Mymensingh Polytechnic Institute,Mymensingh.

Welcome To My Python Programming Class.

Fatema Zohura

Chief Instructor

Computer Science & Technology

Mymensingh Polytechnic Institute, Mymensingh.

Subjcet Name: - Python Programming.

Subjcet Code : 28521

Variables & Data Type.

Computer Science & Technology

2nd Semester (1st shift)

ডাটা ঃ- প্রোগ্রামে ব্যবহৃত যে কোনো মানই ডাটা।

উদাহরণ ঃ

- পূর্ণ সংখ্যা(০,১,২,১৫,২৫,১০০)
- প্রকৃত সংখ্যা (১০.২৫,৩০.২৫,১০০.২০)
- বিভিন্ন বর্ণ প্রভৃতি।

পাইথনে ব্যবহৃত ডাটাসমূহ বিভিন্ন ধরনের ডাটাসমূহ মূলত ক্যারেক্টার ব্যবহার কওে লিখতে হয়।

ক্যারেক্টার ৪টি গ্রুপে ভাগ করা হয়।

- বর্ণ
- সংখ্যা
- বিশেষ ধরনের ক্যারেক্টার
- হোয়াইট স্পেস

টোকেন ঃ

টোকেন ঃ-Token ইংরেজি শব্দটির বাংলা অর্থ হচ্ছে প্রতিক বা চিহ্ন ।যে কোনো প্রোগ্রাম কতকগুলো স্টেটমেন্ট নিয়ে গঠিত। আবার প্রতিটি স্টেটমেন্ট কতকগুলো Word এবং Character সমষ্টি। পাইথনে ব্যবহৃত এরুপ Word বা Character ও Symbol সমূহকে একত্রে টোকেন বলে।

টোকেন	ব্যবহার
কী-ওয়ার্ড (Keyword)	প্রোগ্রামে কোড় লিখার জন্য ব্যবহৃত হয়।
আইডেন্টিফায়ার (Identifier)	ভেরিয়েবল,ফাংশন,স্ট্রাকচার,ক্লাস,ইত্যাদি নামকরণের জন্য ব্যবহৃত হয়।
স্ট্রিং (String)	প্রোগ্রামে একগুচ্ছ ক্যারেক্টার নিয়ে কাজ করার জন্য ব্যবহৃত হয়।
পাঙ্কচুয়েটর (Punctuator)	কী-ওয়ার্ড, আইডেন্টিফায়ার,অপারেটর ও অপারেন্ড,মধ্যে পার্থক্য করার জন্য ব্যবহৃত হয়।
স্পেশাল সিম্বল (Special symbol)	বিশেষ কাজে ব্যবহার করার জন্য ।
অপারেটর,অপারেন্ড এবং এক্সপ্রেশন	ডবভিন্ন গাণিতিক ও যৌক্তিক অপারেশন সম্পন্ন করার জন্য।

পাইথনে ভেরিয়েবল নামকরণের নিয়মাবলি ঃ

কম্পাইলারের সীমাবদ্ধতার কারণে পাইথনে ভেরিয়েবল লেখার সময বেশ কিছু নিয়ম,কানুন মেনে ভেরিয়েবল ডিক্লেয়ার করতে হয়।

১. ভেরিয়েবলের নামের মধ্যে কোনো খালি জায়গা বা স্পেস ব্যবহার করা যাবে না।যেমন- Rectangle area,First number,Summation value ইত্যাদি।

২.ভেরিয়েবল নাম লেখার ক্ষেত্রে প্রথম অক্ষর অবশ্যই একটি Alphabetic letter(upper case or lower case) অথবা Underscore(_) হবে। যেমন- Number,Length,Width,-Length লেখা যাবে কিন্তু1number,@num1,%def লেখা যাবে না।যদিও ভেরিয়েবল শুরুতে Underscore ব্যবহার করা যায়,কিন্তু পাইথনে কনভেনশন হচ্ছে ভেরিয়েবলের নাম সবসময় lower case letter দিয়ে শুরু করা।

৩. প্রথম অক্ষরের পর যে-কোনো Letter,underscore,number ব্যবহার করা যাবে।যেমন-Number1,Rectangle_Area,Area_OF_The_Rectangle.

- ৪. Underscore(_) ভেরিয়েবলে ছাড়া অন্য কোনো বিশেষ চিহ্ন (!,@,#,$,%,”,&,,*,<,>,?,{},[]) ব্যবহার করা যাবে না । যেমন- H#,R#,email@,Age! ইত্যাদি ব্যবহার করা যাবে না।

৫.পাইথন একটি Case sensitive letter। তাই এতে letter,Uppercase letter এর Lowercase letter ভিন্ন ভিন্ন অর্থ বিদ্যমান । অর্থাৎ Area,area,ARea ইত্যাদিকে ভিন্ন ভিন্ন ভেরিয়েবল হিসেবে ডিক্লেয়ার করা যাবে কিন্তুArea,Area কে দুটি আলাদা ভেরিয়েবল হিসেবে ডিক্লেয়ার করা যাবে না।

৬. পাইথনে ভেরিয়েবলর নাম হিসেবে এর কী-ওয়ার্ড সমুহ কে ব্যবহার করা যাবে না। যেমন-

If,else,elif,for,while,break,continue,except,as,in,true,none,def,del,class ইত্যাদি।

৭.ভেরিয়েবল নাম হিসেবে অর্থবোধক নাম ব্যবহার করাই যুক্তিযুক্ত ।অনর্থক নাম পরিহার করাই শ্রেয়। যেমন:আয়তক্ষেত্রের ক্ষেত্রফল নির্ণয় করার জন্য Length,Width,Rectangle_Area ব্যবহার করাই শ্রেয়; A,B,R নয়।

ভেরিয়েবল ও ভেরিয়েবলের মান নির্ধারণঃ

ভেরিয়েবল ঃ ভেরিয়েবল হচ্ছে কম্পিউটার মেমরি সেই নির্ধারিত জায়গা,য়েখানে বিভিন্ন মান জমা করে রাখা যায়। যেমন-

number1=10

Name="Rahim"

ভেরিয়েবলের মান নির্ধারণ ঃ পাইথনে সমতা চিহ্নের(=) মাধ্যমে কোনো ধরনের মান ভেরিয়েবল হিসেবে স্টোর করা যায়,এক্ষেত্রে আলাদা করে কোনো ডিক্লারেশনের প্রয়োজন নেই।সমতা চিহ্নের বাম পাশে ভেরিয়েবলের নাম এবং ডান পাশের সংখ্যাটি ভেরিয়েবলের মান নির্দেশ করে। যেমন-

```
>>> num = 10
>>> print(num)
10
>>> print(num*2)
20
>>> print(num+5)
15
```

ভেরিয়েবলের মান পুনঃনির্ধারণ ঃ পাইথনে কোনো ভেরিয়েবলের মধ্যে একধিকবার নতুন নতুন ভ্যালু স্টোর করা যায়। এভাবে কোনো ভেরিয়েবলের মান বার বার নির্ধারন করার এই প্রক্রিয়াকে রিঅ্যাসাইনমেন্ট অফ ভেরিয়েবল বা ভেরিয়েবলের মান পূনঃনির্ধারন বলে। যেমন-

```
>>>  x = 110
>>> x
110
>>>x="karim"
>>> x
karim
```

একাধিক ভেরিয়েবলের মান নির্ধারণ ঃ

পাইথনে একই লাইনে একাধিক ভেরিয়েবল ডিক্লারেশনের সুবিধা প্রদান করে। এতে একাধিক ভেরিয়েবলের জন্য একটি নির্দিষ্ট মান স্টোর করা যায়। আবার একাধিক ভেরিয়েবলের জন্য যথাক্রমে মানও এক সাথে স্টোর করা যায়। পাইথনে ভেরিয়েবল ডিক্লারেশনের এই পদ্ধতি কে মাল্টিপল ভেরিয়েবল অ্যাসাইনমেন্ট বলে।

যেমন,

```
a=b=c=10
 and
a,b,c = 30,20,"shuvo"
```

ডাটা টাইপ ঃ

সহজ কথায়,প্রোগামে ব্যবহৃত যে কোনো মানই ডাটা।

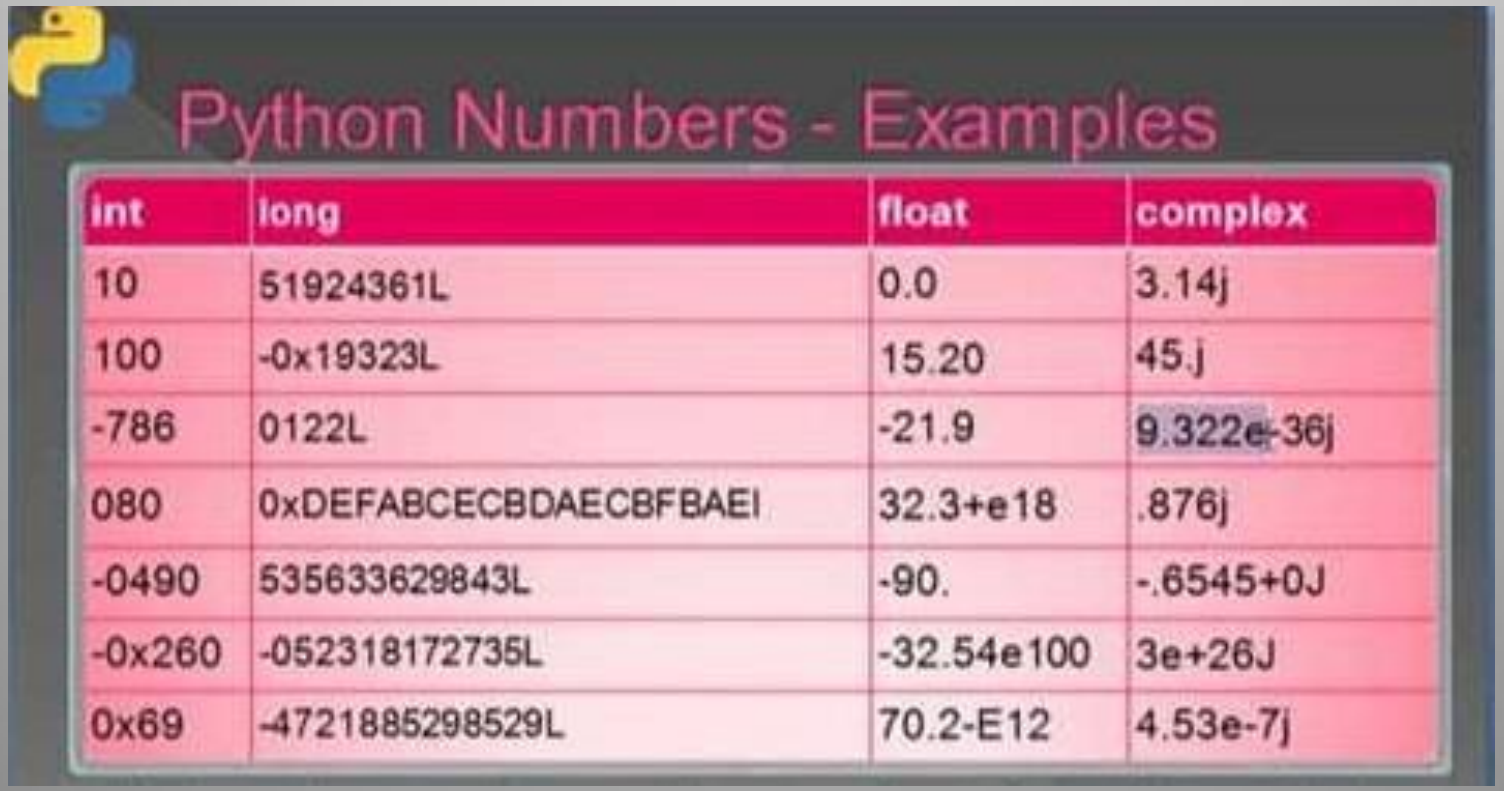
ডাটার মান,ধরন এবং মেমরি স্পেস সংরক্ষণের ভিত্তিতে পাইথনে পাচঁ ধরনের স্ট্যান্ডার্ড ডাটা টাইপ রয়েছে।যেমন,

- সংখ্যা
- স্ট্রিং
- লিস্ট
- টাপল
- ডিকশনারি

সংখ্যা(Number) ঃ Number হচ্ছে যে- কোনো প্রকারের সংখ্যা।

পাইথনে ০৪ ধরনের সংখ্যা সার্পোট করে।

- IntNumber (ছাট পূর্ণ সংখ্যা)
- Long (বড় পূর্ণ সংখ্যা,অক্টাল,কিংবা হেক্সাডেসিম্যাল আকাওে প্রকাশ করা যায়।
- Float (দশমিক বিশিষ্ট সংখ্যা)
- Complex (জটিল সংখ্যা)



স্ট্রিং ঃ

স্ট্রিং হচ্ছে সিঙ্গেল কোটেশন (’ ’) বা ডাবল কোটেশন (’’ ’’) এর ভেতর ব্যবহৃত শূন্য বা তার অধিক অক্ষর(সংখ্যা,চিহ্ন বা বর্ণ)।

যেমন- (‘abc’)

- লিস্ট ঃ লিস্ট হলো স্কয়্যার ব্র্যাকেট([ ]) এ আবদ্ধ ও কমা (,) দিয়ে আলাদা করা আইটেম আমরা যখন একধিক ডাটা এক সঙ্গে রাখতে চাই, তখন লিস্ট ব্যবহার করতে পারি।
  যেমন-
- List= [‘karim’,782,4.5,’rahim’,60.2]

- টাপল ঃ টাপল আর লিস্ট মূলত একই রকম,শুধু পার্থক্য হচ্ছে যে টাপল এ প্যারেনথেসিস() ব্যবহৃত হয়,কিন্তু লিস্ট এ ব্র্যাকেট [] ব্যবহৃত হয়।

  যেমন-

List= (‘karim’,782,4.5,’rahim’60.2)

ডিকশনারি ঃ ডিকশনারি তে কী- ভ্যালু জোড়ায় জোড়ায় থাকে।যে কোনো ডাটা টাইপ কী হতে পারে,যদিও সাধারণ নাম্বার বা স্ট্রিং ই কী হিসেবে রেকর্ডেড হয়। অন্যদিকে যে কোনো সংখ্যা/অবজেক্টই ভ্যালু হিসেবে রেকর্ড হতে পারে।ডিকশনারিকে ব্র্যাকেট { } এর সাহায্যে প্রকাশ করা হয় এবং ব্র্যাকেটের [ ] সাহায্যে ডিকশনারি তে ভ্যালু এসাইন করা হয।

৩.৪ ডাটা টাইপ কনভার্সন

এক টাইপ ভেরিয়েবলকে অন্য টাইপ ভেরিয়েবলে কনভার্ট করার প্রক্রিয়াকে ডাটা টাইপ কনভার্সন বলে্ একে টাইপ কাস্টিং ও বলে।

ফাংশন	বর্ণনা
int (x[,base])	স্ট্রিং x কে পূর্ণসংখ্যার পরিবর্তন করার জন্য এ ফাংশটি ব্যবহৃত হয়।
long(x[,base])	স্ট্রিং x কে একটি দীর্ঘ পূর্ণসংখ্যার পরিবর্তন করার জন্য এ ফাংশটি ব্যবহৃত হয়।
float(x)	ভেরিয়েবর x কে দশমিক বিশিষ্ট সংখ্যায় রুপান্তর করার জন্য এ ফাংশনটি ব্যবহৃত হয়।
chr(x)	ইন্টিজার টাইপ ডাটাকে ক্যারেক্টার টাইপ ডাটায় রুপান্তক করার জন্য এ ফাংশটি ব্যবহৃত হয়।
hex(x)	পূর্ণ সংখ্যার মানকে হেক্সাডেসিম্যাল মানে রুপান্বর করার জন্য এ ফাংশনটি ব্যবহৃত হয়।
str(x)	ভেরিয়েবল x কে string এ রুপান্তর করার জন্য এ ফাংশনটি ব্যবহৃত হয়।
tuple(x)	x কে tuple রুপান্তর করার জন্য এ ফাংশনটি ব্যবহৃত হয়।
list(x)	x কে list রুপান্তর করার জন্য এ ফাংশনটি ব্যবহৃত হয়।
set()	x কে set রুপান্তর করার জন্য এ ফাংশনটি ব্যবহৃত হয়।
dict(d)	ডিকশনারি তৈরি করার জন্য এ ফাংশনটি ব্যবহৃত হয়।

ইন্টিজার সংখ্যায় কনভার্সন ঃ স্ট্রিং অথবা ফ্লোট থেকে ইন্টিজার এ কনভার্ট করার জন্য () ফাংশন ব্যবহার করা হয়।

উদাহরণ ঃ

```
# string to integer conversion
>>> int (“456”)
456
# float to integer conversion
>>> int (12.3)
12
```

দশমিক সংখ্যায় কনভার্সন ঃ স্ট্রিং অথবা ইন্টিজার থেকে ফ্লোট এ কনভার্ট করার জন্য () ফাংশন ব্যবহার করা হয়।

উদাহরণ ঃ

```
# string to float conversion
>>> float (“123.456”)
123.456
# integer to float conversion
>>> float(123)
123.0
```

স্ট্রিং এ কনভার্সন ঃ কোনো ভেরিয়েবল কে স্ট্রিং এ কনভার্ট করার জন্য () ফাংশন ব্যবহার করা হয়।

উদাহরণ ঃ

```
>>> str(1123)
‘1123’
```

Thank You

Subjcet Name: - Python Programming.

Subjcet Code : 28521

Python Operators.

Computer Science & Technology

2nd Semester (1st shift)

পাইথন অপারেটরস
(Python Operator)

ভূমিকা (Introduction)

প্রোগ্রামিং ল্যাংগুয়েজে ডাটা নিয়ে কাজ করতে হলে বিভিন্ন সময়ে বিভিন্ন অপারেটর ব্যবহারের প্রয়োজন হয়। অপারেটর ছাড়া ডাটা নিয়ে কাজ করা অসম্ভব। প্রোগ্রামিং ল্যাংগুয়েজে গাণিতিক ও যৌক্তিক কাজ নিয়ন্ত্রণ করার জন্য কতগুলো বিশেষ ক্যারেক্টার (যেমন: +, -, *, /, >, >=, <, <= ইত্যাদি) ব্যবহৃত হয়। এগুলোই অপারেটর। পাইথনে বিভিন্ন ধরনের অপারেটর রয়েছে এবং প্রত্যেকেই বিশেষ বিশেষ কর্ম সম্পাদন করে থাকে। উক্ত অধ্যায়ে আমারা পাইথনে ব্যবহৃত বিভিন্ন অপারেটর ও তাদের ব্যবহার সম্পর্কিত বিষয়াদি নিয়ে বিশদ আলোচনা করব।

পাইথন অপারেটর ও অপারেটরের শ্রেণীবিভাগ(Python Operators & their Types) :

অপারেটর (Operators) : অপারেটর হল এক্সপ্রেশনে ব্যবহৃত বিশেষ বিশেষ চিহ্ন(symbol) বা শব্দ (word) যা কম্পাইলারকে কোনো বিশেষ গাণিতিক,তুলনামুলক বা যোক্তিক কার্য সম্পাদনে নির্দেশ প্রদান করে। যেমন- a+b, a<b এ '+' এবং '<' ইত্যাদি হল অপারেটর।

অপারেন্ড(Operand): অপারেটর যেসব ডাটা,ভেরিয়েবল বা এক্সপ্রেশন নিয়ে কাজ করে তাদেরকে অপারেন্ড বলে। যেমন- a+b, a<b এ 'a' এবং 'b' ইত্যাদি হল অপারেন্ড।

অপারেটরের শ্রেণীবিভাগ(Operator types) : পাইথনে এক সেট শক্তিশালী অপারেটর আছে। পাইথন অপারেটর গুলিকে তাদের প্রকৃতি ও কার্য অনুযায়ী আট ভাগে ভাগ করা হয়েছে। যথা:

- গাণিতিক অপারেটর(Arithmetic Operator)
- তুলনামূলক অপারেটর(Comparison Operator)
- যৌক্তিক অপারেটর (Logical Operator)
- অ্যাসাইনমেন্ট অপারেটর (Assignment Operator)
- বুলিয়ান অপারেটর (Boolean Operator)
- মেম্বারশিপ অপারেটর (Membership Operator)
- আইডেন্টি অপারেটর (Identity Operator)
- বিটওয়াইজ অপারেটর (Bitwise Operator)

গাণিতিক,তুলনামুলক ও যৌক্তিক অপারেটর(Arithmetic,Comparison & Logical Operators):

গাণিতিক অপারেটর(Arithmetic Operator): বিভিন্ন ধরনের গাণিতিক কার্যাদি,সম্পন্ন করার জন্য যে সকল অপারেটর ব্যাহার করা হয় তাদেরকে গাণিতিক অপারেটর বলে। পাইথনে সাত ধরনের গাণিতিক অপারেটর ব্যবহৃত হয়।

Python Arithmetic Operators

অপারেটর	বর্ণনা	উদাহরণ
+ (যোগ)	অপারেটরের দুইপপাশের মান গুলো যোগ করার জন্য।	a+b=30
- (বিয়োগ)	অপারেটরের বাম পাশের মান থেকে ডান পাশের মান বিয়োগ করার জন্য।	a-b=-10
* (গুণ)	অপারেটরের দুইপাশের মান গুলো গুন করে।	a*b=200
/ (ভাগ)	অপারেটরের দুইপাশের মানকে ডান পাশের মান দিয়ে ভাগ করে।	b/a=2
% Modulus(ভাগশেষ)	অপারেটর	b%a=0
** Exponent(পাওয়ার)	বাম পাশের অপারেন্ড হল বেস আর ডান পাশের অপারেটর হল পাওয়ার। বেস ও পাওয়ার কে ব্যবহার করে পাওয়ার মান নির্ণয় করে।	a**b=10^2
// Floor Division(পূর্ণ সংখ্যার ভাগফল)	অপারেটরের বাম পাশের মান থেকে ডান পাশের মান দিয়ে ভাগ করে ভাগফল নির্ণয় করে এবং সাধারণত দশমিকের পরের সংখ্যা স্কিপ করে যায়।	9//=4 এবং 9.0//2.0

নিম্নে গাণিতিক অপারেটর সমুহের ব¨বহার দেখানো হলঃ

```
>>>3+2
5
 >>>10-4
6
>>>6*7
42
>>>48/3
16.00
>>>10%3
1
>>>3**2
9
>>>5+(9*3)
32
>>>-13+5
-8
>>>1.0/2.0
0.5
>>>1.0//2.00
0.0
```

তুলনামূলক অপারেটর(Comparison Operator): দুই বা ততোধিক অবজে±র মধে¨ তুলনা করার জন¨ তুলনামূলক অপারেটর (Comparison Operator)ব¨বহার করা হয়। এদেরকে রিলেশনাল অপারেটরও (Relational Operator)বলা হয় । প্রোগামে(Comparison Operator)ব¨বহার করা হলে তা থেকে বুলিয়ান ভ¨ালু রিটার্ন পাওয়া যায়।

যেমন-অপারেশন সতি¨ হলে ফলাফল হবে ‘‘True’’অপারেশন মিথ¨া হলে ফলঅফল হবে ‘False’। Comparison Operator এর ক্ষেত্রে মনে রাখতে হবে-একই সাথে বিভিন্ন ধরনের সংখ¨ার মধে¨ তুলনা করা যায় কিন্তু বিভিন্ন ধরনের অবজে±র মধে¨ তুলনা করা যায় না।

Python Comparison Operators

অপারেটর	বর্ণনা	উদাহরণ
= = (ইকুয়াল)	যদি দুই পাশের অপারেন্ড দুইটি সমান হয় তবে শর্তটা সত্যি হয়।	(a==b)
!= (নট ইকুয়াল)	যদি দুইপাশের অপারেন্ড দুইটি সমান না হয় তবে শর্তটা সত্যি হয়।	(a !=b)
> (গ্রেটারদেন)	যদি ডান পাশের অপারেন্ডের চেয়ে বাম পাশের অপারেন্ড বড় হয় তবে শর্তটা সত্যি হয়।	(a>b)
>= (গ্রেটার দ্যান অর ইকুয়াল)	যদি ডান পাশের অপারেন্ডের চেয়ে বাম পাশের অপারেন্ড ছোট হয় তবে শর্তটা সত্যি হয়।	(a >=b)
< (লেস দ্যান)	যদি ডান পাশের অপারেন্ডের চেয়ে বাম পাশের অপারেন্ড বড় বা সমান হয় তবে শর্তটা সত্যি হয়।	(a <b)
<= (লেস দ্যান অর ইকুয়াল)	যদি ডান পাশের অপারেন্ডের চেয়ে বাম পাশের অপারেন্ড ছোট বা সমান হয় তবে শর্তটা সত্যি হয়।	(a <=b)

রিলেশনাল এক্সপ্রেশন (Relation Expression): যে এক্সপ্রেশনে রিলেশনাল অপারেটর থাকে তাকে রিলেশনাল এক্সপ্রেশন বলে। রিলেশনাল এক্সপ্রেশনের মান সবসময় 0 অথবা 1 হয়।

এক্সপ্রেশন	সত্য/মিথ্যা	ফলাফল
(x<y)	True	1
(x>y)	False	0
(x+y)>=5)	True	1
!x	False	0
(x*y)>=(5+x)-2	True	1

নিম্নে রিলেশনাল অপারেটর বা তুলনামূলক অপারেটর সমূহের ব্যবহার দেখানো হল :

```
>>> x=10
>>>y=15
>>>x>y
False
>>>x<=y
True
>>> x= =y
False
>>>x!=y
True
>>>x+7>y
True
>>> x>y
False
>>>
```

যৌক্তিক অপারেটর (Logical Operator) ঃ যেসব অপারেটর যৌক্তিক কার্যাদি সম্পন্ন করার জন¨ ব¨বহার করা হয় তাহলে যৌক্তিক অপারেটর বা(Logical Operator) বলে। পাইথন প্রোগ্রামিং এ এদেরকে Logical Conjunction ও বলে। লজিক¨াল অপারেটর সাধারণত: দুই বা ততোধিক লজিক¨াল এক্সপ্রেশনকে সংযুক্ত করে এবং সত¨ বা মিথ¨া যে-কোনো একটি ফলাফল প্রদান করে।

Python Logical Operators		
অপারেটর	**বর্ণনা**	**উদাহরণ**
and (অ্যান্ড)	যদি দুইটি অপারেন্ডই সত্যিই হয়,কেবল তবেই শর্তটা সত্যি হয়।	(a and b)
Or (অর)	দুইটি অপারেন্ডের যেকোনো একটি সত্যি হলেই শর্তটা সত্যি হয়।	(a or b)
not (নট)	না-বোধক বুঝায়,সত্য অপারেন্ডের আগে not দিলে তা মিথ্যা হয়ে যায় আর মিথ্যা অপারেন্ডের আগে not দিলে তা সত্য হয়ে যায়।	(a not b)

লজিক্যাল অ্যান্ড অপারেশন(Logical and Operator):

ইনপুট		আউটপুট
A	B	Result(A and B)
0	0	0
0	1	0
1	0	0
1	1	1

লজিক্যাল অর অপারেশন(Logical or Operator):

ইনপুট		আউটপুট
A	B	Result(A or B)
0	0	0
0	1	1
1	0	1
1	1	1

লজিক্যাল নট অপারেশন(Logical Not Operation) ঃ

ইনপুট	আউটপুট
A	Result(not A)
0	1
1	0

লজিক্যাল এক্সপ্রেশন (Logical Expression): যে এক্সপ্রেশনে লজিক্যাল অপারেটর থাকে তাকে লজিক্যাল এক্সপ্রেশন বলে। লজিক্যাল এক্সপ্রেশনের মান সবসময় 0 অথবা 1 হয়।

এক্সপ্রেশন	সত্য/মিথ্যা	ফলাফল
(x>y) or (x<y)	True	1
(x>y) and (x<y)	False	0
((x>z) or (x+y)>=5)	True	1
!x	False	0

* নিম্নে লজিক্যাল অপারেটর সমূহের ব্যবহার দেখানো হল:

```
>>>2= = 2 and 3 = = 3
True
>>> 2  = = 2 and 3= = 4
False
>>> 2 = = 2 or 3 = = 4
True
>>>2 = = 5 or 3 = = 4
False
>>> 2 = = 5
False
>>> not 2 = = 5
True
>>>x=5
>>>y=6
>>>z=4
>>>(x>y)and (x<y)
False
>>>(x>y) or (x<y)
True
>>>((x>y) or (x<y))>=5
False
>>>
```

এসাইনমেন্ট, বিটওয়াইজ, মেম্বারশিপ, আইডেন্টিটি ও বুলিয়ান অপারেটর (Assignment Bitwise,Membership,Identity& Boolean Operators) :

অ্যাসাইনমেন্ট অপারেটর(Assignment Operaters): কোনো এক্সপ্রেশন বা ভেরিয়েবলের মানকে অপর কোনো কোন ভেরিয়েবলের মান হিসেবে নির্ধারন করার জন্য যে ধরনের অপারেটর(=)ব্যবহার করা হয় তাকে অ্যাসাইনমেন্ট অপারেটর বলে।যদি x একটি ভেরিয়েবল হয় আর তার মান 10 নির্ধারণ করার প্রয়োজন হয় তবে তার জন্য এস্যাইনমেন্ট অপারেটর ব্যবহৃত হবে।

যেমন :

X=10

উল্লেখ্য যে, x=10 এবং x==10 নয়। কারণ, x=10 হল অ্যাসাইনমেন্ট এক্সপ্রেশন আর x==10 হল রিলেশনাল এক্সপ্রেশন।

Python Assignment Operator

অপারেটর	বর্ণনা	উদাহরণ
=	ডান পাশের অপারেন্ডকে বাম পাশের অপারেন্ডে অ্যাসাইন করে।	c=a+b মানে a+b c
+ =Add AND	দুই পাশের অপারেন্ড যোগ করে যোগফল বাম পাশের অপারেন্ডে অ্যাসাইন করে।	c+=a মানে c=c+a
-=Subtract AND	বামপাশের অপারেন্ড থেকে ডান পাশের অপারেন্ড বিয়োগ করে বিয়োগফল বাম পাশের অপারেন্ডে অ্যাসাইন করে।	c-=a মানে c=c-a
*=Multiply AND	দুই পাশের অপারেন্ডের একাটাকে অন্যটা দিয়ে গুণ করে গুণফল বাম পাশের অপারেন্ডে অ্যাসাইন করে ।	c*=a মানে c=c*a
/= Divide AND	বাম পাশের অপারেন্ডকে ডান পাশের অপারেন্ড দ্বারা ভাগ করে ভাগফল বাম পাশের অপারেন্ডে অ্যাসাইন করে।	c/=a মানে c=c/a
%=Modulus AND	বাম পাশের অপারেন্ডকে ডান পাশের অপারেন্ড দ্বারা ভাগ করে ভাগশেষ বাম পাশের অপারেন্ডে অ্যাসাইন করে।	c%=a মানে c=c%a
**=Exponent AND	দুই পাশের অপারেন্ডের পাওয়ার মান নির্ণয় করে ঐ মান বাম পাশের অপারেন্ডে অ্যাসাইন করে।	c**=a মানে c=c
//=Floor Division	বাম পাশের অপারেন্ডকে ডান পাশের অপারেন্ড দ্বারা ভাগ করে ভাগফল (দশমিক [illegible] পর্যন্ত পূর্ণ সংখ্যা) বাম পাশের অপারেন্ডে অ্যাসাইন করে।	c//=a মানে c=c//a

নিম্নে অ¨াসাইনমেন্ট অপারেটর সমূহের ব¨বহার দেখানো হলঃ

```
>>>a=5
>>>a
5
>>>a+2
>>>a
7
>>>a-=3
>>>a
4
>>>a*=3
>>>a
12
>>>a/=2
>>>a
6.0
>>>c=2
>>>c**=a
>>>c
64.0
>>>c//=a
>>>c
10.0
>>>
```

বিটওয়াইজ অপারেটর (Bitwise Operator) ঃ বাইনারী ডাটা অর্থাৎ বিট বা বাইট নিয়ে বিভিন্ন রকমের লজিক¨াল অপারেশন সম্পন্ন করার জন¨ যেসব অপারেটর ব¨বহার করা হয় তাদেরকে বিটওয়াইজ (Bitwise Operator)বলে। পাইথনে মোট ৬ ধরনের বিটওয়াইজ অপারেটর আছে।

Python Bitwise Operator		
অপারেটর	**বর্ণনা**	**উদাহরণ**
& (বিটওয়াইজ অ্যান্ড)	মূলতঃ দুটো বিটকে গুণ করার জন্য বিটওয়াইজ অ্যান্ড ব্যবহার করা হয়।	a&b=0000 1100
| (বিটওয়াইজ অর)	দুটি বিটকে যোগ করার জন্য বিটওয়াইজ অ্যান্ড ব্যবহার করা হয়।	a|b=0011 1101
^ (বিটওয়াইজ অ্যাক্স অর)	উভয় অপারেন্ড যদি একই হয় তাহলে আউটপুট হবে 0 অন্যতায় 1 ।	a^b=0011 0001
~ (বিটওয়াইজ নেগেশন)	একটি মাত্র অপারেন্ড নিয়ে কাজ করে। অপারেন্ডের মান 0 হলে আউটপুট 1, অপারেন্ডের মান হলে আউটপুট 1।	~a=1100 0011
<< (লেফট শিফট)	ডাটা বিটকে বাম দিকে শিফট করার জন্য লেফট শিফট অপারেটর ব্যবহার করা হয়।	a<<2
>> (রাইট শিফট)	ডাটা বিটকে ডান দিকে শিফট করার জন্য রাইট শিফট অপারেটর ব্যবহার করা হয়।	a>>2

বিটওয়াইজ অ¨ান্ড অপারেশন(Bitwise Operation):

ইনপুট		আউটপুট
A	B	Result(A&B)
0	0	0
0	1	0
1	0	0
1	1	1

উদাহরণ ঃ যদি A=1,B=2 হলে (A&B) =?

A=1 ⇒ 0000 0001

B=2 ⇒ 0000 0010

A&B ⇒ 0000 0000 [1&0, 0&1=0]

বিটওয়াইজ অর অপারেশন(Bitwise or Operation)ঃ

ইনপুট		আউটপুট
A	B	Result(A|B)
0	0	0
0	1	1
1	0	1
1	1	1

উদাহরণ ঃ যদি A=1, B=2 হলে (A|B) =?

A=1 → 0000 0001

B=2 → 0000 0010

A/B → 0000 0011 [1&0=0, 0&1=0]

বিটওয়াইজ অ¨াক্স অর অপারেশন(Bitwise Xor Operation)ঃ

ইনপুট		আউটপুট
A	B	Result(A^B)
0	0	0
0	1	1
1	0	1
1	1	0

উদাহরণ ঃ যদি A=1, B=2 হলে (A^B)=?

A=1 ⇒ 0000 0001

B=2 ⇒ 0000 0010

A^B ⇒ 0000 0011 [1&0=0, 0&1=0]

বিটওয়াইজ নেগেশন অপারেশন (Bitwise Negation)ঃ

ইনপুট	আউটপুট
A	Result (~A)
0	1
1	0

বিটওয়াইজ শিফট অপারেশন (Bitwise Shift Operation) ঃ প্রোগ্রামে অনেক সময় এক বা একাধিক বিট বাম দিকে বা ডান দিকে শিফট করতে হয়। এ কাজে শিফট অপারেটর দুটো ব¨বহার করা হয়। লেফট শিফট অপারেটর ডাটা বিটকে বাম দিকে এবং রাইট শিফট অপারেটর বিটকে ডান দিকে শিফট করে।

লেফট শিফট অপারেশন(Left Shift Operation)ঃ

উদাহরণ-১ ঃ যদি A=2 হলে, A<<2=?

A=2 ⇒ 0000 0010

A<<2 ⇒ 0000 1000

এখানে A এর বাইনারী ভ¨ালুকে দু'বিট লেফট শিফট করলে A এর মান হবে (A<<2=8)

উদাহরণ-১ ঃ যদি A=2 হলে, A<<5=?

A=2  0000 0010

A<<5  0100 0000

এখানে A এর বাইনারী ভ¨ালুকে পাচঁ বিট লেফট শিফট করলে A এর মান হবে (A<<5=64)

রাইট শিফট অপারেশন (Right Shift Operation) ঃ

উদাহরণ-১ ঃ যদি A=4 হলে, A<<2=?

A=4  0000 0100

A<<2  0000 0001

এখানে A এর বাইনারী ভ¨ালুকে দুই বিট লেফট শিফট করলে A এর মান হবে (A<<2=1)

উদাহরণ-১ ঃ যদি A=64 হলে, A<<5=?

A=64  0100 0000

A<<5  0000 0010

এখানে A এর বাইনারী ভ¨ালুকে পাচঁ বিট লেফট শিফট করলে A এর মান হবে (A<<5=2)

মেম্বারশিপ অপারেটর (Membership Operator) ঃ মেম্বারশিপ অপারেটর পাইথন ল¨াংগুয়েজে ব¨বহৃত বিশেষ ধরনের অপারেটর কোনো একটি ভেরিয়েবল একটি বিশেষ লি÷,টাপল কিংবা ষ্ট্রিং এর মেম্বার কিনা তা নির্ণয়ের জন¨ যে সকল অপারেটর ব¨বহার করা হয় তাদেরকে মেম্বারশিপ অপারেটর বলে। যদি কোন একটি ক¨ারে±ার কোনো নির্দিষ্ট ষ্ট্রিং এর অন্তর্ভুক্ত হয় তবে রিটার্ণ আসবে ‘‘True”। আর যদি অন্তর্ভুক্ত না হয় তবে রিটার্ন আসবে ‘‘False”। পাইথনে দুই ধরনের মেম্বারশিপ অপারেটর বিদ¨মান। যথাঃ in এবং not in অপারেটর।

Python Membership Operator		
অপারেটর	বর্ণনা	উদাহরণ
In	যদি বাম পাশের অপারেন্ডটি ডান পাশের অপারেন্ডের ভিতর খুজেঁ পাওয়া যায়, তবেই কেবল শর্তটা সত্যি হয়।	1st1 =[`Mahi`, `Mahedi`, `Tonny`, `Munni`, `Sumaiya`, `Tahsin`, `Suhaima`] if`Mahi` in 1st1:print(Name Mahi exists in 1st`)
Not in	যদি বাম পাশের অপারেন্ডটি ডান পাশের অপারেন্ডের ভিতর খুজেঁ পাওয়া না যায়, তবেই কেবল শর্তটা সত্যি হয়।	1st 1 = [`Mahi`, `Mahedi`, `Tonny`, `Munni`, `Sumaiya`, `Tahsin`, `Suhaima`] if`Mahi` not in 1st1:print(Name Mahi does not exists in 1st`)

মেম্বারশিপ অপারেটরের কিছু উদাহরণ নিম্নরুপ ঃ

```
>>> a=`bangladesh`
>>> b=`desh`
>>> b in a
True
>>>a in b
False
>>> a not in b
True
>>>> b not in a
False>>>
```

আইডেন্টিটি অপারেটর(Identity Operator) ঃ দুইটি অবজে±র মেমোরি লোকেশনের মধে¨ তুলনা করার জন¨ অর্থাৎ দুইটি ভেরিয়েবলই একই মেমোরি লোকেশনে কিনা তা নির্ণয়ের জন¨ যেসব অপারেটর ব¨বহার করা হয় তাদেরকে আইডেন্টিটি অপারেটর বলে। পাইথনে মোট দুইটি আইডেন্টিটি অপারেটর ব¨বহৃত হয়। যথা ঃ is এবং is not অপারেটর।

Python Identity Operator		
অপারেটর	**বর্ণনা**	**উদাহরণ**
is	যদি দুই পাশের অপারেন্ড দুইটি একই অবজেক্ট বা জিনিসকে নির্দেশ করে,কেবল তবেই শর্তটা সত্যি হয়।	a=`bangladesh` b=`desh` if a is b:print(`a is b`) else:print(a is not b`)
is not	যদি দুই পাশের অপারেন্ড দুইটি একই অবজেক্ট বা জিনিসকে নির্দেশ না করে,কেবল তবেই শর্তটা সত্যি হয়।	a=`bangladesh` b=`desh` if a is not b:print(`a is not b`) else:print(a not b`)

আইডেন্টিটি অপারেটরের কিছু উদাহরণ নিম্নরুপ:

```
>>> a=`Bangladesh`
>>> b =12
>>> a is b
False
>>> a is not b
True
>>>`bangladesh`
>>> b=`desh`
>>> if a is b: print(‘a is b)
else:print(`a is not`)
a is not b
>>> if a is not  b :print (`a is not  b`)
else:print (a not b’)
a is not b
>>>
```

বুলিয়ান অপারেটর (Boolean Operators)ঃ বুলিয়ান অপারেটর হল সেসব অপারেটর যারা সত¨ অথবা মিথ¨া মান রিটার্ন করে। পাইথনে সত¨- মিথ¨া (True/False) যাচাই এর ক্ষেত্রে Boolean বা Logical Operator ব¨বহার করা হয়।

পাইথনে Boolean ভ¨ালু হল দুইটি

- True (T must be Capital Letter)
- False (F must be Capital Letter)

পাইথনে সত¨ মিত¨া যাচাই্এর ক্ষেত্রে ব¨বহার করা হয়। বুলিয়ান অপারেশনে নিম্নবর্ণিত ক্ষেত্রে মিথ¨া মান রিটার্ন করে ঃ

- None অর্থাৎ অবজে± None কোনো শব্দ থাকলে, মিথ¨া হিসাবে বিবেচিত হবে।
- False অর্থৎ অবজে± False নামে কোনো শব্দ থাকলে,মিথ¨া হিসাবে বিবেচিত হবে
- শূন¨ সংখ¨া থাকলে (যেমন- 0, 0.0, 0j) থাকলে, তা মিথ¨া হিসাবে বিবেচিত হবে।
- যে-কোনো খালি সিকুয়েন্স (যেমন-``, (), [ ]) থাকলে, তা মিথ¨া হিসাবে বিবেচিত হবে।
- যে-কোনো খালি ম¨াপিং (যেমনু { }) থাকলে, তা মিথ¨া হিসাবে বিবেচিত হবে।

এছাড়া অন¨ সকল ভ¨ালু বা সংখ¨া বা অবজে± সত¨ () হিসাবে বিবেচিত হবে

উদাহরণ ঃ

```
>>> bool(0)
 False
 >>> bool(4)
True
>>> bool(0.0)
 False
 >>> bool(2.00)
 True
 >>> bool(-15)
True
 >>> bool(0j)
 False
 >>> bool ( ( ) )
 False
 >>> bool ({} )
 False
 >>> bool ( [ ] )
 False
  >>> bool(None)
  False
 >>> bool(Fasle)
 False
  >>>
```

পাইথনে Boolean ভ¨ালু নির্ণয়ের জন¨ bool () ফাংশন ব¨াবহার করা হয়। যেমন – -0 এর Boolean ভ¨ালু জানতে চাইলে লিখব bool (0), তারপর Enter চাপলে, পাইথনে সাথে সাথে Boolean ভ¨ালু রিটার্ন করবে। পাইথনে মোট তিনটি বুলিয়ান অপারেটর আছে। যথাঃ AND,OR,NOT। AND এর বেলায় যদি সবগুলো ভেরিয়েবল এর মান সত¨ হয় তবে এক্সপ্রেশনটি সত¨ হয় অন¨থায় এক্সপ্রেশনটি মিথ¨া হয়। OR এর বেলায় যদি কমপক্ষে একটি ভেরিয়েবল এর মান সত¨ হয় তবে এক্সপ্রেশনটি সত¨ হয় অন¨থায় এক্সপ্রেশনটি মিথ¨া হয়। NOT একটি ইউনারি অপারেটর। এটি সাধারনত কোনো ভেরিয়েবল অথবা এক্সপ্রেশন এর বিপরীত ভ¨ালু রিটার্ন করে ।

Python Logical Operators

অপারেটর	বর্ণনা	উদাহরণ
and (অ্যান্ড)	যদি দুইটি অপারেন্ডই সত্যিই হয়,কেবল তবেই শর্তটা সত্যি হয়।	(a and b)
Or (অর)	দুইটি অপারেন্ডের যেকোনো একটি সত্যি হলেই শর্তটা সত্যি হয়।	(a or b)
not (নট)	না-বোধক বুঝায়,সত্য অপারেন্ডের আগে not দিলে তা মিথ্যা হয়ে যায় আর মিথ্যা অপারেন্ডের আগে not দিলে তা সত্য হয়ে যায়।	(a not b)

বুলিয়ান অপারেটরের কিছু উদাহারণ নিম্নরুপঃ

```
>>> 2==2
 True
 >>> 2 !=2
False
>>> 5>3
False
>>> 5>3
True
>>> 5<=3
 False
>> > 5>=3
True
>>> 2==2 and 3==3
True
>>> 2==2 and 3==4
False
 >>> 2==2 or 3==4
True
 >>> 2==5or 3==4
False
 >>> 2==5
 False
>>> not 2==5
```

অপারেটর প্রেসিডেন্স ও অ¨াসোসিয়েটিভিটি

(Operators Precedence & its Associativity)ঃ

অপারেটর প্রেসিডেন্স (Operators Precedence) ঃ কোনো এক্সপ্রেশনে একাধিক অপারেটর ব¨বহৃত হলে কম্পাইলার যে অনুক্রমের (order) ভিত্তিতে অপারেটরের অগ্রগণ¨তা (কোনটার কাজ আগে, কোনটার কাজ পরে) নির্ধারণ করে তাকে অপারেটরের প্রেসিডেন্স (Operators Precedence) বলে।

সাধারণ গণিতে যেমন- যোগ বা বিয়োগের আগে গুন ও ভাগ করে নিতে হয় তেমনি প্রোগ্রামিং এর বেলায় ও এই অপারেটর গুলোর একটা অগ্রাধিকার মূলক নিয়ম আছে। সেই নিয়ম মেনেই একটি ে÷টমেন্টের মধে¨ থাকা একাধিক অপারেটরের অপারেশন সম্পন্ন হবে । এটা গণিতের সরল করার নিয়মের সাথেই মিলে যায়। যেমন- প্রথমে ব্রাকেটের কাজ, তারপর প্াওয়ার/এক্সপোনেন্ট, অতঃপর,গুন ও ভাগ এবং শেষে যোগ ও বিয়োগ ইত¨াদি। যোগ, বিয়োগ, গুন, ভাগ বাদে ও প্রোগ্রামিং এ আর্ও বেশ কিছু অপারেটর আছে। এসব অপারেটরের ও অগ্রগণ¨তা ভিন্ন ভিন্ন। যেমন ঃ

```
>>> False==False or True
True
>>>False==(False or True)
False
```

উপরোক্ত উদাহরণে প্রথম ে÷টমেন্টে = = এর অগ্রাধিকার or চেয়ে বেশি। আর পরবর্তী ে÷টমেন্টে or অপারেশন অগ্রাধিকার পেয়েছে কারণ এটি একটি বন্ধনীর মধে¨ অবস্থান করছে।

উদাহরণ ঃ

```
a =20
b =10
c =15
d =0
e = (a+b)*c/d
print (“value of (a+b)*c/d is”,e)
e = (a+b) *c)/d
print (“value of (a+b)*c/d is,”e)
e = (a+b)*(c/d);
print (“value of (a+b)*c/d is,”e)
e=a+(b*c)/d;
print (“value of a +(b*c)/d is”, e)
```

আউটপুট ঃ

```
            value of (a+b)*c/d is 90
value of (a+b)*c/d is 90
value of (a+b)*c/d is 90
 value of a+(b*c)/d is 50
```

অপারেটর অ¨াসোসিয়েটিভিটি (Operators Assoitivity) ঃ কোনো এক্সপ্রেশনে যদি একাধিক অপারেটর ব¨াবহৃত হয় এবং এসকল অপারেটরের প্রেসিডেন্স যদি সমান হয় তবে এদের কাজ ডান দিক থেকে শুরু হবে না বাম দিক থেকে শুরু হবে তা যে বৈশিষ্টে¨র মধ¨মে নির্ধরিত হয় তাকে অপারেটর অ¨াসোসিয়েটিভিটি (Operetors Associativity)বলে ।

উদাহরণ-১ ঃ

The** operator associates right to left : 3**5**2 is 3 **(5**2) as it would be in math:3^5^2

উদাহারণ-২ঃ

```
>>> 2**2**2
16
>>> 2**2**2**2
65536
>>> (2**2**2)**2
256
>>> 3**3**3
7625597484987
>>> (3**3)**3
19683
>>> 3**(3**3)
7625597484987
```

নিম্নে পাইথনে ব¨বহৃত অপারেটরসমুহের প্রেসিডেন্স ও অ¨াসোসিয়োটিভিটি উল্লেখ করা হল ঃ

Operators Precedence & Associativity in Python			
ক্রমিক	অপারেটর	বর্ণনা	অ্যাসোসিয়োটিভিটি
1	( )	Parentheses	বাম থেকে ডানে
2	**	Exponent	ডান থেকে বামে
3	+x, -x, ~x	Unary plus, Unary minus, Bitwise Not	ডান থেকে বামে
4	*, /, //, %	Multiplication,Division,Floor division,Modulus	ডান থেকে বামে
5	+, -	Addition, Subtraction	ডান থেকে বামে
6	<<, >>	Bitwise Shift Operatos	ডান থেকে বামে
7	&	Bitwise AND	ডান থেকে বামে
8	^	Bitwise XOR	ডান থেকে বামে
9	|	Bitwise OR	ডান থেকে বামে
10	==, !=, >, >=,<=,is,is not,in, not in,	Comparisions, Identity, Membership Operator	ডান থেকে বামে
11	Not	Logical NOT	ডান থেকে বামে
12	and	Logical AND	ডান থেকে বামে
13	or	Logical OR	ডান থেকে বামে

ensingh Polytechnic Institute,Mymensi

Welcome To My Python Programming Class.

Fatema Zohura
Chief Instructor
Computer Science & Technology
Mymensingh Polytechnic Institute, Mymensingh.

Subjcet Name: - Python Programming.

Subjcet Code : 28521

Branching Structure.

Computer Science & Technology

2nd Semester (1st shift)

অধ্যায়

৫

Branching Structure

(ব্রাঞ্চিং স্ট্র্যাকচার)

- আমাদের আজকের পাঠ হলো , (decision making)/ সিদ্ধান্ত গ্রহণ। অর্থাৎ যেকোনো প্রোগ্রামিং ল্যাংগুয়েজ এর একটি গুরুত্বপূর্ণ বিষয় হলো (decision making) সিদ্ধান্ত গ্রহণ।

- কন্ট্রোল ফ্লো স্টেটমেন্ট প্রধানত ২ ধরনের হয়:
- 1. Branching &
- 2. Looping

- ১। **Conditional branching Statement:**

  কন্ডিশনাল কন্ট্রোল স্টেটমেন্ট নির্দিষ্ট শর্ত (condition) সাপেক্ষে এক বা একাধিক statement কে কার্যকর করে। condition সত্য হলে এক ধরণের statement(s) আর মিথ্যা হলে অন্য ধরণের statement(s) কার্যকর করে।

- **পাইথন ভাষাতে সাধারণত চার ধরণের কন্ডিশনাল কন্ট্রোল স্টেটমেন্ট পরিলক্ষিত হয়। যথাঃ**

(১) if statement

(২) if…else statement

(৩) elif statement

(৪) nested if statement

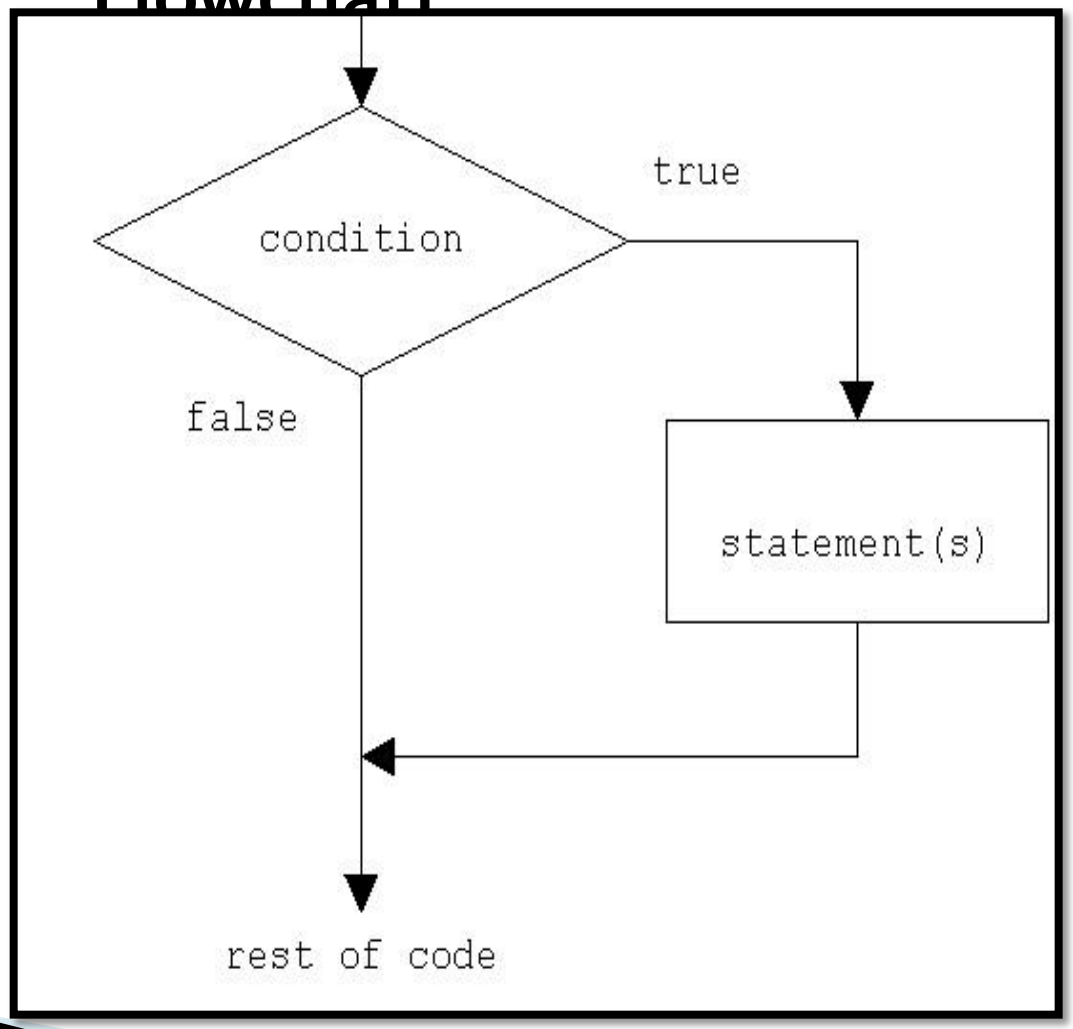
2| **Unconditional branching Statement:**যখন কোনো প্রোগ্রাম আনকন্ডিশনাল কন্ট্রোল স্টেটমেন্ট এর প্রভাবে কার্যকর হয়, তখন তাকে Unconditional branching Statement বলে।

5.2. if স্টেটমেন্ট

Syntax of if statement

```
if (condition):
    statements
```

Flowchart:



Example:

```
if (hour < 18) :
  greeting
= "Good day"
```

Example-2:

```
>>>  a = a
>>>  a
10
>>>  if a < 20:
...    print(“ a is less than 20.”)
  File “ <stdin>” , line 2
                    print(“ a is less than 20.”)
IndentationErrore : expected an indented
black
```

উপরোক্ত প্রোগ্রামটি রান করলে পাইথন একটি এরর থ্রো করেছে। এটা হচ্ছে indentation error। পাইথনে ইনডেন্টেশন বাধ্যতামুলক। তাই যখনই কোথাও ইনডেন্টেশন না করা হয়

তখনই পাইথন এইরকম এরর থ্রো করে। সঠিক ইনডেন্টেশন মাধ্যমে এই

উদাহরণ-৩ঃ

```
>>> a=30
>>> b=10
>>> if a>b:
        print("a is greater than b")

Output  :  a is greater than b
>>>
```

উদাহরণ-৪ঃ

```
>>> a=30
>>> b=10
>>> if a>b:
      print("a is greater than b")

>>>
```

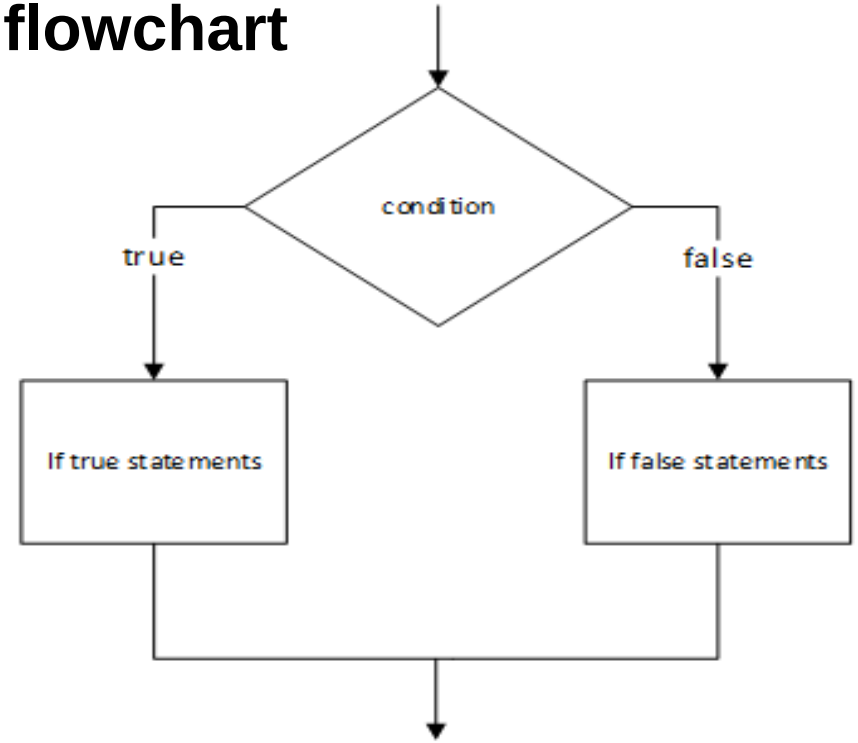
উদাহরণ-৩ ও উদাহরণ-৪ এর আউটপুট লক্ষ করলে দেখা যাবে যে, উদাহরণ-৩ এর ক্ষেত্রে কন্ডিশন সত্য হওয়ায় প্রিন্ট করছে a is greater than b কিন্তু উদাহরণ-৪ এর ক্ষেত্রে কন্ডিশন সত্য

if else স্টেটমেন্ট (if else statement) : প্রোগ্রামে 'অন্যথায়' অর্থে if স্টেটমে ন্টের সাথে else স্টেটমেন্ট ব্যবহৃত হয়।

শর্তসাপেক্ষে কোনো স্টেটমেন্ট সম্পাদনের জন্য এটি ব্যবহৃত হয়। পাইথনে if স্টেটমেন্ট ব্যবহার করে নির্দিষ্ট একটি কন্ডিশনের উপর ভিত্তি করে কিছু স্টেটমেন্ট বা কোড ব্লককে রান করানো যায়। যদি কন্ডিশনটি বা এক্সপ্রেশনটি সত্য হয় তাহলে এর আওতাভুক্ত স্টেটমেন্ট বা স্টেটমেন্টসমূহ রান হয়, আর মিথ্যা হলে else স্টেটমেন্টের পরবর্তী স্টেটমেন্ট বা স্টেটমেন্টসমূহ রান হয়।

```
if else statement Syntax:
        if (condition):
                statement1
else:
                statement2
```

flowchart

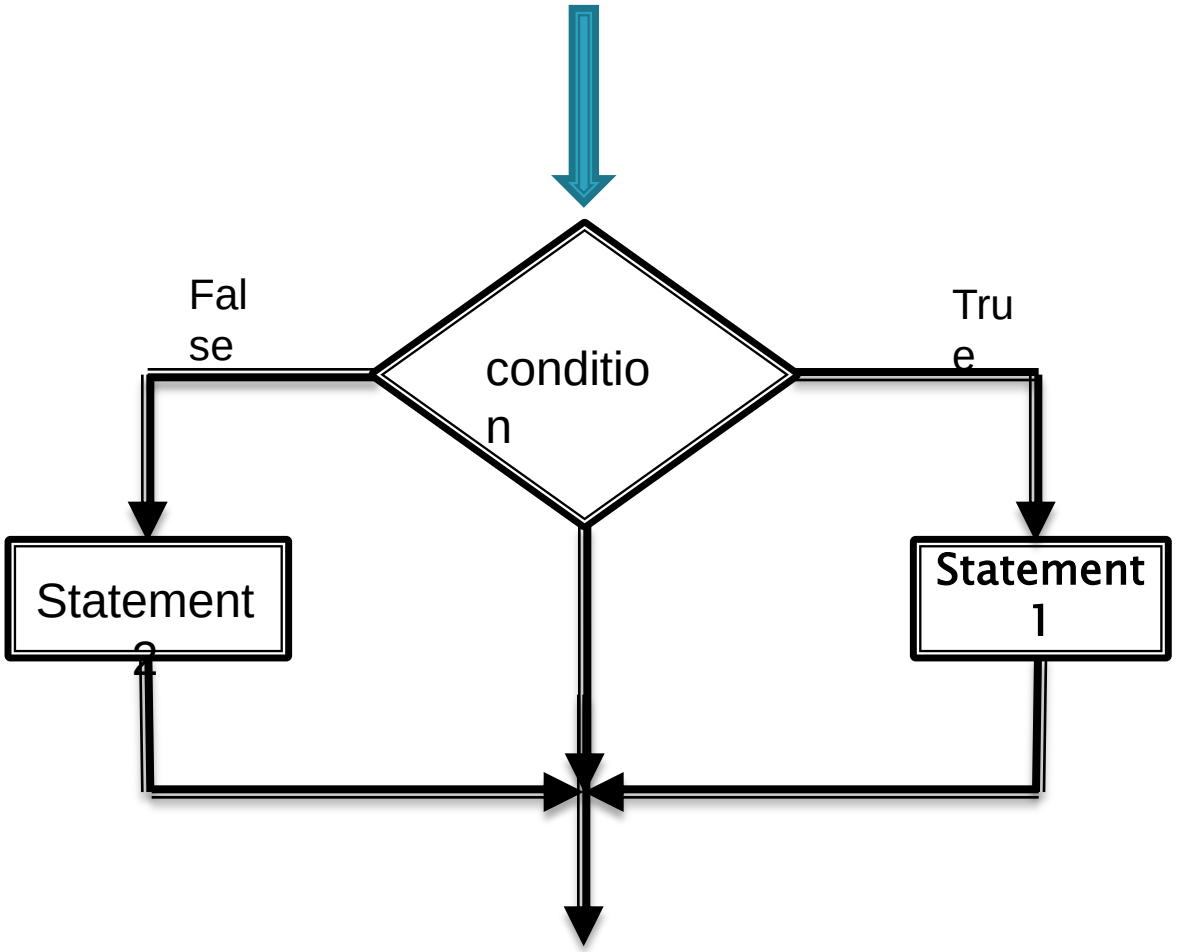


Example:

```
if (hour < 18) :
  greeting = "Good day";
else :
  greeting = "Good evening"
```

- সিনট্যাক্স (Syntax)ঃ

```
if condition:
    statements
else:
    statements
```



চিত্রঃ if else স্টেটমেন্ট এর ফ্লোচার্ট

- উদাহরণ-১

```
>>> a=10
>>> b=20
>>> if  a>b:
            print("a is larger than b")
      else:
            print("b is larger than a")

Output :  b is larger than a
>>>
```

উদাহরণ–২

```
>>> n=int(input("Enter your Number=  ")

Enter your Number =  21

>>> if  n%2=0:
            print("Even Number")
      else:
            print("Odd Number")

Output : Odd Number
 >>>
```

উদাহরণ–৩

```
>>> number=int(input("Enter your Number:")
Enter the Number:   - 3

 >>> if   Number>=0:
            print("This is an positive Number or Zero")
       else:
            print("This is a Negative Number")
```

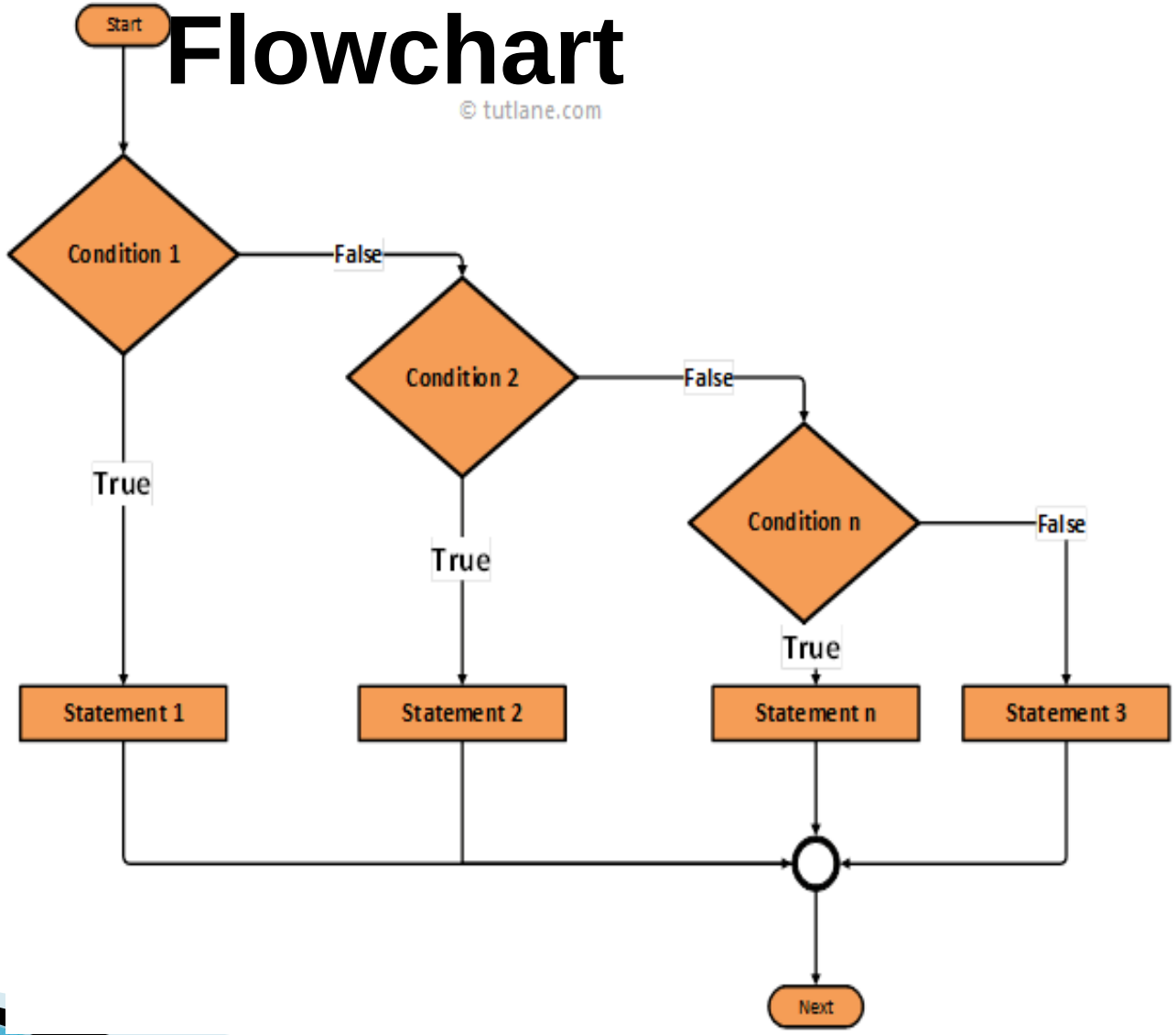
Output : This is a Negative Number

```
 >>>
```

Elif statement syntax:

```
if (condition1):
        statement1
elif (condition2):
        statement2
elif (condition3):
        statement3
else:
        statement
```

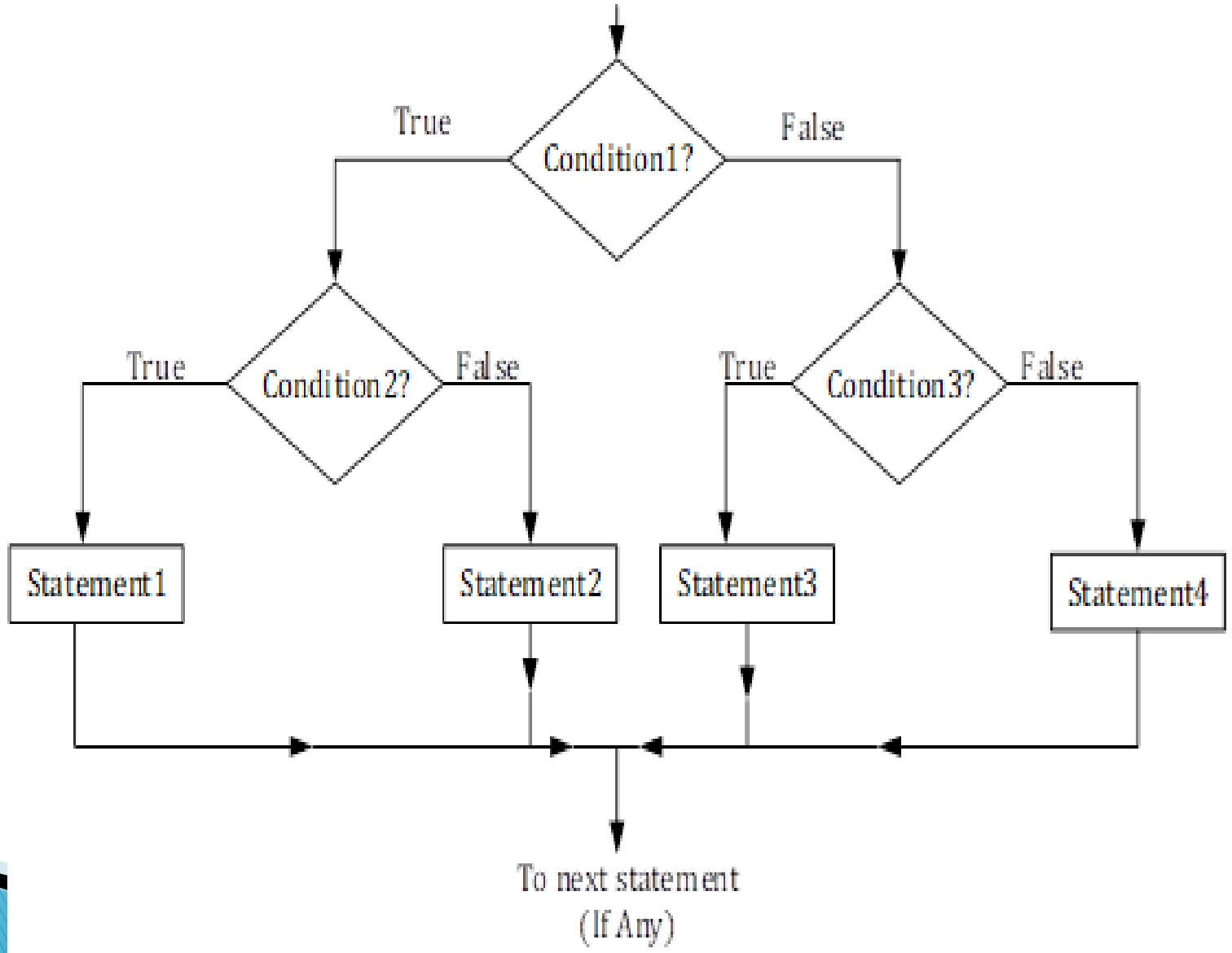
Elif statement Flowchart



- **Nested if else syntax:**

- **if** (Condition1):
  if(Condition2):
  Statement1
  else :
  Statement2
  else:
  if(Condition3):
  Statement3
  else:
  Statement4

Nested if else Flowchart




Thank You
Everyone

ensingh Polytechnic Institute,Mymensi

Welcome To
My Python Programming Class.

Fatema Zohura
Chief Instructor
Computer Science & Technology
Mymensingh Polytechnic Institute, Mymensingh.

Subjcet Name: - Python Programming.

Subjcet Code : 28521

Looping Structure.

Computer Science & Technology

2nd Semester (1st shift)

ভূমিকাঃ

যে কোনো প্রোগ্রামিং ল্যাঙ্গুয়েজ এর একটি গুরুত্তপূর্ণ বিষয় হল লুপিং। লুপ মানে একই কাজ বার বার করা। লুপ ব্যবহার করে কাজ সহজ হয়। কন্ডিশনাল লজিকের একটি অন্যতম উদাহরণ হল লুপ। এখানে আমরা পাইথনের বিভিন্ন লুপিং সম্পর্কিত স্টেটমেন্ট ও তাদের ব্যবহার নিয়ে আলোচনা করব।

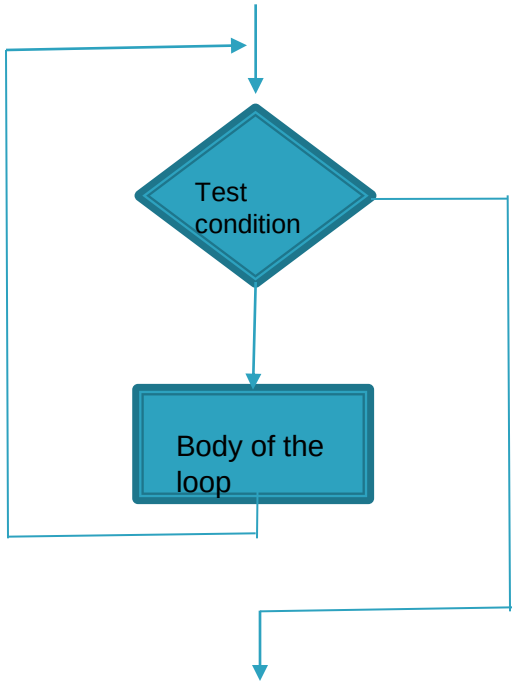
কন্ডিশনাল ও আনকন্ডিশনাল ফ্লোঃ

লুপ দুটি প্রধান অংশ নিয়ে গঠিত। যথাঃ

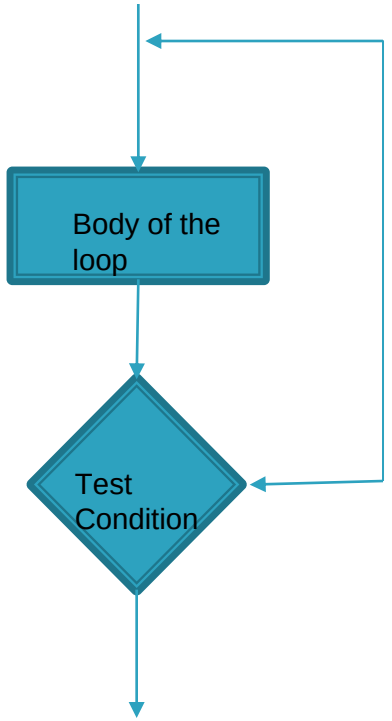
- লুপের মূল অংশ ।
- কন্ট্রোল স্টেটমেন্ট।

লুপ স্ট্রাকচার দুই ধরনেরঃ

>> এন্ট্রি কন্ট্রোল।
>> এক্সিট কন্ট্রোল ।



(a) Entry Control



(b) Exit Control

লুপিং দুই প্রকারঃ

- কন্ডিশনাল লপিং ফ্লো।
- আন কন্ডিশনাল লপিং ফ্লো।

দুই ধরনের কন্ডিশনাল লুপিং ফ্লো স্টেটমেন্ট আছে। যথাঃ

- for স্টেটমেন্ট।
- while স্টেটমেন্ট।

for লুপ স্টেটমেন্টঃ

"প্রোগ্রাম এ এক বা একাধিক স্টেটমেন্ট একটি নিদিষ সংখ্যকবার এক্সিকিউট করার জন্য for লুপ ব্যবহার হয়।"

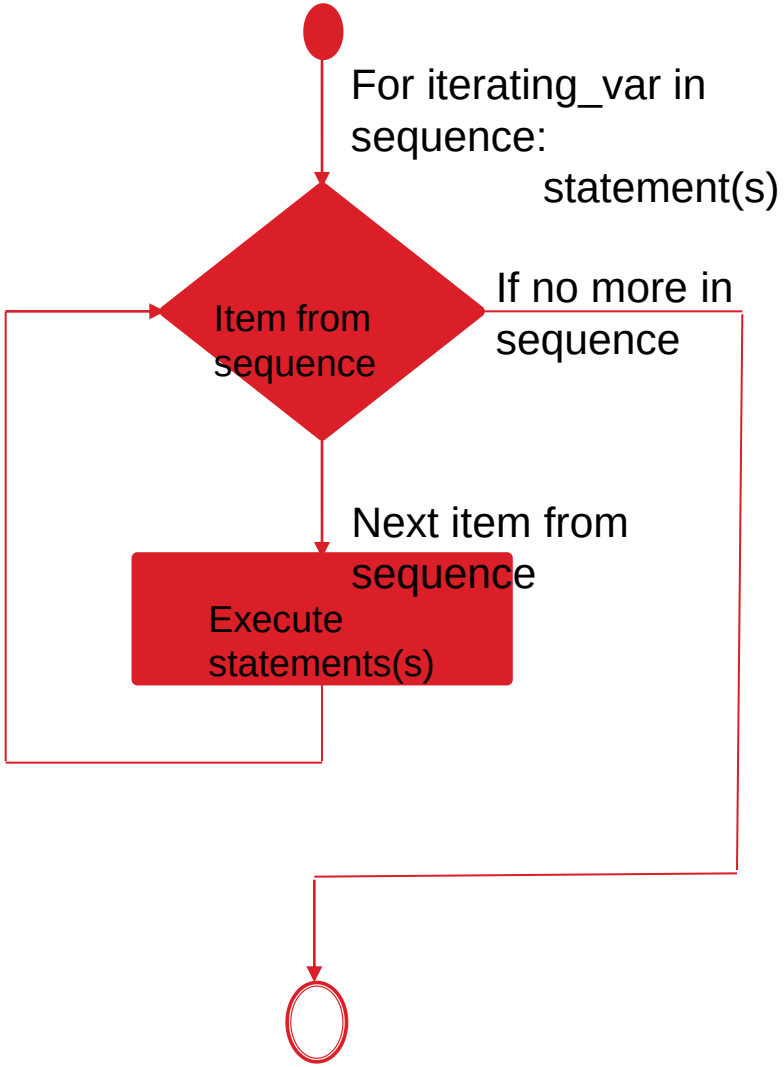
ইনপুটঃ

```
For x in range(3)
        print('Hell
o')
print('Good Bye')
```

আউটপুটঃ

```
Hello
Hello
Hello
Good Bye
```

ফ্লোচার্টঃ



ডাটা স্ট্রাকচারের সকল ধরণের ডাটার মধ্যে লুপ চালানোর জন্য for লুপ ব্যবহার করা হয়।

লিস্টের সাথে for লুপ এর ব্যবহারঃ

পাইথনের কোনো লিস্টকে for লুপ ব্যবহার করে ইটারেট করা হয়।

উদাহরনঃ

```
friends=['Tahsin','Mahi','Mahdee']
For friend in friends:
        print('Happy New Year: ',friend)
        print('Thanks a Lot.\n')
```

আউটপুটঃ

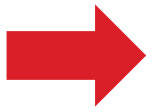
```
Happy New Year:
Tahsin
Thanks a Lot.

Happy New Year:
Mahi
Thanks a Lot.

Happy New Year:
Mahdee
Thanks a Lot.
```

স্ট্রিং এর সাথে for লুপ এর ব্যবহারঃ

পাইথনের কোনো স্ট্রিংকে for লুপ ব্যবহার করে ইটারেট করা হয়।

ইনপুটঃ

```
Name='Mahi'
For letter in Name:
        print(letter)
```



আউটপুটঃ

```
M
a
h
i
```

For লুপ এর সাথে range() function এর ব্যবহারঃ

for লুপ এর সাথে range() function কে বিভিন্ন ভাবে ব্যবহার করা হয়। range() function এ দুই সেট প্যারামিটার আছে। যথাঃ

```
>> range(start)
>> range(stop)
>> range(step)
```

এখানেঃ

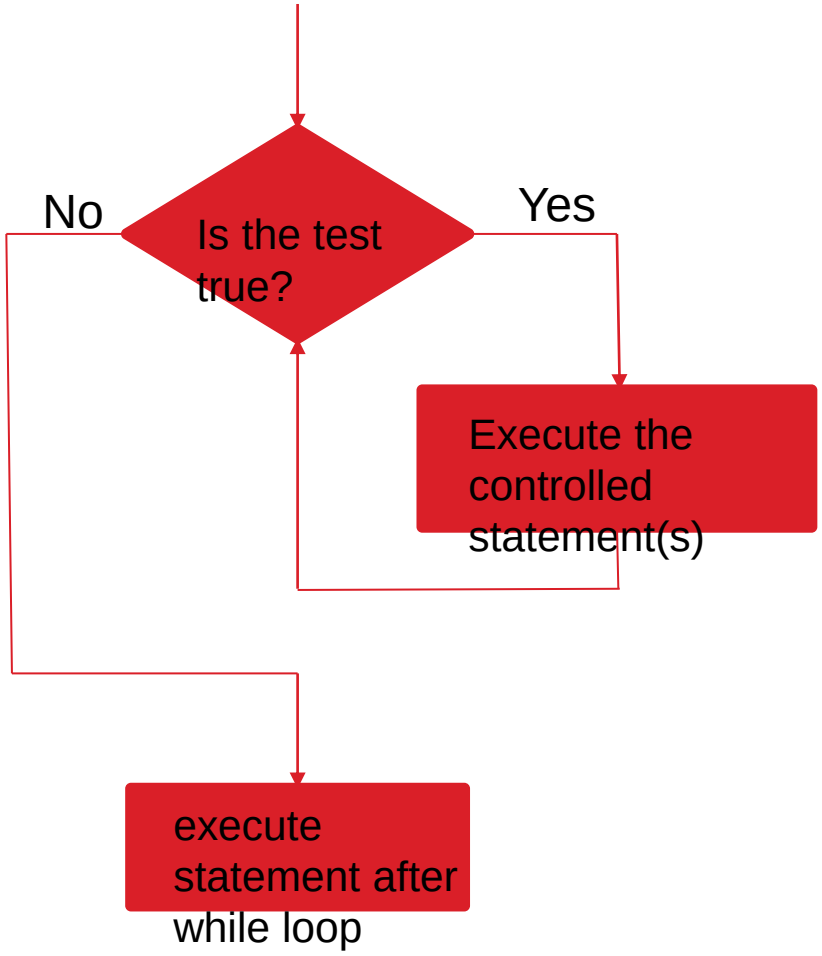
>> Start হল সিকোয়েন্সের ইনিসিয়াল ভ্যালু।

>> Stop হল সর্বশেষ মানের পূর্বের মান এবং

>> Step হল কাউন্টার ভেরিয়েবলের ইনক্রিমেন্ট বা ডিক্রিমেন্ট ভ্যালুকে বুঝায়।

While লুপ স্টেটমেন্টঃ

ফ্লোচার্টঃ



চিত্রঃ while loop

While loop এর সাধারণ গঠন হচ্ছেঃ

While Condition:
    statements

while লুপের কন্ডিশনটি যতক্ষণ পযন্ত সত্য হবে, ততক্ষন পযন্ত while লুপটি চলবে। কন্ডিশন মিথ্যা হলেই প্রোগ্রামের এক্সিকিউশন শেষ হয়ে যাবে।

ইনপুটঃ

```
a = 1
while a < 10:
        print(a)
        a=a+1
print ('This is the End')
```

আউটপুটঃ

1
2
3
4
5
6
7
8
9
This is the End

While লুপের সাথে if এর ব্যবহারঃ

ইনপুটঃ

```
count = 0
sum = 0.0
number = 1
print (“Enter 0 to exit the loop.”)
while number != 0:
        number = float(input(“Enter a
number: “))
        if number != 0:
                count = count + 1
                sum = sum +
number
        if number == 0:
                print(“The average
was: ”, sum/count)
```

আউটপুটঃ

```
Enter 0 to exit the loop.
Enter a number: 10
Enter a number: 5
Enter a number: 7
Enter a number: 14
Enter a number: 0
The average was: 9.0
```

while লুপের সাথে if else ও elif এর ব্যবহারঃ

ইনপুটঃ

```
number = 7
guess = 0
count = 0
print(“Guess the number.“)
while guess != number:
        guess = int(input(“Enter the
Number: ”))
        count = count + 1
        if guess == number
                print (“You are
right.”)
        elif guess < number”
                print (“It’s Bigger.”)
        elif guess > number:
                print (“It’s not so
big.”)
        if count > 3:
                print (“You are
banned. Try again Later…”)
```

আউটপুটঃ

```
Guess the number.
Enter the Number: 10
It’s not so big.
Good Job
Enter the Number: 15
It’s not so big.
Good Job
Enter the Number: 5
It’s Bigger.
Good Job
```

Infinite, Nested লুপ স্টেটমেন্টঃ

Infinite লুপ স্টেটমেন্টঃ

পাইথন এ দুই ধরণের ইনফিনিটি লুপ করা যায়। যথাঃ

>> infinite for লুপ স্টেটমেন্ট
>> infinite while লুপস্টেটমেন্ট

Nested লুপ স্টেটমেন্টঃ

পাইথন এ দুই ধরণের নাস্টেড লুপ করা যায়। যথাঃ

>> nested for লুপ স্টেটমেন্ট
>> nested while লুপ স্টেটমেন্ট

Break, Continue ও Pass স্টেটমেন্টঃ

লুপের এক্সিকিউশন কন্ট্রোল করার জন্য লুপ কন্ট্রোল স্টেটমেন্ট ব্যবহার করা হয়। পাইথনে তিন ধরণের লুপ কন্ট্রোল স্টেটমেন্ট ব্যবহার করা হয়। যথাঃ

>>Break স্টেটমেন্ট
>>Continue স্টেটমেন্ট
>>Pass স্টেটমেন্ট

Break স্টেটমেন্ট:

ইনপুটঃ

```
For letter in ‘string’:
        if letter ==
“n”:

        break
        print(letter
)
Print(“The End”)
```

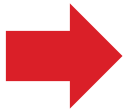
আউটপুটঃ

```
s
t
r
i
The End
```

continue স্টেটমেন্টঃ

ইনপুটঃ

```
for letter in "Rahat & Probhas":
    if letter == "&":
        continue
    print(letter)
print("The End")
```



আউটপুটঃ

```
R
a
h
a
t

P
r
o
b
h
a
s
The End
```

pass স্টেটমেন্টঃ

ইনপুটঃ

```
For letter in “Python”:
        pass
Print(letter)
```

আউটপুটঃ

```
last letter:
n
```

```
for number in
range(10):
    pass
print("last number:
",number)
```

```
last number:  9
```

লুপিং স্টেটমেন্টস ব্যবহার করে প্রোগ্রামঃ

ইনপুটঃ

```
number=int(input("Please, input the
number: "))
count=1
while count<=10:
    print(number,
'x',count,'=',number*count)
    count += 1
```

আউটপুটঃ

```
Please, input the
number: 16
16 x 1 = 16
16 x 2 = 32
16 x 3 = 48
16 x 4 = 64
16 x 5 = 80
16 x 6 = 96
16 x 7 = 112
16 x 8 = 128
16 x 9 = 144
16 x 10 = 160
```

লুপিং স্টেটমেন্টস ব্যবহার করে প্রোগ্রামঃ

ইনপুটঃ

```
num= int(input("Enter a
Number: "))
if num > 1:
    for i in range(2,num):
        if(num % i) == 0:
            print(num, "is not a
prime number")
            break
    else:
        print(num,"is a prime
number")

else:
    print(num,"is not a prime
number")
```

আউটপুটঃ

```
Enter a Number: 2
2 is a prime number

Enter a Number: 10
10 is not a prime
number
```

Thank You

লিস্ট(list)

ভূমিকা (Introduction):

পাইথনে ছয় ধরনের বিল্ট ইন টাইপ ডাটা স্ট্রাকচার আছে।সেগুলো হচ্ছে numeric,sequence,mapping,class,instance.এবং exception. এর মধ্যে বহুল ব্যবহৃত বেসিক ডাটা স্ট্রাকচার হচ্ছে sequence ।প্রথম ইনডেক্স শূন্য, তারপর ১ এবং এরপর ক্রমানুযায়ী মান বাড়তে থাকে । পাইথনে আবার তিন ধরনের বেসিক sequence টাইপ ডাটা কচাস্ট্রার আছে।যেমন:list, tuple, এবং xrange object । পাইথনে লিস্ট হলো একটা স্মার্ট ডাটা টাইপ।কারণঃ প্রোগ্রামিং এ লিস্টের ব্যবহারিক প্রয়োগ অনেক বেশি।সি, সি++ বা জাভা ল্যাংগুয়েজে যাকে অ্যারে বলা হয় পাইথনে তাকেই লিস্ট বলে।তবে অ্যারের চেয়ে লিস্টের প্রায়োগিক সক্ষমতা বেশি।

লিস্ট ও লিস্টের উপাদান(Define List and its Elements Type):

- লিস্ট (list): list হচ্ছে পাইথনের সবচেয়ে বৈচিএপূণ ডাটা টাইপ যা স্কয়ার ব্র্যাকেট[ ]এর ভেতর কমার

- সাহায্যে উপাদান সহ প্রকাশ করা হয়।অন্যান্য ল্যাংগুয়েজে যাকে অ্যারে বলা হয় পাইথনে তাকে লিস্ট

- বলে।তবে listএর উপাদানগুলো অ্যারের মত একই রকম ডাটা টাইপ হবার প্রয়োজন নেই। যেমন:
- List1=[physics,chemistry, 1997,2000];
- List2=[1,2,3,4,5];
- List3=[“a”,”b”,”c”,”d”];

- লিস্টের এলিমেন্টগুলো ইনডেক্স অনুযায়ী (০,১,২,৩……)সাজানো থাকে। ইনডেক্সের মান সবসময় 0 থেকে শুরু হয়। লিস্টকে sliced,concatenated ইত্যাদি করা যায়।

- অনেক টাইপের ডাটা রাখা যেতেলিস্টের এলিমেন্ট টাইপ:পাইথনে লিস্টের মধ্যে পারে।যেমনঃ
- ইন্টিজার টাইপ ডাটা
- ফলোটিং টাইপ ডাটা
- স্ট্রিং
- লিস্ট
- টাপল

লিস্ট ও ইন্টিজার টাইপ ডাটা(list & Integer Type Data) : শুধুমাএ ইন্টিজার সংখ্যা ব্যবহার করে যে লিস্ট গঠিত হয় তাকে ইন্টিজার টাইপের লিস্ট বলে।

উদাহরণঃ

```
>>> Numbers=[5,7,4,6,3,18,1,9]
>>> print(Numbers)
>>>  [5,7,4,6,3,18,1,9]
 >>>  print (Numbers[5])
 >>>  18
 >>>  print (Numbers[3])
  >>> 6
  >>> Numbers.sort()
  >>> print(Numbers)
  >>> [1,3,4,5,6,7,9,18]
  >>>
```

লিস্ট ও ফ্লোটিং পয়েন্ট টাইপ ডাটা (list& Floating Point Type Data)

: শুধুমাত্র ফ্লোটিং পয়েন্ট টাইপ সংখ্যা ব্যবহার করেও লিস্ট তৈরি করা যায়।

- যেমনঃ
- >>> Numbers=[5.4,3.7,10.4,25.6,33.7,7.18,172.34,97.99]
- >>> print (Numbers)
- [5.4,3.7,10.4,25.6,33.7,97.99]
- >>> print (Numbers[3])
- 25.6
- >>> Numbers.sort()
- >>> print (Numbers)
- [3.7,5.4,7.18,10.4,25.6,33.7,97.99,172.34]
- >>>

লিস্ট ও স্ট্রিং(list&string):একাধিক স্ট্রিং এর সমষ্টি নিয়ে পাইথনে লিস্ট তৈরি করা যায়।যেমন:

- যেমনঃ>>>Names=[“Fatima”,”Rahima”,Sumaiya”,”Tahsin”,”Tanveer”]
- >>>print(Names[1],Names[0],Names[3])
- Rahima Fatima Tahsin
- >>>
- লিস্টের মধ্যে লিস্ট (List within List) : লিস্টের মধ্যে ও লিস্ট তৈরি করা যায়।
- যেমন:
- list1=[‘physics’, ‘chemistry’, 1997,2000];
- list2=[1,2,3,4,5];
- list3=[“a”,”b”,”c”,”d”];

মিস্কড ডাটা টাইপ ও লিস্ট(List &Mixed type data):একাধিক টাইপ ও লিস্ট তৈরি করা যায়।

- উদাহরণ:
- >>>MyList=[“Tanveer”,2017,99.5]
- >>>print(MyList)
- [“Tanveer;2017,99.5]
- >>>print(MyList[2]
- 99.5
- লিস্টের উপাদান অ্যাক্সেসিং ঃলিস্টের উপাদান সমূহকে অ্যাক্সেসং করার জন্য অনেক পদ্ধতি প্রচললিত আছে।পদ্ধতি সমূহ হচ্ছে ঃ
- >Indexing
- >Negative Indexing
- >Slicing

ইনডেক্সিং (Indexing) পদ্ধতি: যে পদ্ধতিতে লিস্টের কোনো আইটেমকে অ্যাক্সেস করার জন্য ইনডেক্স নম্বর ব্যবহার করা হয় তাকে ইনডেক্সিং পদ্ধতি বলে।

- লিস্টের ইনডেক্স নম্বর শুরু হয় ০ থেকে।
- উদাহরণ: my-list=[‘p’,’r’,’o’,’b’,’e’]
- print(my-list[0]
- print(my-list[2]
- print(my-list[4]
- n-list=[“Happy”,[2,0,1,5]]
- print(n-list[0][1])
- print(n-list[1][3])
- আউটপুট: p
- o
- e
- a
- 5
- >>>

উদাহরণ2

- B=[‘Mahi;’Mahbub;’Mahfuz;’Samsul;96002,88.87]
- Print(b[0])

  Print(b[1])

  Print(b[1:5])

  Print(b[:5])

  0Print(b[2:])

আউটপুট: Mahi

Mahbub

[‘Mahbub;’Mahfuz;’Samsul;96002]

[‘Mahi;’Mahbub;’Mahfuz;’Samsul;96002]

[’Mahfuz;’Samsul;96002,88.87]

>>>

1.

Append():

Syntax:

List.append(item)

Example:

```
List =[‘Mahi’, ‘Mahdee’,’Suhaima;]
List.appent(‘Tahsin’)
Prist(Update list is :’,List)
List2=[Tonny’,’Munny’,’Smaiya’]
List1.append(List2)
Print(‘Update list is :’,List2)
```

Output:

Updated list is: [‘Mahi’,’Mahdee’,’Suhaima ,’Tahsin]

Updated list is: [‘Tonny,’Munny,’Sumaiya’]

Extend():

List1.extend(listr2)

Syntax:

:

List1.extend(list2)

Example:

```
List1=[‘Mahi,’Mahdee,’Suhaima’]
List2=[‘Tonny,’Munny,’Sumaiya’]
List1.extend(List2)
Print(List1)
```

Output

[Mahi,'Mahdee,'Suhaima,'Tonny,'Munny,'Sumaiya]

2. Insert():

Syntax:

xt:

list.insert(index,element)

Example:

```
vowel=['a,'e',i','u']
#inserting element to list at 4th position
Vowel.insert(3,'0)
Print ('Updated List :',vowel)
List1=[{1,2},[5,6,7]]
X=(3,4)
#inserting x to the list
List1.insert(1,x)
Print
('Updated List :',list1))
```

Output: Updated List:['a,'e,'i,'o,'u]

: Updated List:[{1,2},(3,4),[5,6,7]]

3.

Remove():

Syntax:

List.remove(element)

Example: vowel=[‘a,’e’,i’,’o’,’u’]

Vowel.remove(‘o’)

Print(Updated List:’,vowel)

Animal=[‘cat,’dog,’dog,’pig,’dog’]

Animal.remove(‘dog’)

Print(‘Updated animal list:,’ animal)

Output

Updated List [‘a’,e’,i’,u’]

Updated animal list:[‘cat’,’dog’,’pig’,’dog’]

Pop():

Syntax:

List.pop(index)

Example:

Language=[‘Python’,’Java’,’C++’”,’C’]

Return_value=languge.pop(3)

Print(‘Return Value:’,return_value)

Print(Updated List:’,language)

```
List1=[{1,2},[5,6,7]]
X=(3,4)
#inserting x to the list
List1.insert(1,x)
Print (‘Updated List :’,list1))
```

Output: Updated List:[‘a,’e,’i,’o,’u]

: Updated List:[{1,2},(3,4),[5,6,7]]

4.

Index():

Syntax::

List.index(element)

Example:

```
vowels=['a','e','i','o','l','u']
#element 'e'is searched
Index = vowels.index('e')
#index is printed
Print('The index  of e:',index)
#element 'l' is  searched
Index =  vowels.index('l')
#only the first  index of  the element is printed
Print('The index of i:', index
```

Output:

```
The index  of e:1
The index of i: 2
```

5.

Output:

Return Value: Cobol

Updated List: [Python,’Java,’C++,’C]

Clear():

Syntax:

List.clear()

Example:

List=[‘Python’,’Java,’C++,’C’]

List.clear()

Print(Updated List:’,list)

Output:

Updated List:[]

6.

Count():

Syntax:

List.count(element)

Example:

```
Vowels=[‘a’,’e’,’i’,’o’,’I’,u’]
#count element ‘I’
Count = vowels.count(‘I’)
#print count
Print (‘The count of i is:’,count)
#count  element ‘p’
Count = vowels.count(‘p’)
#print count
Print(‘The  count of p is:’,count)
```

Output:

The count of i is:2

The count of p is:0

7.

Sort():

Syntax:

```
List.sort()
```

Example:

```
#vowels  list
vowels = [‘e’,’a’,’u’,’o’,’i’]
#sort the vowels
Vowels.sort()
# print vowels
Print(‘Sorted list:’, vowel s)
```

Output:

```
Sorted list:[‘a’,’e’,’I’,’o’,’u’]
```

8.

Reverse()

Syntax:

List.reverse()

Example:

```
Os= [‘Windows’,’macOS’,’Linux’]
Print(‘Original List:’,os)
#List Reverse
Os.reverse()
#updated list
Print(‘Updated List:’, os)
Reversed_list = os[::-1]
Print(‘Updated List:’,reversed_ list)
For o in reversed(os):
    Print(o)
```

9.

Output:

Original List:['Windows','macOS','Linux']

Updated List:['Linux','macOS','Windows']

Updated List:['Windows','macOS','Linux']

Windows

macOS

Linux

The End