডাউনলোড করে নিতে পারবেন ।

উইন্ডোজ ব্যবহারকারীরা পাইথন ডিরেক্টরীকে আপনাদের সিস্টেম পাথে যোগ করে নিন । অর্থাৎ C:\Python2x এই লোকেশনটিকে আপনার PATH ভ্যারিয়েবলে যোগ করে নিন । এজন্য:

- Computer এর উপর রাইট ক্লিক করে “Properties” এ যান ।
- বাম পাশে “Advanced System Settings” এ ক্লিক করুন ।
- নিচের দিকে থাকা “Environment Variables” এ ক্লিক করুন ।
- “System Variables” এর ভিতরে “Path” এন্ট্রি খুজে বের করে “Edit” বাটন চাপুন ।
- এবার এর শেষে C:\Python25 লিখে ওকে করে বের হয়ে আসুন ।
- কমান্ড প্রম্পট খুলুন (cmd.exe) । টাইপ করুন ঃ python । এন্টার চাপুন ।

নিচের মত লেখা দেখাবেঃ

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\maSnun>python
Python 2.5.4 (r254:67916, Dec 23 2008, 15:10:54) [MSC v.1310 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

এরকম দেখালে বোঝা গেল আমরা পাইথন ইন্সটলেশন শেষে এটাকে রান করাতে পেরেছি কমান্ড লাইনে । এবার আসুন সত্যিকারের একটি পাইথন প্রোগ্রাম লিখি । টেক্সট এডিটর খুলে টাইপ করুন ঃ

```
greetings = "hello world!"
print greetings
```

ফাইলটিকে ডেস্কটপে সেইভ করুন “test.py” নামে । এবার কমান্ড প্রম্পট বা টার্মিনাল খুলে নিচের কমান্ডদুটো দিন ।

```
cd Desktop
python test.py
```

আউটপুট হবে নিচের মত (উইন্ডোজে)ঃ

```
C:\Users\maSnun>cd Desktop

C:\Users\maSnun\Desktop>python test.py
hello world!

C:\Users\maSnun\Desktop>
```

আমরা সফলভাবে একটি পাইথন প্রোগ্রাম লিখলাম ও রান করলাম । এরপরে আমরা আরো গভীরে যাব ।

কুইক টিপস

এই টিপস গুলো মনে রাখলে পাইথন শিখতে সুবিধা হবে । এগুলো আমাদের দৈনন্দিন জীবনে পাইথন ডেভেলপমেন্টের সময় নানাভাবে সাহায্য করবে ।

ইন্টারএক্টিভ শেল

কমান্ড লাইনে শুধু পাইথন ইন্টারপ্রেটার রান করালে (কোন ফাইল নেইম ছাড়া) পাইথনের ইন্টারএক্টিভ শেল চালু হয় । এখানে কোন এক্সপ্রেশন টাইপ করলে পাইথন সাথে সাথে সেটিকে ইভ্যালুয়েট করে আউটপুট দেখাবে । যেহেতু বার বার ফাইলে সেইভ করে রান করার প্রয়োজন হয় না, তাই দ্রুত কোন কিছু টেস্ট করে দেখা বা প্রোটোটাইপিং এর জন্যে খুবই কাজের জিনিস এটি ।

আমি প্রায়ই এটাকে ক্যালকুলেটর হিসেবে ব্যবহার করি :)

```
$ python
Python 2.7.8 (default, Nov 15 2014, 03:09:43)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.51)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> my_list = [1, 2, 3, 5, 90]
>>> for x in my_list:
...     print x
...
1
2
3
5
90
>>>
```

`type()` , `dir()` , `help()` ফাংশন এর ব্যবহার

পাইথনে প্রোগ্রামিং ও ডিবাগিং এর ক্ষেত্রে এই ফাংশনগুলো অত্যন্ত কাজের । এগুলো পাইথনের গ্লোবাল নেইমস্পেসের অংশ । তাই এগুলো কোন মডিউল ইম্পোর্ট করা ছাড়াই ব্যবহার করা যায় । শুরুতেই তাই এগুলোর ব্যবহার জানলে আমরা আমাদের নানা ধরনের সমস্যার সমাধানে এগুলোকে ব্যবহার করতে পারবো ।

```
>>> type(my_list)
<type 'list'>
>>> type(my_list[0])
<type 'int'>
>>>
```

উপরের অংশ যদি মনযোগ দিয়ে লক্ষ্য করে থাকেন তাহলে দেখবেন `type()` ফাংশনটি কোন চলক বা নামের ধরন বা টাইপ বলে দেয় । যেমনঃ `type(my_list)` দিলে বোঝা গেল এটি একটি লিস্ট । `type(my_list[0])` দিলে দেখা গেল এই লিস্টের প্রথম আইটেমের টাইপ ইন্টিজার । কোন ভ্যারিয়েবল এর টাইপ জানা না থাকলে এটি ব্যবহার করে জেনে নিতে পারি এটি কি টাইপ । এরপর সেই টাইপ অনুযায়ী পরবর্তী অপারেশন চালানো যায় ।

dir() কমান্ডটি কোন অবজেক্টের ইন্সপেকশনে ব্যবহার করা হয় । এই ফাংশনটি ব্যবহার করে আমরা ঐ অবজেক্টের বিভিন্ন প্রোপার্টি এবং মেথডের নাম জানতে পারি ।

```
>>> details = dir(my_list)
>>> type(details)
<type 'list'>
>>> len(details)
45
>>> details
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__delslice__', '_
>>> 'extend' in details
True
>>>
```

`dir()` ফাংশনটি একটি লিস্ট রিটার্ন করে । এই লিস্টটি আমরা প্রোগ্রাম্যাটিক্যালি আমাদের প্রয়োজনে ব্যবহার করতে পারি । বিশেষ করে যখন আমরা জানি না একটি ভ্যারিয়েবলে থাকা অবজেক্টটির বৈশিষ্ট্য কি বা এটি কি করতে পারে । এই লিস্ট দেখে আমরা কিছু ধারনা পাই এই অবজেক্ট এর সাধারন ব্যবহার সম্পর্কে ।

`help()` ফাংশনটি আমাদের কোন অবজেক্ট সমপর্কে সাহায্যকারী তথ্য সরবরাহ করে । মূলত এটি ইন্টার এ্যাকটিভ শেল থেকে সহজেই ডকুমেন্টেশন পড়তে সহায়তা করে থাকে -

```
>>> help(list)
```

এই কমান্ডের মাধ্যমে আমরা লিস্ট টাইপ সম্পর্কে বিস্তারিত জানতে পারবো ।

কমান্ড লাইনে `-i` আর্গুমেন্ট এর ব্যবহার

```
$ python -i my_file.py
```

পাইথন কমান্ড লাইনে `-i` আর্গুমেন্ট এর ভ্যালু হিসেবে কোন পাইথন ফাইল নেইম পাস করলে পাইথন প্রথমে ঐ ফাইল টি রান করে এবং ফলাফল সহ ইন্টারএ্যাক্টিভ শেল চালু করে দেয় । ফলে আপনার ঐ ফাইলে আপনি যে সব অপারেশন চালাবেন, সেগুলো নিয়ে এই ইন্টারএ্যাক্টিভ শেলে কাজ করতে পারবেন । যেমন: পাইথন ফাইলটির মধ্যে ডিফাইন করা ভ্যারিয়েবলগুলো আপনি এই শেলে পাবেন । ডিবাগিং এবং দ্রুত প্রটোটাইপিং এটি বেশ কাজে দেয় ।

কমেন্টস

যে কোন প্রোগ্রামিং ল্যাঙ্গুয়েজেই কমেন্ট অত্যন্ত গুরুত্বপূর্ন বিষয় । কমেন্ট হল কোডের সেই অংশ বিশেষ যা ইন্টারপ্রেটার এক্সিকিউট করবে না । কমেন্ট লেখা হয় মূলত যারা পরবর্তীতে এই কোড পড়বেন তাদের জন্য । প্রোগ্রামের বিভিন্ন বিষয় সোর্স কোডের মধ্যেই ব্যখ্যা করা হয় কমেন্টের মাধ্যমে ।

পাইথনে আমরা পাউন্ড বা হ্যাশ ক্যারেক্টার ব্যবহার করে কমেন্ট লিখে থাকি । যেমন:

```
# this is a comment
print "this is not"

# my cool add function
# adds a and b
# then returns the value
def add(a,b):
    return a+b # everything after the pound sign is  a comment
```

দেখা যাচ্ছে – কমেন্ট মাল্টিপল লাইনে হতে পারে, শুধু লাইনের শুরুতে পাউন্ড চিহ্ন বসালেই হল । একই লাইনে কিছু কোড এর পরে পাউন্ড সাইন ব্যবহার করে কমেন্ট লেখা যায় । তবে খেয়াল রাখা দরকার, একবার পাউন্ড সাইন দিয়ে কমেন্ট লেখা শুরু করলে তারপর থেকে ঐ লাইনের বাকিটা কমেন্ট হিসেবে বিবেচিত হবে । পাইথনে কমেন্ট শেষ করার ব্যবস্থা নেই, তাই সি বা জাভার মত কমেন্ট ব্লকও (/…./) সম্ভব না ।

ইন্ডেন্টেশন

অন্যান্য প্রোগ্রামিং ল্যাঙ্গুয়েজ থেকে পাইথনে আসলে প্রথম যে সমস্যাটি চোখে পড়ে তাহল পাইথনের ইন্ডেন্টেশন বেইজড কোড ব্লক । পাইথনের একই ইন্ডেন্টেশন সম্বলিত পর পর অবস্থিত লাইন গুলো একই কোড ব্লকের অন্তর্ভুক্ত । উদাহরণ না দিলে হয়ত বিষয়টি স্পষ্ট হবে না ।

```
if True:
....print "hello world"
....print "Hi there"
....print "4 space indentation"
```

এখানে আমরা ডট (.) দিয়ে স্পেইস বুঝিয়েছি । নিজে টাইপ করার সময় ডট এর পরিবর্তে স্পেইস ব্যবহার করুন না হলে প্রোগ্রাম রান করবে না । এখানে দেখুন ৩টি প্রিন্ট স্টেটমেন্ট আছে যারা একটির পর আরেকটি অবস্থিত এবং প্রত্যেকটি ৪টি স্পেইস দিয়ে ইন্ডেন্ট করা । এর ফলে এই প্রিন্ট স্টেটমেন্ট গুলো একটি কোড ব্লক হিসেবে কাজ করে । কোন স্টেটমেন্ট এ যদি একই ইন্ডেন্টেশন না থাকত সেক্ষেত্রে পাইথন এক্সেপশন (এরর) থ্রো করত ।

```
if True:
....print "hello world"
....print "Hi there"
....print "4 space indentation"
else:
....print "another block"
....print "same indentation as of the if block"
....print "but terminated with the else condition"
```

এখানে দেখুন প্রথম ব্লকের পর ইন্ডেন্টেশন আগের জায়গায় ফিরিয়ে নিয়ে ব্লকের সমাপ্তি টানা হয়েছে । পরে আবার ৪ স্পেইস ইন্ডেন্ট করে আরেকটি ব্লকের সূচনা করা হয়েছে । আগের ব্লক এবং এই ব্লক দুটোরই একই ইন্ডেন্টেশন কিন্তু এরা একই ব্লক নয় । কারণ এদের মাঝে else আছে ।

এবার দেখি নেস্টেড ব্লক:

```
if True:
....print "hello world"
....print "Hi there"
....print "4 space indentation"
....if True:
........print "nested block"
........print "8 spaces"
....print "back to prev block"
```

এখানে আমরা স্পেইস এর পরিমান বাড়িয়ে দিয়ে নেস্টেড ব্লক তৈরি করলাম । একইভাবে স্পেইস এর পরিমান সমান পরিমানে কমিয়ে নিয়ে আগের ব্লকে ফিরে গেলাম ।

পাইথনের হোয়াইটস্পেস বেইজড ইন্ডেন্টেশন বুঝতে প্রথম প্রথম বেশ কষ্ট হয় । এজন্য দরকার অনুশীলন । নিজে নিজে ইন্ডেন্ট করার চেষ্টা করুন । গুগলে অনুসন্ধান করে দেখতে পারেন পাইথনের ইন্ডেন্টেশন নিয়ে ।

ভ্যারিয়েবলস এ্যান্ড ডাটা টাইপস

নেইমস

প্রোগ্রামিং করতে গিয়ে প্রায়শই আমাদের কিছু ডাটা সংরক্ষণের প্রয়োজন হয় । এই ডাটা পাইথন কম্পিউটারের মেমরিতে সংরক্ষণ করতে পারে । ভ্যারিয়েবল হলো মেমরির একটা ব্লক যেখানে আমরা প্রয়োজনমত ডাটা সংরক্ষন করি । এরপর ভ্যারিয়েবলটাকে আমরা একটা নাম দিয়ে দেই যাতে ঐ ব্লকটাকে খুব সহজে আমরা ব্যবহার করতে পারি ।

```
name = "Abu Ashraf Masnun"
print name
```

এখানে দেখুন, আমরা প্রথমে একটা ভ্যারিয়েবল এ আমার নামটা সংরক্ষণ করলাম । এরপর ঐ ভ্যারিয়েবলটাকে নাম দিলাম `name` । পরে যখন ঐ ভ্যারিয়েবলটা আউটপুট করার দরকার পড়লো তখন তাকে নাম ধরেই প্রিন্ট স্টেটমেন্ট এ ব্যবহার করলাম । এভাবেই মূলত আমরা ভ্যারিয়েবল এ নানা ধরনের ডাটা সংরক্ষণ করতে পারি ।

টাইপস

আমাদের উপরের উদাহরণে আমরা দেখলাম আমার নামটি কিভাবে সংরক্ষণ করলাম । নাম ছাড়াও আমাদের নানা ধরনের ডাটা সংরক্ষণ করার প্রয়োজন হয় - পূর্ণ সংখ্যা, দশমিক সহ পূর্ণ সংখ্যা, কোন কিছুর তালিকা ইত্যাদি । বিভিন্ন ধরণের ডাটা সংরক্ষণের জন্য তাই পাইথনের ভ্যারিয়েবলগুলো বিভিন্ন টাইপের হতে পারে ।

টাইপ ইন্সপেকশন

যে কোন ভ্যারিয়েবল কোন টাইপের ডাটা সংরক্ষণ করছে তা জানার জন্য আমরা বিল্ট ইন `type()` ফাংশনটি ব্যবহার করতে পারি । এই ফাংশনটিকে কোন ভ্যারিয়েবল পাস করে দিলে এটি আমাদের জানিয়ে দিবে তার ডাটা টাইপ ।

পাইথন ইন্টারএ্যাক্টিভ শেল থেকে ঝটপট কিছু উদাহরন দেখে নেই:

```
>>> s = "string"
>>> type(s)
<type 'str'>
>>> integer = 45
>>> type(integer)
<type 'int'>
>>> float_val = 23.5
>>> type(float_val)
<type 'float'>
>>>
```

ডাইনামিক টাইপিং

পাইথনে ভ্যারিয়েবলগুলো সহজেই তাদের ডাটার টাইপ পরিবর্তন করতে পারে । ধরুন একটি ভ্যারিয়েবল প্রথমে স্ট্রিং টাইপের ডাটা রাখতো, আমরা চাইলেই সেই ভ্যারিয়েবলটিতেই ইন্টিজার ভ্যালু এ্যাসাইন করতে পারি। এই যে ভ্যারিয়েবল এর টাইপ ইচ্ছামত পরিবর্তন করার সুযোগ - এটাকে বলা হয় ডাইনামিক টাইপিং । আর যদি একটা ভ্যারিয়েবল এর টাইপ পরবর্তীতে পরিবর্তন না করা যায়, তখন সেটাকে বলা হয় স্ট্যাটিক টাইপিং ।

পাইথনে ডাইনামিক টাইপিং সিস্টেম বিদ্যমান । তাই আমরা এরকম করতে পারি:

```
myvar = "hi"
print myvar
myvar = 23
print myvar
```

স্ট্রং টাইপিং

পাইথনে কোন একটি টাইপের ভ্যালুকে অন্য টাইপে কনভার্ট করতে গেলে সেটা এক্সপ্লিসিটলি করতে হয় । যেমন: ধরুন আপনার বয়স সংরক্ষণ করা আছে `age` ভ্যারিয়েবলে । নাম আছে `name` এ ।

```
age = 23
name = "masnun"
```

আপনি চাইছেন দুটোকে জোড়া দিয়ে একটি নতুন স্ট্রিং বানাতে । এখন `age` যেহেতু ইন্টিজার, সেহেতু এটাকে প্রথমে স্ট্রিং এ কনভার্ট করে নিতে হবে, তবেই কিনা আপনি এ দুটোকে জোড়া লাগাতে পারবেন ।

```
print "I am " + str(age) +" years old"
```

কোন কোন ভাষায় এই টাইপ কনভার্শনটা অটোমেটিক্যালি করে নেয়, সেক্ষেত্রে সেটাকে উইক টাইপিং বলা হয় । পাইথন যেহেতু নিজে থেকে করে না, এটাকে তাই বলা হয় স্ট্রং টাইপিং ।

কমন টাইপস

এখানে আমরা কিছু কমন ডাটা টাইপ দেখবো:

বুলিয়ান

হ্যা কিংবা না, সত্য কিংবা মিথ্যা - কোন ভ্যারিয়েবল যখন ঠিক বিপরীতধর্মী দুইটা ভ্যালুর যে কোন একটা গ্রহন করে তখন আমরা সেটাকে সচারচর বুলিয়ান টাইপ দিয়ে প্রকাশ করি । পাইথনে একটা বুলিয়ান ভ্যারিয়েবল এর ভ্যালু হতে পারে হয় `True` অথবা `False` ।

```
male = True
old = False
```

পাইথনের বিল্টইন `bool()` ফাংশনটি ব্যবহার করে আমরা যে কোন টাইপের ভ্যারিয়েবল কে বুলিয়ানে কনভার্ট করতে পারি । এখানে কিছু উদাহরণ দেওয়া হলো:

```
print bool(True)
print bool(False)
print bool("text")
print bool("")
print bool(' ')
print bool(0)
print bool()
print bool(3)
print bool(None)
```

উপরের কোডটি রান করে আউটপুট দেখে বোঝার চেষ্টা করুন কি ভ্যালুর জন্য বুলিয়ান কি ভ্যালু পাওয়া যাবে ।

নান

নান বা `None` হচ্ছে বিশেষ ধরনের ডাটা টাইপ যেটা নির্দেশ করে এই ভ্যারিয়েবলটির কোন ভ্যালু নেই ।

```
>>> n = None
>>> type(n)
<type 'NoneType'>
>>>
```

নাম্বার্স

বোঝাই যাচ্ছে এই টাইপের কাজ । বিভিন্ন ধরণের সংখ্যা ধারণ করার জন্য আমরা বিভিন্ন ধরণের টাইপ ব্যবহার করে থাকি ।

পূর্ণ সংখ্যার জন্য আমরা `int` বা ইন্টিজার ব্যবহার করে থাকি ।

```
>>> type(age)
<type 'int'>
```

পাইথনের বিল্ট ইন `int()` ফাংশনটি ব্যবহার করে আমরা অন্য টাইপ থেকে ইন্টিজারে কনভার্ট করতে পারি ।

দশমিক ঘর পর্যন্ত মান সংরক্ষণ করতে আমরা `float` টাইপ ব্যবহার করি । যেমন:

```
>>> price = 23.45
>>> type(price)
<type 'float'>
>>>
```

এটির জন্যও `float()` ফাংশনটি বিদ্যমান যেটির মাধ্যমে আমরা অন্য টাইপ থেকে ফ্লোট ভ্যালু পেতে পারি ।

স্ট্রিংস

টেক্সট ডাটা সংরক্ষণ করার জন্য আমরা স্ট্রিং ব্যবহার করে থাকি ।

```
my_string = "I am a Python developer!"
```

স্ট্রিং টাইপের জন্য বিল্ট ইন ফাংশন হলো - `str()`

লিস্ট

সাধারণত কোন তালিকা সংরক্ষনের জন্য লিস্ট ব্যবহার করা হয় । (থার্ড) ব্রাকেটের ভিতরে কমা দিয়ে একেকটি আইটেম সেপারেট করে দিলেই লিস্ট তৈরি হয়ে যাবে । আসুন উদাহরণ দেখি:

```
my_list = [1,"a string",45.56]
print my_list[0]
print my_list[1]
print my_list[2]
```

লক্ষ্য করুন লিস্ট এ আমরা বিভিন্ন ধরনের ডাটা টাইপ সংরক্ষন করতে পারি একটা তালিকাবদ্ধ উপায়ে । আমরা চাইলেই এই তালিকার আইটেমগুলো ইচ্ছেমত পরিবর্তন করতে পারি । লিস্ট তৈরি করার জন্য বিল্ট ইন ফাংশন `list()` ।

টাপল

টাপল ও লিস্ট এর মতই । শুধু মূল পার্থক্য টাপল এর আইটেমগুলো আমরা চাইলেই পরিবর্তন করতে পারি না ।

টাপল ডিফাইন করা হয় `()` ব্যবহার করে -

```
tpl = (1, 2, 3)
```

অন্য টাইপগুলোর মতই `tuple()` ফাংশনটি কল করে আমরা টাপল পেতে পারি ।

সেট

সেট হচ্ছে আনঅর্ডার্ড লিস্ট - অর্থাৎ আইটেমগুলো কোন নির্দিষ্ট অর্থবহ অর্ডারে থাকে না । এবং সেট এর আইটেমগুলো ইউনিক হয় । অর্থাৎ একটি সেটে একই আইটেম একাধিকবার থাকতে পারে না । উদাহরণ:

```
>>> set1 = set(['a', 'b', 'c', 'c', 'd'])
>>> print set1
set(['a', 'c', 'b', 'd'])
>>>
```

এখানে দেখুন `set()` ফাংশনে পাস করা লিস্টটিতে অনেক ডুপ্লিকেট ভ্যালু ছিলো । এটি থেকে আমরা যে সেটটি পেলাম সেটিতে ডুপ্লিকেট কোন ভ্যালু নেই! `set()` ফাংশনটি কল করে নতুন সেট তৈরি করা যায় ।

ডিকশনারি

ডিকশনারি হলো এমন একটি ডাটা টাইপ যেখানে আমরা একটি কি (key) এর সাথে মিল রেখে একটি ভ্যালু সংরক্ষণ করি । সহজে বলা যায়, ডিকশনারি হলো কতগুলো কি-ভ্যালু (key-value) জোড় এর সমষ্টি ।

```
>>> me = {"name": "masnun", "email": "masnun [at] transcendio.net"}
>>> me
{'name': 'masnun', 'email': 'masnun [at] transcendio.net'}
```

এখানে `me` একটি ডিকশনারি যার `name` কি এর ভ্যালু হলো `masnun` ।

অন্যান্য

পাইথনে টাইপ সিস্টেম অত্যন্ত গুরুত্বপূর্ণ ভূমিকা পালন করে । পাইথনে অসংখ্যা বিল্ট ইন ও ইউজার ডিফাইনড টাইপ আছে । আলোচনার খাতিরে অধিক জনপ্রিয় টাইপগুলোর মধ্যেই আপাতত সীমাবদ্ধ থাকছি আমরা ।

স্ট্রিং অপারেশনস

সিঙ্গল কোট, ডাবল কোট ও এস্কেইপ ক্যারেক্টার

পাইথনে সিঙ্গল বা ডাবল কোটেশন দুটোর মাধ্যমেই স্ট্রিং ডিফাইন করা যায় । তবে যেটি দিয়ে স্ট্রিং শুরু করবেন, শেষও করতে হবে সেটি দিয়েই । এক ধরণের কোটেশনের মধ্যে অন্য কোটেশন সরাসরি প্রিন্ট হয়ে যাবে । যে কোটেশন দিয়ে স্ট্রিং ব্যবহার করা হচ্ছে তার ভিতরে যদি ঐ কোটেশন চিহ্নটি কোন কারণে ব্যবহার করতে হয় তবে তার আগে একটি ব্যাক স্ল্যাশ ব্যবহার করতে হয় । আমরা কিছু উদাহরণ দেখি:

```
print "double quotation"
print 'here is the single'
print "single quote - ' in a double quoted string"
print 'double quotes - " in a single quoted string'

print "here comes the escaped quotes - \" "
print 'here is the single one - \' '
```

একটি ফাইলে এই কোড টাইপ করে রান করে দেখুন কি আউটপুট দেখায় । এখানে ব্যাকস্ল্যাশ ব্যবহার করে আমরা কোট টাকে এড়িয়ে যেতে পারি তাই এটাকে (`\`) পাইথনে এস্কেইপ ক্যারেক্টার বলা হয় ।

সিঙ্গল কোট বা ডাবল কোট ব্যবহার করে মাল্টিলাইন স্ট্রিং করা একটু জটিল। কারন তখন আমাদের নিউলাইন এস্কেইপ করতে হয় । তাই মাল্টিলাইন স্ট্রিং এর জন্য সিঙ্গল কোট বা ডাবল কোট ব্যবহার করতে চাইলে আমরা `+` ব্যবহার করে একাধিক লাইন যোগ করে নিতে পারি অথবা নিচের সিনট্যাক্স ব্যবহার করতে পারি ।

মল্টিলাইন স্ট্রিং

এই বিশেষ সিনট্যাক্সটি ব্যবহার করে আমরা পাইথনে সহজেই মাল্টিলাইন স্ট্রিং তৈরি করতে পারি ।

```
rochona = """The cow is a domestic animal. It has four legs and a long tail and
We have a cow."""

print rochona
```

প্রিন্টিং

পাইথনে আমরা কোন আউটপুট দেখাতে চাইলে প্রিন্ট স্টেটমেন্টটি ব্যবহার করে থাকি । উদাহরন:

```
print "Hello world!"
```

লিস্ট হিসেবে ক্যারেক্টার এ্যাক্সেস

পাইথনে স্ট্রিং হলো ক্যারেক্টারের লিস্ট । তাই আমরা লিস্ট এ্যাক্সেস সিনট্যাক্স ব্যবহার করে যে কোন পজিশনে থাকা ক্যারেক্টার এ্যাক্সেস করতে পারি ।

```
my_string = "Hello Python"
print my_string[2]
```

স্ট্রিং লেন্থ

যেহেতু স্ট্রিংও এক ধরনের বিশেষ লিস্ট, তাই এর লেন্থও একই ভাবে বের করা যায় । আমরা যদি my_string স্ট্রিংটির লেন্থ বের করতে চাই তাহলে নিচের মত করে `len()` ফাংশনটি ব্যবহার করবো:

```
print len(my_string)
```

আপার কেইস ও লোয়ার কেইস

টপিকের নাম শুনেই বুঝতে পারছেন এই সেকশনে আমরা কি নিয়ে আলোচনা করবো ।

```
big = "AAA"
print big.lower()

small = "aaa"
print small.upper()
```

সব পাইথন স্ট্রিং অবজেক্ট এর - `upper()` এবং `lower()` নামে দুটি মেথড থাকে । এদের কল করে আমরা ঐ স্ট্রিংকে আপার কেইস বা লোয়ার কেইসে কনভার্ট করতে পারি ।

ফরম্যাটেড স্ট্রিং

ধরুন আমাদের ৩টি ভ্যারিয়েবল আছে -

```
name = "masnun"
age = 24
email = "masnun@transcendio.net"
```

এখন যদি আমরা এই তিনটি ভ্যারিয়েবল দিয়ে একটি অর্থবহ বাক্য প্রিন্ট করতে চাই, তাহলে আমাদের এরকম কিছু করতে হবে:

```
print "My name is " + name + ". I am " + str(age) + " years old. You can reach me via ema
```

দেখুন, এখানে আমরা অনেকগুলো ভ্যারিয়েবল যোগ করেছি একটা একটা করে । ভ্যারিয়েবল এর সংখ্যা বাড়লে কোড এর অবস্থা আরো অগোছালো হয়ে যাবে । তাছাড়া দেখুন `age` এর টাইপ স্ট্রিং না হওয়ায় এটাকে জোর করে স্ট্রিং করার প্রয়োজন হয়েছে। এই ঝামেলাপূর্ন ফরম্যাটিং এর কাজটাই আমরা খুব সহজে করতে পারি একটু অন্যভাবে । এক্ষেত্রে আমরা ঐ স্ট্রিংটিতে কিছু প্লেসহোল্ডার রাখবো, এবং পরে এই প্লেস হোল্ডার গুলোর স্থানে ভ্যারিয়েবলগুলোর ভ্যালু বসিয়ে প্রিন্ট করবো । উদাহরন দেখি:

```
print "My name is %s, I am %d years old and my email address is %s " % (name, age, email)
```

এখানে `%s` দিয়ে বোঝানো হয় যে এই জায়গাটায় একটা স্ট্রিং ভ্যালু বসবে, `%d` এর স্থানে ইন্টিজার । স্ট্রিং টি সম্পূর্ণ হওয়ার পর `%` এর পর আমাদের ভ্যারিয়েবল গুলো একটি টাপল এর মধ্যে পাস করে দিলেই হলো ।

স্পেশাল ক্যারেক্টার এবং এস্কেইপ সিকুয়েন্স

স্ট্রিং এর ভিতরে আমরা বিশেষ কিছু ক্যারেক্টার ব্যবহার করতে পারি যেগুলো একটু ভিন্ন ধরনের । যেমন আমরা যদি নতুন লাইন তৈরি করতে চাই সেক্ষেত্রে আমরা নিউলাইন ক্যারেক্টার টি ব্যবহার করতে পারি -

```
print "Hello\nWorld!"
```

এখানে `\n` টি নিউলাইন ক্যারেক্টার । এমনি ভাবে `\t` দিয়ে নিউ ট্যাব ব্যবহার করা যায় । বিভিন্ন অপারেটিং সিস্টেমে এধরনের কিছু স্পেশাল ক্যারেক্টার থাকে যা আমরা সরাসরি পাইথন স্ট্রিং এ ব্যবহার করতে পারি ।

এখানে লক্ষ্য করুন দুটি স্পেশাল ক্যারেক্টারের আগেই `\` বিদ্যমান । আমরা আগেই দেখেছি, এটি হচ্ছে পাইথনে এস্কেইপ ক্যারেক্টার । এস্কেইপ ক্যারেক্টারের পর এক বা একাধিক ক্যারেক্টার বসিয়ে আমরা যে এস্কেইপ সিকুয়েন্স পাই । কিছু প্রচলিত এস্কেইপ সিকুয়েন্স নিচে দেওয়া হলো -

সিকুয়েন্স	পরিচিতি
`\\`	একটা ব্যাকস্ল্যাশ
`\'`	সিঙ্গল কোট (')
`\"`	ডাবল কোট (")
`\a`	বেল
`\b`	ব্যাকস্পেইস
`\f`	ফর্মফিড
`\n`	লাইন ব্রেক
`\N{name}`	ইউনিকোড ক্যারেক্টার এর নাম
`\r ASCII`	ক্যারিজ রিটার্ন (ম্যাক ওস এক্স এ নিউ লাইন ক্যারেক্টার)
`\t`	ট্যাব
`\uxxxx`	১৬ বিট হেক্সাডেসিম্যাল ভ্যালু সম্বলিত ইউনিকোড ক্যারেক্টার
`\Uxxxxxxxx`	৩২ বিট হেক্সাডেসিম্যাল ভ্যালু বিশিষ্ট ইউনিকোড ক্যারেক্টার
`\v`	ভার্টিক্যাল ট্যাব
`\ooo`	`ooo` অক্টাল ভ্যালু বিশিষ্ট ক্যারেক্টার
`\xhh`	`hh` হেক্সাডেসিম্যাল ভ্যালুওয়ালা ক্যারেক্টার

(এই টেবিল টি জেড শ এর লার্ন পাইথন দ্যা হার্ড ওয়ে বইটি থেকে অনুবাদকৃত)

লিস্টস ইন ডেপথ

লিস্ট শব্দের বাংলা অর্থ তালিকা। আমাদের বোধহয় ব্যখ্যা করার দরকার পড়ে না তালিকা কি জিনিস । পাইথনেও লিস্ট একই কাজ করে । সহজ কথায় লিস্ট হল কতগুলো আইটেমের একটি তালিকা । অনেক প্রোগ্রামিং ল্যাঙ্গুয়েজে লিস্ট ডিক্লেয়ার করার সময় বলে দিতে হয় লিস্টের আইটেমগুলোর টাইপ কি হবে, পাইথনে তার দরকার পড়ে না । একটি লিস্টের আইটেমগুলো বিভিন্ন টাইপের হতে পারে ।

কিভাবে লিস্ট ডিক্লেয়ার করব? (থার্ড) ব্রাকেটের ভিতরে কমা দিয়ে একেকটি আইটেম সেপারেট করে দিলেই লিস্ট তৈরি হয়ে যাবে । আসুন উদাহরণ দেখি:

```
my_list = [1,"a string",45.56]
print my_list[0]
print my_list[1]
print my_list[2]

print my_list

print type(my_list[0])
print type(my_list[1])
print type(my_list[2])
```

প্রথমে কোডগুলো মনোযোগ দিয়ে পড়ুন । বোঝার চেষ্টা করুন এর আউটপুট কি হতে পারে ।

বরাবরের মত একটি পাইথন ফাইলে এই কোডগুলো লিখে রান করে দেখুন কি আউটপুট দেখায় । `type()` ফাংশনটির ব্যবহার আমরা আগেই দেখেছি । আউটপুট দেখে মিলিয়ে নিন আপনি কি আশা করেছিলেন আউটপুট হিসেবে আর কি এসেছে আউটপুট । যদি না মিলে, বোঝার চেষ্টা করুন কোথায় বুঝতে পারেন নি ।

এই কোড থেকে আমরা কি কি দেখলাম:

- কিভাবে লিস্ট ডিক্লেয়ার করতে হয়
- লিস্টের আইটেমগুলোর একটি ইন্ডেক্স ভ্যালু থাকে।
- এই ইন্ডেক্স ভ্যালু ব্যবহার করে আমরা n-তম আইটেমের মান বের করতে পারি ।
- এই ভ্যালুর মান 0 থেকে শুরু হয় । অর্থাৎ প্রথম আইটেমের ইন্ডেক্স 0, দ্বিতীয়টির 1 এভাবে n-তম আইটেমের ইন্ডেক্স (n-1) ।

লিস্ট সম্পর্কে আরো জানার আগে আমরা `range()` ফাংশনটির ব্যবহার দেখে নেই । এই ফাংশনটির একটি উদাহরণ দেখি :

```
print range(0,10)
print range(0, 100, 10)
```

এই ফাংশনটি সংখ্যার লিস্ট তৈরি করে । এর সিগনেচার অনেকটা এরকম: `range(min,max,step)` । এখানে `min` হল নূন্যতম ভ্যালু যেটা থেকে লিস্ট শুরু হবে । `max` হল সর্বোচ্চ ভ্যালু যেটার ঠিক আগের ভ্যালু পর্যন্ত লিস্ট তৈরি হবে । `step` হল মধ্যবর্তী ব্যবধান ।

উপরোক্ত কোড রান করালে প্রথমে আমরা পাব 0 থেকে শুরু করে 10 এর ঠিক আগের ভ্যালু অর্থাৎ 9 পর্যন্ত । যদি step না দেওয়া হয় তাহলে পাইথন এটার ভ্যালু 1 ধরে নেয় । দ্বিতীয় বার আমরা step হিসেবে 10 দিয়েছি । তাই এবার আমরা 0 থেকে শুরু করে প্রতি 10 ঘর পর পর সংখ্যার লিস্ট পাব 90 পর্যন্ত ।

আমরা লিস্ট প্র্যাকটিস করার জন্য range() ফাংশনটি ব্যবহার করে দ্রুত লিস্ট তৈরি করে নিব । আসুন ফেরা যাক লিস্টে । আমরা দেখেছি কিভাবে ইন্ডেক্স ব্যবহার করে আমরা লিস্টের আইটেমগুলো এক্সেস করেছি । ধরুন আমাদের লিস্টের সব ডাটা লাগবে না, আমরা একটি নির্দিষ্ট রেঞ্জ নিয়ে কাজ করতে চাই । পাইথন আমাদের সেই সুবিধা দেয় (যা অন্য অনেক ল্যাঙ্গুয়েজে পাওয়া যায় না) । আসুন দেখি কিভাবে:

এই উদাহরণটি নিজেরা চেষ্টা করার জন্য প্রথমেই একটি লিস্ট তৈরি করে নেই ।

```
sl = range(1,11) # A list containing the integers from 1 to 10
```

আসুন এবার লিস্ট নিয়ে নাড়া চাড়া করা যাক:

```
list1to5 = sl[0:5]
print list1to5

list2to7 = sl[1:7]
print list2to7
```

এই কোড রান করালে দেখা যাবে `list1to5` একটি লিস্ট যার ভ্যালু 1 থেকে 5 । `sl[0:5]` বলতে বোঝানো হয় `sl` নামক লিস্টের 0-তম আইটেম থেকে শুরু করে 5-তম আইটেমের আগের আইটেম পর্যন্ত আইটেমগুলো নিয়ে তৈরি একটি লিস্ট । এবার নিজে নিজেই বোঝার চেষ্টা করুন `list2to7` এর ভ্যালু কি হতে পারে এবং কেন ।

এবার নিজে কিছু কাজ করুন:

- 3 থেকে 9 পর্যন্ত লিস্ট বোঝাতে আমরা কি লিখব?
- `sl[:5]` এর ভ্যালু কত হবে?
- `sl[4:]` এর ভ্যালু কত হবে?
- `sl[:]` এর ভ্যালু কত হবে? কেন?

আমরা `range()` ফাংশনে step এর ব্যবহার দেখেছিলাম । লিস্টের ক্ষেত্রেও step ব্যবহার করা যায় । যেমন:

```
print sl[0:10:2]
print sl[0:9:3]
```

অর্থাৎ শেষে আরেকটি কোলন দিয়ে আমরা step ভ্যালুটি নির্দেশ করে থাকি । তাই প্রথম ক্ষেত্রে আমরা 0-তম আইটেম থেকে শুরু করে 2টি আইটেম বাদ দিয়ে দিয়ে 10-তম আইটেমের আগের আইটেম পর্যন্ত যে আইটেগুলো আছে সেগুলোর লিস্ট পাব । নিজে নিজে বোঝার চেষ্টা করি দ্বিতীয় ক্ষেত্রে কি ঘটছে ।

যে কোন ভ্যালুর আগে মাইনাস চিহ্ন দিলে তার অবস্থান বিপরীত দিক থেকে বিবেচনা করা হয় । তাই শেষ দিক থেকে 5-তম আইটেমের ভ্যালু হবে `sl[-5]` । এভাবে শেষ দিক থেকে 2-তম আইটেমের আগ পর্যন্ত আইটেমগুলোর লিস্ট পাব: `sl[:-2]` । step এর ভ্যালু নেগেটিভ হলে গনণা উল্টো দিকে হবে । যেমন শেষ দিক থেকে 2-তম আইটেমের আগের আইটেম থেকে শুরু করে 3-তম আইটেম পর্যন্ত আইটেমগুলো 2 ধাপ করে পেছালে আমরা যে লিস্টটি পাব তার জন্য আমাদের কে লিখতে হবে : `sl[-2:3:-2]`

এভাবে নিজেরা ইচ্ছামত লিস্ট তৈরি করে তার বিভিন্ন অংশ আলাদা করার চেষ্টা করি । প্রথমবার দেখে লিস্টের সিন্ট্যাক্স খুব জটিল মনে হতে পারে । কিন্তু কিছুদিন অনুশীলন করলেই ঠিক হয়ে যাবে । পাইথনের চমৎকার ফিচারগুলোর মধ্যে অন্যতম হল লিস্ট এর এই ব্যবহার । একটি লিস্ট এর যে কোন অংশ নিয়ে আরেকটি লিস্ট খুব সহজেই তৈরি করা যায়।

আমি এর আগে `dir()` ফাংশনের কথা উল্লেখ করেছিলাম । পাইথনে কোন নাম সম্পর্কে টেকনিক্যাল বিষয় (প্রোপার্টিজ, মেথডস ইত্যাদি) জানার জন্য আমরা এই ফাংশনটি ব্যবহার করি । আসুন ঝটপট একটি লিস্টের উপর এই ফাংশনটি প্রয়োগ করি :

```
>>> list = [1,2,3]
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__delsli
ce__', '__doc__', '__eq__', '__ge__', '__getattribute__', '__getitem__', '__gets
lice__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '
__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__r
educe_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__
', '__setslice__', '__str__', 'append', 'count', 'extend', 'index', 'insert', 'p
op', 'remove', 'reverse', 'sort']
>>>
```

আমরা দেখতে পাচ্ছি একটি লিস্টের অনেকগুলো মেথড ও প্রোপার্টি রয়েছে । যেগুলোর আগে এবং পরে __ (ডাবল আন্ডারস্কোর) রয়েছে সেগুলো নিয়ে আমরা মাথা ঘামাব না । বাকিগুলোর মধ্যে আমরা দেখতে পাচ্ছি:

- `append()`
- `count()`
- `extend()`
- `index()`
- `insert()`
- `pop()`
- `remove()`
- `reverse()`
- `sort()`

কোন লিস্টের শেষে আরেকটি আইটেম যোগ করতে আমরা append() ব্যবহার করি । যেমন:

```
>>> l = [1,2]
>>> l.append(3)
>>> print l
[1, 2, 3]
>>>
```

এখানে l একটি লিস্ট যেটার শেষে আমরা 3 যোগ করলাম । কোন লিস্টে একটি আইটেম কতবার আছে তা জানার জন্য আমরা count() ব্যবহার করি । যেমন:

```
>>> l = [1,1,1,1,1,2,2,2,4,4,4,5,5,5,5,5,5,5,5,7,7,2,1,3]
>>> l.count(1)
6
>>> l.count(7)
2
>>> l.count(3)
1
>>> l.count(5)
9
>>> l.count(4)
3
>>> l.count(2)
4
>>>
```

তাহলে দেখলাম l লিস্টটিতে বিভিন্ন আইটেম কতবার আছে তা কিভাবে বের করা যায় । লক্ষ্য করুন অন্যান্য অনেক প্রোগ্রামিং ল্যাঙ্গুয়েজে count বা length এই ধরনের ফাংশন, মেথড বা প্রোপার্টি দিয়ে লিস্টের আকার বা আইটেমের সংখ্যা নির্নয় করা হয় । পাইথনে count এর ব্যবহারটি কিছুটা ভিন্ন । আর পাইথনে একটি লিস্ট এর আইটেম সংখ্যা বের করতে আমরা `len()` ফাংশনটি ব্যবহার করি । যেমন:

```
lst = [1,1,1,2,2,2,2,4,4,4,4]
print len(lst)
```

এই কোড রান করে দেখুন আউটপুট কি দেখায় । একটি লিস্টের শেষে আরেকটি লিস্ট যোগ করতে আমরা `extend()` ব্যবহার করি । যেমন:

```
>>> lst1 = [1,2]
>>> lst2 = [2,3]
>>> lst1.extend(lst2)
>>> lst1
[1, 2, 2, 3]
>>>
```

এখানে `lst1` লিস্টটির শেষে `lst2` যোগ করলাম । এর ফলে `lst1` এর আইটেমগুলোর সাথে `lst2` এর আইটেমগুলোও যুক্ত হয়ে গেল ।

এবার আসা যাক `index()` এ । কোন লিস্টে কোন আইটেম এর অবস্থান বা ইন্ডেক্স জানতে আমরা এটি ব্যবহার করি । লিস্টে যদি ঐ আইটেম একাধিকবার থাকে তাহলে প্রথম অবস্থানটি পাওয়া যাবে। যেমন:

```
>>> lst1 = [1,2,2,3]
>>> lst1.index(2)
1
>>>
```

এখানে 2 আইটেমটি দুবার এসেছে – 1 এবং 2 ইন্ডেক্সে । তাই প্রথম অবস্থানটি পেলাম আমরা ।

`insert()` – এটি ব্যবহার করে আমরা একটি লিস্টের যে কোন অবস্থানে আরেকটি আইটেম যোগ করতে পারি । যেমন:

```
>>> l = [1,2,3]
>>> l.insert(2,4)
>>> l
[1, 2, 4, 3]
>>>
```

দেখুন `insert()` মেথডটি দুটি প্যারামিটার গ্রহন করে । প্রথম প্যারামিটারটি দ্বারা আমরা নির্দেশ করি কোন অবস্থানে আইটেমটি বসাতে হবে, দ্বিতীয় প্যারামিটার হিসেবে আমরা সরাসরি আইটেমটিকে পাস করি । অর্থাৎ, `l.insert(2,4)` এর অর্থ হল `l` লিস্টের 2-তম অবস্থানে 4 আইটেমটিকে যোগ করা ।

এবার আসা যাক `pop()` মেথডে । এই মেথডটি লিস্টের সর্বশেষ আইটেমটি রিটার্ন করে এবং ঐ লিস্ট থেকে এটিকে বাদ দিয়ে দেয় । যেমন:

```
>>> l = [2,5,4]
>>> l
[2, 5, 4]
>>> s = l.pop()
>>> s
4
>>> l
[2, 5]
>>>
```

দেখা যাচ্ছে এখানে `l.pop()` মেথড কলটি। লিস্টের সর্বশেষ আইটেম 4 কে রিটার্ন করছে (যেটিকে আমরা s নামে সংরক্ষণ করলাম) । এবং এই একই সাথে। লিস্টটি থেকে এই শেষ আইটেম 4 কেও বাদ দিয়ে দিয়েছে ।

এবার আসা যাক `remove()` মেথডে । এটি `insert()` মেথডের ঠিক উলটো কাজ করে থাকে । এটিকে কোন অবস্থান দেখিয়ে দিলে সেই অবস্থানের আইটেমটিকে রিমুভ করাই এর কাজ । আসুন একটি উদাহরণ দেখে নেই:

```
>>> l = [2,5,4]
>>> l.remove(2)
>>> l
[5, 4]
>>>
```

আমরা দেখলাম কিভাবে `remove()` মেথডের সাহায্যে যে কোন একটি আইটেম আমরা রিমুভ করতে পারি । এই মেথডটি ছারাও আমরা পাইথনের বিল্ট ইন ফাংশন `del()` ব্যবহার করেও কোন আইটেম রিমুভ করতে পারি । যেমন:

```
l = [2,5,4,4]
del(l[3])
print l
```

এরপরে চলুন দেখে নেই `sort()` এবং `reverse()` মেথডের ব্যবহার । `sort()` ব্যবহার করে যে কোন লিস্টকে সর্ট বা স্বাভাবিকভাবে সাজানো হয়:

```
>>> s = [2,5,1,6,2,9,3]
>>> s.sort()
>>> s
[1, 2, 2, 3, 5, 6, 9]
>>>
```

`reverse()` মেথডটি লিস্ট কে উলটো ভাবে সাজানোর জন্যে ব্যবহার করা হয়:

```
>>> l = ['a', 'b', 'c']
>>> l.reverse()
>>> l
['c', 'b', 'a']
>>>
```

অপারেটরস

পাইথনে কমন কিছু অপারেটর আছে যেগুলো নানা ধরনের অপারেশনে সহায়তা করে থাকে । এর মধ্যে বেশীরভাগই ম্যাথ সিম্বল যেগুলো দিয়ে আমরা ম্যাথমেটিক্যাল অপারেশন চালাতে পারি ।

`+` প্লাস বা যোগ

`-` মাইনাস বা বিয়োগ

`/` স্ল্যাশ বা ভাগ

`*` এস্টেরিস্ক বা গুন

`%` পার্সেন্ট বা মডুলাস বা ভাগশেষ

`<` লেস দ্যান বা ক্ষুদ্রতর

`>` গ্রেটার দ্যান বা বৃহত্তর

`<=` লেস দ্যান অর ইকুয়াল অর্থাৎ ক্ষুদ্রতর অথবা সমান

`>=` গ্রেটার দ্যান অর ইকুয়াল অর্থাৎ বৃহত্তর অথবা সমান

ইংরেজী নাম গুলো উল্লেখ করলাম কারণ অন্য কোথাও পাইথন রিলেটেড কিছু পড়তে গেলে সেটা ইংরেজীতে হবে সেটাই স্বাভাবিক । তাই এই ইংরেজী নামগুলোই আমাদের শেখা উচিৎ ।

এবার উদাহরণসহ দেখা যাক এদের কোনটার কাজ কি:

যোগ বিয়োগ গুন ভাগ

এগুলোর কাজ বোধহয় বলার অপেক্ষা রাখে না । আমরা কিছু উদাহরণ দেখব।

```
num = 3
num = num + 1
print num # Output: 4

num = num - 2
print num # Output: 2

num = num * 8
print num # Output: 16

num = num / 4
print num # Output: 4
```

আপনারা এই কোডটি কোন পাইথন ফাইলে টাইপ করে (কখনোই কপি পেস্ট করবেন না প্লিজ) রান করে দেখুন । এবার পাইথনের ইন্টারএক্টিভ শেলে নিজের ইচ্ছামত কিছু যোগ বিয়োগ গুন ভাগ করে দেখুন পাইথন এর এই শেলটিকে আসলে ক্যালকুলেটর হিসেবে ব্যবহার করা যায় কিনা :)

আমি করলাম:

```
C:\Users\maSnun\Desktop>python
Python 2.5.4 (r254:67916, Dec 23 2008, 15:10:54) [MSC v.1310 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 3+ 5
8
>>> (3+5)*(8/2)+(4-3)
33
>>>
```

একটু লক্ষ্য করলে দেখবেন পাইথনও ক্যলকুলেশন করার সময় BODMAS এর প্যটার্ন ফলো করে । এটাকে অপারেটর প্রিসিডেন্সও বলা হয় ।

পার্সেন্ট বা মডুলাস (%)

এটি দিয়ে আমরা ভাগশেষ বের করি । যেমন:

```
print 15 % 7 #Output: 1
```

১৫ কে ৭ দিয়ে ভাগ করলে ভাগশেষ ১ থাকে, এটা কে না জানে, কিন্তু বলুন তো ১৫৭৯ কে ৩৭ দিয়ে ভাগ দিলে ভাগশেষ কত আসবে? নিজেই করে দেখুন না, অবশ্যই পাইথন ব্যবহার করে ।

কম্প্যারিজন অপরেটরস (<, >, <=, >=):

এগুলো দিয়ে আমরা দুটি সংখ্যা বা এক্সপ্রেশনের মানের তুলনা করতে পারি । যেমন:

```
>>> 5 < 4
False
>>> 5 > 3
True
>>> 4 > 5
False
>>> 4 < 5
True
>>> 5 <= 5
True
>>> 5 <= 10
True
>>> 5 >= 6
False
>>> 5 >= 5
True
>>> 5 >= 3
True
>>>
```

অর্থাৎ আমরা যখন টাইপ করব 5 > 4, পাইথন আমাদেরকে জানিয়ে দিবে এটা সত্যি (সম্ভব) কিনা । যেমন: 5 > 4 এ আমরা পাব True কেননা ৪ অপেক্ষা ৫ বড় কিন্তু 5 > 10 এ পাব False কারণ ৫ কোনভাবেই ১০ এর চেয়ে বড় না ।

কিন্তু 5 >= 5 কেন True? কারণ ইকুয়াল সাইন থাকলে বোঝায় “সমান অথবা বৃহত্তর” । এখানে ৫ যেহেতু ৫ এর সমান সেহেতু এটি True ।

এভাবে নিজে নিজে কিছু সংখ্যা নিয়ে এই প্রতীকগুলো ব্যবহার করে দেখুন । আশাকরি আরো ভাল ধারণা পাবেন ।

এবার দেখা যাক জটিল এক্সপ্রেশন কিভাবে বিশ্লেষন করা সম্ভব:

```
>>> (35 - 14) > (1001 - 999 )
True
>>>
```

এক্ষেত্রে ৩৫ থেকে ১৪ বাদ দিলে থাকছে ২১, অপরদিকে ১০০১ থেকে ৯৯৯ বাদ দিলে থাকছে ২ । ২১ অবশ্যই ২ অপেক্ষা বড়, তাই এক্সপ্রেশনের ভ্যালু হবে True.

এই জিনিসগুলো বার বার নিজে অনুশীলন করে নিন, আশা করি পাইথনে সাধারণ হিসাব নিকাশ খুব সহজেই করে নিতে পারবেন ।

কন্ট্রোল ফ্লো

একটি প্রোগ্রাম কিভাবে চলবে, কোন ধাপের পর কোন ধাপ অনুসরন করবে সেই সিদ্ধান্ত নেওয়াই কন্ট্রোল ফ্লো । আমরা এই চ্যাপ্টারে দেখবো কিভাবে আমরা একটি পাইথন প্রোগ্রামের ফ্লো কন্ট্রোল করতে পারি ।

ইফ স্টেটমেন্ট

ইফ স্টেটমেন্ট ব্যবহার করে আমরা কোন একটি ভ্যালুর উপর নির্ভর করে প্রোগ্রাম এর ফ্লো কন্ট্রোল করতে পারি ।

```
if x < 0:
    x = 0
    print 'Negative changed to zero'
elif x == 0:
    print 'Zero'
elif x == 1:
    print 'Single'
else:
    print 'More'
```

`if` এর পর আমরা একটি এক্সপ্রেশন ব্যবহার করি, যদি এই এক্সপ্রেশনের ভ্যালু বুলিয়ান `True` এর সমতুল্য হয় তাহলে এই স্টেটমেন্টের সাথে সংশ্লিষ্ট কোড ব্লকটি রান করে । আর যদি `True` না হয় তবে পাইথন ঐ কোড ব্লক স্কিপ করে যায় ।

`elif` ব্যবহার করে আমরা একাধিক `if` পার্ট যোগ করতে পারি । যদি `if` বা কোন `elif` ব্লকই এক্সিকিউটেড না হয় (অর্থাৎ কোনটার ভ্যালুই `True` না হয়) তাহলে `else` ব্লকের কোড রান করে ।

উল্লেখ্য `else` কিংবা `elif` অপশনাল । `elif` হলো `else if` এর সংক্ষিপ্ত রুপ ।

ফর স্টেটমেন্ট

পাইথনে `for` একটি সিকুয়েন্স (যেমন: `list` , `set` , `dictionary` , `tuple` ইত্যাদি) এ থাকা আইটেমগুলো একটার পর একটা এ্যাক্সেস করে এবং প্রতিবার একটি লোকাল ভ্যারিয়েবলে ঐ আইটেমটি পাস করে দেয় । আমরা ফর স্টেটমেন্টের সাথে থাকা কোড ব্লকে এই ভ্যারিয়েবল নিয়ে কাজ করতে পারি । একটি আইটেম প্রসেস করা শেষ হলে ফর স্টেটমেন্টটি পরবর্তী আইটেম এ মুভ করে । এটাকে ইটারেশন বলা হয় ।

অর্থাৎ ফর স্টেটমেন্ট যে কোন সিকুয়েন্স বা ইটারেটর এর লুপিং বা ইটারেশনের কাজে ব্যবহার করা হয় ।

```
words = ['cat', 'window', 'defenestrate']
for w in words:
    print w, len(w)
```

হোয়াইল স্টেটমেন্ট

হোয়াইল স্টেটমেন্টে পাস করা এক্সপ্রেশন যতক্ষন পর্যন্ত `True` হয় ততক্ষন পর্যন্ত লুপ চলতে থাকে । পাইথন ইন্টারপ্রেটার প্রতিবার হোয়াইল লুপে ঢুকে প্রথমেই কন্ডিশন টা চেক করে নেয়, যদি `True` হয় তাহলে সাথে থাকা কোড ব্লক এক্সিকিউট করে, তারপর আবার কন্ডিশন চেক করে নতুন করে শুরু করে - এভাবেই লুপটি চলতে থাকে । যখন ঐ এক্সপ্রেশনের ভ্যালু `False` হয় তখন পাইথন কোড ব্লকটি স্কিপ করে পরবর্তী স্টেটমেন্ট বা এক্সপ্রেশনে চলে যায় ।

```
i = 0
while i < 10:
    print i
    i = i + 1
```

যেমন এখানে যতক্ষন পর্যন্ত `i` এর ভ্যালু 10 এর নিচে থাকবে ততক্ষন পর্যন্ত লুপ চলবে । প্রতিবারই `i` এর মান এক করে বাড়তে বাড়তে যখন 10 এর ছোট থাকবে না তখন লুপ শেষ হয়ে যাবে ।

ব্রেক, কন্টিনিউ এবং পাস

কোন লুপ থেকে বের হতে `break` স্টেটমেন্টটি ব্যবহার করা হয় । যেমন:

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print n, 'equals', x, '*', n/x
            break
```

কোন লুপের কোন একটি ইটারেশন স্কিপ করে পরবর্তী ইটারেশনে মুভ করার জন্য `continue` স্টেটমেন্টটি ব্যবহার করা হয় । এক্ষেত্রে `continue` স্টেটমেন্ট এর পর থেকে ঐ ইটারেশনের আর কোন কোড এক্সিকিউট করা হয় না, বরং সরাসরি পরবর্তী ইটারেশনের শুরুতে চলে যায় ।

```
for num in range(2, 10):
    if num % 2 == 0:
        print "Found an even number", num
        continue
    print "Found a number", num
```

`pass` স্টেটমেন্টটি বেশ মজার । এটি কোন কাজ করে না, এটাকে মূলত প্লেইস হোল্ডার হিসেবে ব্যবহার করা হয় । যেসব জায়গায় পাইথন কোন না কোন স্টেটমেন্ট আশা করে কিন্তু আমরা কোন স্টেটমেন্ট দিতে চাই না, তখন আমরা `pass` ব্যবহার করি ।

যেমন একটি ফাকা ফাংশন বডি:

```
def does_nothing(*args):
    pass
```

এমনিতে পাইথন ফাংশন এ অবশ্যই একটি বডি থাকা লাগবে যেখানে ফাংশনের মূল কাজটি করার কথা । কিন্তু আমরা আপাতত একটি ফাংশন চাই যেটা কোন কাজ করবে না, তাই আমরা `pass` ব্যবহার করলাম যাতে পাইথন ইন্টারপ্রেটার কোন এরর থ্রো না করে ।

ফাংশন

আমাদের প্রোগ্রামের যে অংশগুলো বার বার আসে সেগুলোকে আমরা পুনরায় ব্যবহারযোগ্য একক (reusable unit) হিসেবে ব্যবহার করতে পারি ফাংশনের সাহায্যে । গনিতে যেমন দেখেছি কোন ফাংশন একটি ইনপুট নিয়ে সেটার উপর বিভিন্ন ধরনের ম্যাথ করে আউটপুট দেয়, প্রোগ্রামিংএও সেই একই ব্যাপার ঘটে । আপনি এক বা একাধিক প্যারামিটার পাস করবেন একটি ফাংশনে, ফাংশনটি প্রসেস করে আপনাকে আউটপুট "রিটার্ন করবে" । তবে প্রোগ্রামিং এর ক্ষেত্রে সবসময় যে ইনপুট থাকতে হবে বা আউটপুট দিতে হবে এমন কোন কথা নেই ।

একটি ফাংশন আসলে কিছু স্টেটমেন্টের সংকলন । যখনই কোন ফাংশন কল করা হয় তখন এই ফাংশনের ভিতরে থাকা স্টেটমেন্টগুলো এক্সিকিউট করা হয় । পাইথনে আমরা ফাংশন ডিক্লেয়ার করার জন্য def কি-ওয়ার্ডটি ব্যবহার করি । আসুন দেখে নেই একটি ফাংশন:

```
def hello():
    print "Hello World!"
    return 0
```

প্রথমে আমরা def কি-ওয়ার্ডটি লিখেছি । তারপার ফাংশনের নাম – "hello", এবং তারপর (). যদি আমরা ফাংশনটিতে কোন ইনপুট দিতে চাই সেক্ষেত্রে প্যারামিটারগুলো এই () এর মধ্যে কমা দিয়ে আলাদা করে লিখতে হবে । আমরা দেখতে পাচ্ছি, এই ফাংশনে আমরা কোন ইনপুট দিচ্ছি না । ফাংশনটি "Hello world!" প্রিন্ট করবে । সি-প্রোগ্রামিং এর সাথে মিল রেখে (এবং রিটার্ন স্টেটমেন্টের ব্যবহার দেখানোর জন্য) আমরা 0 রিটার্ন করছি । আসলে এই স্টেটমেন্টের কোন দরকার ছিল না ।

এবার আসুন দেখা যাক পাইথনে কিভাবে আমরা ফাংশন প্যারামিটার পাস করব ।

```
def sayHello(name):
    print "Hello, "+name+" !"
```

এই ফাংশনটিকে কল করুন এভাবে: `sayHello("maSnun")`

কমান্ড লাইন

কমান্ড লাইনে নানা ধরনের কাজ করার জন্য সচারচর ব্যবহৃত পাইথন ফাংশন, ক্লাস এবং মডিউল নিয়ে আলোচনা করবো আমরা ।

ব্যবহারকারীর ইনপুট নেওয়া

পাইথনে প্রোগ্রামিং করতে করতে এমন অনেক সময় আসবে যখন আমাদের প্রয়োজন পড়বে ব্যবহারকারীর কাছ থেকে কিছু ইনপুট নেওয়া । এজন্য আমরা `raw_input()` ফাংশনটি ব্যবহার করতে পারি:

```
print "What is your name?"
name = raw_input()
print "You said your name is %s" % name
```

মজার ব্যাপার হচ্ছে এই `raw_input` ফাংশনটির সাথে চাইলে আমাদের প্রশ্নটিও এভাবে পাস করে দিতে পারি:

```
name = raw_input("What is your name?")
print "You said your name is %s" % name
```

এই ফাংশানটি ইউজার ইনপুট স্ট্রিং হিসেবে পাস করবে । সুতরাং আমরা যদি সংখ্যা নিতে চাই তাহলে এটাকে পরিবর্তন করে নিতে হবে:

```
number1 = int(raw_input("N1:"))
number2 = int(raw_input("N2:"))
print number1 + number2
```

এখানে আমরা `int()` ফাংশনটি ব্যবহার করে স্ট্রিং ভ্যালু থেকে এটিকে ইন্টিজারে কনভার্ট করে নিয়েছি ।

পাইথনে ইনপুট নেওয়ার জন্য আরেকটি ফাংশন রয়েছে - `input()` কিন্তু সমস্যা হচ্ছে, `raw_input()` যেখানে স্ট্রিং ভ্যালু আশা করে, `input()` সেখানে পাইথন কোড আশা করে । অর্থাৎ যেই ইনপুট দেওয়া হবে, পাইথন তা কোড হিসেবে বিবেচনা করে আউটপুট দেখাবে । ইউজার যদি ইচ্ছা করে ক্ষতিকর কোড টাইপ করে দেন তাহলে এটার মাধ্যমে সিস্টেমের ক্ষতিসাধন করা সম্ভব তাই এটার ব্যবহার অনুৎসাহিত করা হয় ।

কমান্ড লাইন আর্গুমেন্টস এবং `sys.argv`

আমরা পাইথন প্রোগ্রাম রান করার সময় কিছু অপশন বা আর্গুমেন্ট পাস করতে পারি । এই অপশন বা আর্গুমেন্টগুলো আমরা ফাইল নেইমের পরে একটা একটা করে স্পেইস দিয়ে আলাদা করে দিতে পারি । পাইথন এই আর্গুমেন্টগুলোকে `sys` মডিউলের `argv` লিস্টে জমা করে । একটি উদাহরন দেখে নেই:

`main.py` ফাইলের কন্টেন্ট:

```
import sys
print sys.argv
```

টার্মিনালে রান করলাম এভাবে:

```
$ python main.py 1 2 3
```

এখানে 1, 2 এবং 3 প্রত্যেকে একেকটি কমান্ড লাইন আর্গুমেন্ট ।

আউটপুট:

```
['main.py', '1', '2', '3']
```

`sys.argv` এর প্রথম আইটেমটি হয় ঐ ফাইল নেইম । এরপর কমান্ড লাইন আর্গুমেন্টগুলো থাকে একটার পর একটা ।