

```
public class Main{
    public static void main(String args[]){
        Test t;
        System.out.println(t.i);
    }
}
```

- (A) 0 (B) A garbage Value
 (C) The program will cause an runtime exception because the variable i was not initialized
 (D) The program will cause an compile error because the object t was not initialized.

e. Interfaces in java are meant to be _____. [জাতীয় ইন্টারফেস ক্লাসে বোকাসে হবে _____.] [Combine 4 bank - AP - 2020]

- (A) Extended (B) Implemented
 (C) Overridden (D) Used by creating object

f. Which of the following statements is/are true about inheritance in java [জাতীয় ইনহেরিটেন্স সম্বন্ধে কোন চেটনাটি সত্য?] [Combine 4 bank - AP - 2020]

1. Private methods are final
 2. Protected methods are final
 3. Private methods cannot be overridden
 4. Protected members of a class are accessible by inherited classes of another package.
- (A) 1,3,4 (B) 1,3 (C) 2,3,4 (D) 2,4

g. Which of the following statements is not true for java language[জাতীয় কোন চেটনাটি সত্য নহ?] [Combine 4 bank - AP - 2020]

- (A) The number 1 can be used instead of the keyword True
 (B) Trying to store a fraction value in an int datatype causes compiler error
 (C) Static members of a class can be accessed without creating objects of that class
 (D) If not specified otherwise, the initial value of an integer variable is 0

h. Java uses a keyword ____ to preface a block of code that is likely to cause an error condition and throw an exception. [একটি দ্রুবের কোভ প্রিফেস করার জন্য জাতীয় একটি কীওয়ার্ড ____ ব্যবহার করে যা সম্ভবত cause এবং error condition এবং একটি exception throw করে।] [BSC 2 Combined Bank - Senior Officer (IT)-2020]

- a) throw b) catch
 c) finally d) try

i. Which is not the feature of JAVA OOP?[জেন ফিচার ক্লাসে নাই] [Janata Bank-ANE-2020]
 a) Multiple Inheritance
 b) Multi-level inheritance meaning
 c) Compile time Polymorphism
 d) Runtime polymorphism

j. Which information is not correct for any constructor of a java class?[যে কোন কল্পনাকৃতিরের জন্য জাতীয় ক্লাসে কোন ইনফরমেশন তি সত্য নহ?] [Sonali Bank Ltd. Assistant Database Administrator-2020]

- (a) Constructor is not inherited
 (b) Constructor has no return type
 (c) Constructor can be final
 (d) Constructor can be overloaded

k. Which language was used to build android operating system?[জাতীয়ত অ্যারোড্রইড সিস্টেম তৈরিতে কোন শাস্ত্রজ্ঞ ব্যবহার হব?] [Sonali Bank Ltd. Assistant Database Administrator-2020]

- (a) Java (b) Python
 (c) Kotlin (d) Android is not an operating system

উত্তরসংক্ষেপ:

১	২	৩	৪	৫	৬	৭	৮	৯	১০	১১
c	c	a	D	B	A	B	d	a	c	a

Python Programming

[Syllabus: Object Oriented Programming Basics; Class & Object, Properties Of Object Oriented Programming.]

পাইথন প্রোগ্রামিং এব় স্ক্রিপ্টিক শর্তা (Basic Idea of Python Programming)

Python Programming language (পাইথন প্রোগ্রামিং ভাষা): Python একটি General-Purpose Interpreted, সাধারণ, সহজবোধ, উদ্দেশ্য-কেন্দ্রিক এব় উচ্চমানের প্রোগ্রামিং শ্যাখায়ে।

Guido van Rossum কে Python Programming language এর Inventor (জনক) বলা হয়।

পাইথন এর বিভিন্ন ভাস্তু এব় প্রকাশকাল:

ভাস্তু	প্রকাশকাল
পাইথন ১.০ (অসম স্টার্টআপ প্রকাশনী)	১৯৯৪ এর জানুয়ারি
পাইথন ৩.৯ (শেষ অপেক্ষিত ভাস্তু)	২০২০ সালের ৪-ই অক্টোবর

অর্থ: Python Programming language (পাইথন প্রোগ্রামিং ভাষা) এর বৈশিষ্ট্য তলো লিখুন।

উত্তর: Python Programming language (পাইথন প্রোগ্রামিং ভাষা) এর বৈশিষ্ট্য তলো:

1. Python সহজবোধ: এখানে স্টা-ওয়ার্ড এর ব্যবহার তুলনামূলক কম। Python এর শব্দবিন্যাস এব় ভাষা-গঠনও অপেক্ষাকৃত সহজ।
2. Python সহজে পড়া যায়: এর কোডিং পদ্ধতি খুব সহজে বোধ যায়।
3. Python এর সোর্স কোড সহজে মেইন্টেইন করা যায়।
4. Python এর সাইক্রো অনেক সমৃদ্ধ এব় ঘর সাইজের, যা UNIX, Windows, Macintosh ইত্যাদি অপারেটিং সিস্টেমে সাপোর্ট করে। অর্থাৎ পাইথন প্র্যাটফর্ম ইন্স্টলেশনের সহজ অটোম্যাটিক সেট হয়ে যাবে।
5. Python এ ইন্টারেক্টিভ (পারস্পরিক) কোডিং এর সুবিধা রয়েছে, যার ফলে টুকিটকি বিভিন্ন কাজের কোড এর জন্য একই সাথে Testing ও Debugging করা যায়।
6. Python Portable, অর্থাৎ বিভিন্ন রকমের ডার্কওয়ার Platform এ এটি একই ভাবে কাজ করে।
7. Python প্রথমে কিছু Low Level Module দিয়ে শুরু করা যায়, এব় পরে প্রয়োজন অনুযায়ী প্রয়োজন নাম রকম টুলস (Tools) সহযোগিতা করে সেই Module কস্টো আরো সমৃদ্ধ করতে পারে।
8. প্রায় সব ধরনের বাণিজ্যিক ডাটাবেইজ এ Python এর ইন্টারফেস থাকে।
9. Python GUI Applications সাপোর্ট করে, যার সাহজে বিভিন্ন System Call, Library & Windows system বৈরি অগ্রণী ছানাক্ষুণি করা যায়।
10. বড় সাইজের প্রয়োজনের জন্য Shell টিপ্পিং এর মতো Python এর ক্লিন্ট ভাল কাজ করে।

অর্থ: পাইথনের ব্যবহার তলো লিখুন।

উত্তর: পাইথন একটি অসাধারণ এব় শক্তিশালী প্রোগ্রামিং শ্যাখায়ে বিদ্যুত এব় শহুরু ব্যবহার রয়েছে। সাধারণত মুক্ত সফটওয়্যার নির্মাণের জন্য পাইথন ব্যবহার করা হয়। টেক জগতের বিভিন্ন বড় বড় প্রকল্প সমূহ: Zope Application Server, AMnet Distributed File Store, ইন্টিউবসহ আরো অনেক বড় বড় প্রকল্পে পাইথন

ব্যবহার করা হয়। নিচে পাইথনের কিছু উচ্চতমূর্ণ ব্যবহার তুলে ধরা হলো।

1. গ্রেবডিটিক এব় অটোমেশন টাইপের সফটওয়্যার তৈরিতে।
2. বাণো ইনফরমেটিকস এর ক্ষেত্রে।
3. ন্যাচারাল ল্যাঙ্গুেজ প্রসেসিং এর ক্ষেত্রে।
4. মেশিন লার্নিং ও অর্টিফিশিয়াল ইন্টেলিজেন্সের এর ক্ষেত্রে।
5. গ্রেবে জ্ঞান তৈরিতে।
6. ইন্টেলিজেন্ট সিকিউরিটি টুলসের ক্ষেত্রে।
7. সাইবার সিকিউরিটি ও কোর সিকিউরিটি টুলস হিসেবে।
8. ইন্টারনেট প্রোগ্রাম ও Database Application এ।
9. ইন্টারনেট প্রিসিং ও তথ্য বিশ্বেদে।
10. এফিকাল ইউজার ইন্টারফেস তৈরিতে।
11. গ্রেবে জ্ঞান সিকিউরিটি ভ্যানার হিসেবে।

Python এনভারিনেট ভেরিয়েবল:

নিচে Python এর কিছু উচ্চতমূর্ণ এনভারিনেট ভেরিয়েবল এর নাম ও সংক্ষিপ্ত বর্ণনা দেয়া হল:

ভেরিয়েবল (Variable)	বর্ণনা (Description)
PYTHON PATH	এটি অনেকটাই PATH ভেরিয়েবলের মত। এই ভেরিয়েবলের কাজ হচ্ছে Python প্রোগ্রাম এ ইম্প্লোট করা মডিউল ফাইল ভল্ডের লোকেশন টিক করা এব় সেই অনুযায়ী Python interpreter কে নিম্নের সিদ্ধান্ত করা। PYTHONPATH ভেরিয়েবলটি Python সোর্স কোড ও এর লাইব্রেরি ভিরেক্ট রেকর্ড করে। PYTHONPATH ভেরিয়েবলটি সাধারণত Python ইন্স্টলেশনের সব অটোম্যাটিক সেট হয়ে যাবে।
PYTHON HOME	এই ভেরিয়েবলটি Python সোর্স কোডের ইনিশিয়ালজেশন ফাইল এর লোকেশন রেকর্ড করে, এব় যখনই interpreter চালু করা হবে তখনই এই ভেরিয়েবলটি ফাইলটির নাম pythonrc.py, এব় এটে PYTHONPATH ভেরিয়েবল সেট ও মডিফাই করার ক্ষমতা দাতে।
PYTHON STARTUP	এই ভেরিয়েবলটি Windows এ ব্যবহৃত হয় এব় এর কাজ হচ্ছে Python কে ইন্স্ট্রাকশন দেয়া যাতে এটি যেকোনো ইম্প্লোট স্টেটমেন্টের প্রথম কেস-ইনসেপ্টিভ ম্যাচ শুরু দের ক্ষেত্রে পারে। এই ভেরিয়েবল এ যেকোনো মান সেট করলেই ভেরিয়েবলটি একটিপ্পেট হবে।
PYTHON CASEOK	এটি একটি alternative module search path, এটি সাধারণত PYTHONSTARTUP ও PYTHONPATH এর ভিরেক্টরিস এই সমৃদ্ধ থাকে, এব় এর কাজ হচ্ছে মোডিউল লাইব্রেরির ছানাক্ষুণি করা যায়।

Python প্রোগ্রাম এর কোড রাখ করা:

Python প্রোগ্রাম এর কোড তাও রাখ করা যায়। যেমন:

1. সেব অপারেটিং সিস্টেমে ক্লাউড-লাইন ইন্টারফেসের ধাকে অথবা সেব টাইডে (Shell Window) ধাকে, সেব ক্ষেত্রে সরাসরি

Interactive Interpreter থেকে Python প্রোগ্রাম এর কোড রাখ করা যায়। এসব ক্ষেত্রে, ক্ষমতা লাইনে Python প্রোগ্রাম এর কোড লেখে Enter চাপুন এব় সরাসরি Python এ কোডিং করা যায়।

2. Python Application এর সাথে ইন্টারফেসের সংযুক্ত করে ক্ষমতা লাইনে ক্লিক রাখ করানো যায়।
3. অপারেটিং সিস্টেমের Graphical User Interface (GUI) Application যদি Python সাপোর্ট করে তবে GUI Environment থেকেও Python রাখ করানো যায়।

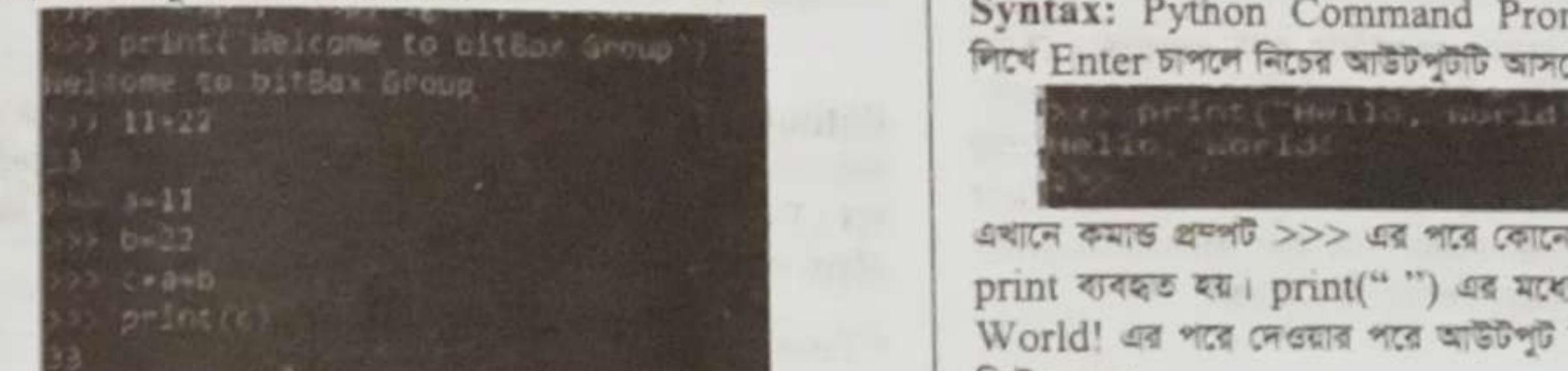
```
else:  
    return(recur_fibo(n-1)+recur_fibo(n-2))
```

Module Scope Variables (মডিউল কোড ভেরিয়েবল): পাইথনের স্টার্টার্ট লাইব্রেরিতে বিভিন্ন ধরনের মডিউল আছে এব় প্রতিটি মডিউলের মধ্যে একাধিক মডিউল রয়েছে। এই অন্তে আমরা একেজনে কোনো মডিউল বা তার মেথডকে কল করতে পারি। যেমন:

```
import math  
print(math.ceil(7.7))
```

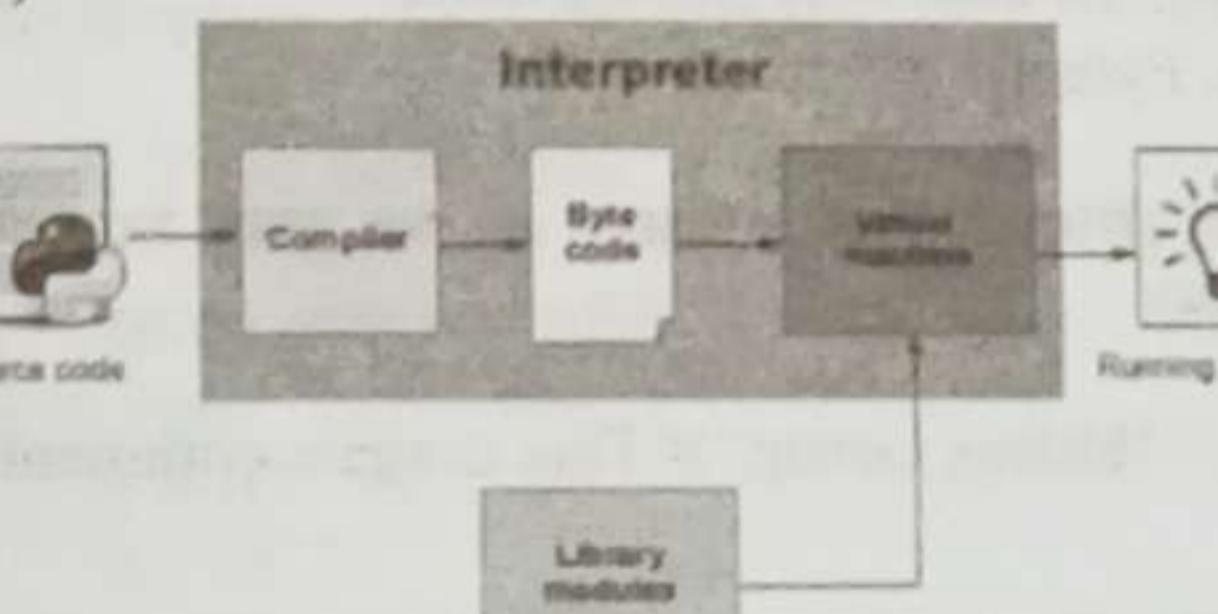
Main Program (মেইন প্রোগ্রাম): প্রোগ্রামের এই অংশে মূল কোড গুলো দেখা হয়। প্রোগ্রামের মধ্যে ব্যবহৃত অন্যান্য মেথড ও অবজেক্ট সমূহ এই অংশেই কল করতে হয়।

পাইথন প্রোগ্রাম স্ট্রাকচার ও এর ইন্টারিটেশন প্রতিক্রিয়া:



```
>>> print("Welcome to bitBox Group")  
Welcome to bitBox Group  
>>> 11+22  
33  
>>> 3*11  
33  
>>> 6*22  
132  
>>> 4*4*4  
64  
>>> 2**100  
1.12589990684e+30  
>>>
```

Python Interpretation Process (পাইথন ইন্টারপ্রিটেশন প্রক্রিয়া):



পাইথনের গোড়ার স্ট্রাকচার (Python Overall Structure):

পাইথন প্রোগ্রামের স্ট্রাকচার চারটি অংশে বিভক্ত। যেমন:

1. Import Statements
2. Function and Class Definition
3. Module Scope Variables and
4. Main Program

Import Statements (ইম্পোর্ট স্টেটমেন্টস): কোনো লাইব্রেরি ক্লাসকে প্রয়োজন করতে চাইলে সে ক্লাসটি বা ক্লাসগুলো যে প্রাক্কেজে আছে তা ইম্পোর্ট স্টেটমেন্টস এর মাধ্যমে ডিক্রেয়ার করতে হয়। যেমন: import random, import math ইত্যাদি।

Function and Class Definition (ফাংশন ও ক্লাস ভেক্সিনেশন): এই অংশে প্রয়োজন ব্যবহৃত যাবতীয় ফাংশন ও ক্লাস সমূহকে ডিফাইন বা বর্ণনা করা যায়। যেমন:

def recur_fibo(n):

if n<=1:

return n

1. আইডেন্টিফায়ার এর নাম 'A' থেকে 'Z' অথবা 'a' থেকে 'z' অব্দে underscore (_) দিয়ে তা হ্যাএ এব় এরপর যেকোনো অক্ষর, underscore কিংবা সংখ্যা (0-9) ব্যবহৃত হয়।

2. Python এর Identifier হিসেবে @, %, % ব্যবহার করা যায় না।

3. ভাষাড়া, Python কেইস সেপিটিভ (Case Sensitive), অর্থাৎ A এব়, a এখানে আলাদা অর্থ বহন করে। যেমন: Python identifier এ bitBox এব় BitBox এর দুটি আলাদা নাম এব় তারা ভিন্ন ভিন্ন আইডেন্টিফায়ার। ক্ষমতা ক্লাস name তলো বড় হ্যাট হাতের অক্ষর দিয়ে তা হ্যাএ অর্থ বহন করে, অন্য সব নামকরণ তক হ্যাট হাতের অক্ষর দিয়ে।

4. কোন identifier যদি দুটুমাত্র একটি underscore (_) দিয়ে তা হ্যাএ তাৰ অৰ হচ্ছে এটি একটি 'private identifier'।

5. যদি কোন identifier পৰ পৰ দুটি underscore (_) দিয়ে নামকরণ হয় তাৰ মেতি 'strongly private identifier'।

৬. যদি কোন identifier এর নাম পর দুটি underscore (_) দিয়ে শেষ হয়, তবে তাকে language-defined special name বলে।

কী-ওয়ার্ড (Key-word) সমূহ: নিচে Python এর কী-ওয়ার্ড তালোর লিস্ট দেয়া হল: এই কী-ওয়ার্ড তালোর বিশেষ অর্থ রয়েছে এবং Python এ কোন প্রকার কল্পনাটি, ভেরিয়েবল অথবা আইডেন্টিফাইয়ার হিসেবে এই শব্দগুলো ব্যবহার করা যায়। Python কী-ওয়ার্ড এ শুধুমাত্র হোট হতের অকর ব্যবহৃত হয়।

and	def	exec	if	not	return
assert	del	finally	import	or	try
break	elif	for	in	pass	while
class	else	from	is	print	with
continue	except	global	lambda	raise	yield

লাইন ও মার্জিন: পাইথনে লাইন এবং মার্জিন খুবই গুরুত্বপূর্ণ। কেবল লাইন এবং মার্জিন দ্বারা স্টেটমেন্টের অবস্থান নির্ধারণ করা হয়। যেমন-ক্ষেত্রে স্টেটমেন্টকে একই Block এর সুবাকে প্রতিটি স্টেটমেন্টের প্রক্রিয়া সমান সংখ্যাক গ্যাপ রাখতে হয়। অন্যথার, একই Block'র Statement তালো ভিন্ন ভিন্ন মার্জিনে তরঙ্গ হলে প্রোগ্রামিং এ error দেখা দিবে। উদাহরণশৰূপ, নিচের কোডটি ঠিক আছে।

```
if True:
    print("Answer")
    print("True")
else:
    print("Answer")
    print("False")
```

কিন্তু, print("Answer") এবং print("False") এর মার্জিন এ ভিন্নভাৱে দাকায় নিচের কোডটি ঠিক নয়। কারণ, একই Block'র পর পর দুটি স্টেটমেন্ট এর মার্জিন ভিন্ন হলে পাইথনে error দেখাব।

```
if True:
    print("Answer")
    print("True")
else:
    print("Answer")
    print("False")
```

একাধিক লাইনের স্টেটমেন্ট: Python এর Statement সাধারণত স্বতন্ত্র লাইনে দিয়ে শেষ হয়। তবে কোন স্টেটমেন্টের শেষে continuation character () দিয়ে স্টেটমেন্টটি আরও চলবে (continued statement) বোানো হয়। যেমন:

```
total = item_one +
       item_two +
       item_three
```

কিন্তু, কোন স্টেটমেন্ট যদি [], {}, বা () Bracket'র ভেতর থাকে তবে continuation character () দেয়ার দরকার নেই। যেমন:

```
days = ['Saturday', 'Sunday', 'Monday', 'Thursday',
        'Friday']
```

Python এ কোটেশন এর প্রয়োগ: Python এর string literal এ একটি (), দুটি ("") কিংবা তিনটি ('' বা "") কোটেশন চিহ্ন ব্যবহার করা যায়, তবে যেই কোটেশন দিয়ে বাক্য তরঙ্গ রয়েছে, সোতি দিয়েই শেষ করতে হবে। তিনটি কোটি ('' বা "") দিয়ে দুটি দুই বা ততোধিক লাইনের string literal সম্পর্ক করা যায়। যেমন: নিচের সবকটি string সঠিক।

```
word = 'bitBox'
sentence = "This is bitBox Publications."
paragraph = """This is a bitBox ICT Master Copy. It
is made for CSE Examiners Students."""
```

Python এ কমেন্ট: String literal এর বাইরে ঝাল চিহ্ন (#) টাইপ করে যদি কোন বাক্য দেখা হয়, তবে সোতি একটি 'কমেন্ট' হিসেবে গণ্য হবে। Python Interpreter সব ধরনের কমেন্টকে ignore করবে বা এডিপ্রোগ্রাম দেখা দিবে। উদাহরণশৰূপ, নিচের কোডটি ঠিক আছে।

```
# First comment
print ("Hello, Python!") # second comment
```

এই ক্ষেত্রটি রান করলে নিচের রেজাল্টটি আসবে।

```
Hello, Python!
```

যেকোনো স্টেটমেন্ট, কিংবা এক্সপ্রেশন এর পরে কমেন্ট সংযোজন করা যায়। যেমন:

```
name = "bitBox Group" # This is again comment
```

শূন্য লাইন (Blank Lines): কোন লাইনে শুধুমাত্র স্পেস, অথবা কোন কমেন্ট থাকলে সেটাকে blank line বলে, এক Python তালো সম্পর্কে ignore করে। Interactive interpreter session চলাকালীন blank line দিয়ে একাধিক বাক্যের স্টেটমেন্ট এর শেষ বোানো হয়। এখন নিচের প্রোগ্রামটি লক্ষ্য করি।

```
#!/usr/bin/python
raw= input("\n\nPress the Enter Key to Exit.")
```

উপরের প্রোগ্রামটি রান করলে, ফলাফল হিসেবে খুঁটি হবে, “Press the Enter Key to Exit.” এবং তারপর কী-বোর্ডে কোন কিছু টাইপ করে Enter press করা মাঝেই প্রোগ্রামটি শেষ হবে। এখানে “\n\n” ব্যবহার করার জন্য আসল বাক্যটি দেখানোর আগে দুটি blank line আসবে।

একই লাইনে একাধিক স্টেটমেন্ট: একই লাইনে (অবশ্যই একই Block এ) একাধিক স্টেটমেন্ট বোাতে সেমি-কোলন (;) চিহ্ন ব্যবহার করা হয়। যেমন:

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

25

[নোট: স্ট্রিং থেকে ইন্টেজার এ কনভার্ট এর সময় ইনপুটে নিউমেরিক ক্যারেক্টার ইনপুট নিতে হবে, অন্যথায় Conversion এ error দেখা দেব। যেমন:

```
>>> int("420a")
```

Traceback (most recent call last):

```
File "<stdin>", line 1, in <module>
```

ValueError: invalid literal for int() with base 10:

'420a'

ক্লোট এ কনভার্সন: স্ট্রিং অথবা ইন্টেজার থেকে ক্লোট এ কনভার্ট করার জন্য float() ফাংশন ব্যবহার করা হয়। যেমন:

```
>>> float("420.125") # String to float Conversion
```

420.125

```
>>> float(420) # Integer to float Conversion
```

420.0

[নোট: i. স্ট্রিং থেকে ক্লোট এ কনভার্ট এর সময় ইনপুটে নিউমেরিক ক্যারেক্টার এবং শুধু একটি দশমিক পঠেটি ইনপুট দেওয়া যাবে, অন্যথায় Conversion এ error দেখা দেব।

ii. দশমিকযুক্ত স্ট্রিং কে ইন্টেজারে কনভার্ট করতে হলেও প্রথমে স্ট্রিং টিকে ক্লোট কনভার্ট করতে হবে, হিসায়ত ক্লোটকে ইন্টেজারে কনভার্ট করতে হবে। যেমন:

```
>>> float("420.125")
```

420.125

```
>>> int(420.125)
```

420

```
>>> int("420.125")
```

Traceback (most recent call last):

```
File "<stdin>", line 1, in <module>
```

ValueError: invalid literal for int() with base 10:

'420.125'

]

স্ট্রিং এ কনভার্সন: str() ফাংশন ব্যবহার করে যেকোন ভ্যারিয়েবলকে স্ট্রিং -এ কনভার্ট করা যায়। যেমন:

```
>>> str(420)
```

'420'

যখন print() ফাংশন এর ভেতর একাধিক ভ্যারিয়েবল দেখা হয় তখন স্ট্রিং -এ কনভার্সন ব্যবহার করতে হয়। যেমন:

```
>>> print("Float = " + str(20.50) + " Integer = " +
```

str(75))

Float = 20.50 Integer = 75

**পাইথন ভ্যারিয়েবল ও ভাইট টাইপ
(Python Variable and Data Types)**

Python ভ্যারিয়েবল (Variable): Variable শব্দের অর্থ- 'চলক' অর্থাৎ যার মান পরিবর্তনশীল। Variable হল Memory তে বরাক্ষকৃত একটি জায়গা। Programmer গুলি লোকেশন ব্যবহার করে ভাতো সংরক্ষণ ও পরবর্তিতে প্রয়োজন অনুযায়ী Retrieve করতে

পারে। এ ভেটা হতে পারে নিউমারিক (যে কোন সংখ্যা) অথবা ক্যারেক্টর, (a,b,c...Z) ইত্যাদি। ভ্যারিয়েবল এ যে ধরনের ডাটা রাখা হয়, তাই হচ্ছে ডেটা টাইপ।

প্রশ্ন: পাইথন ভেরিয়েবল নামকরণের নিয়মগুলি কোন শিখ?

উত্তর: প্রতিটি কম্পাইলারের কিছু সীমাবদ্ধতা আছে। আর তাই কম্পাইলারের সীমাবদ্ধতা করনে পাইথন ভেরিয়েবল লেখার সহজ কিছু নিয়ম ও নীতি মেনে ভেরিয়েবল ডিক্রিয়ার করতে হয়। নিচে সেই নিয়ম তলো তুলে ধরা হলো:

- ভেরিয়েবলের নাম লেখার ক্ষেত্রে অবশ্যই প্রথম একটি অস্কেপ্টেড লেটার (Upper Case বা Lower Case) অথবা Underscore () হবে। যেমন: bitbox, Number, Length, area, volume, data লেখা যাবে। কিন্তু 1stNumber, @number, 5thNumber, %area ইত্যাদি লেখা যাবে না। যদিও ভেরিয়েবলের ক্ষেত্রে Underscore () ব্যবহার করা যায়, কিন্তু পাইথনের কস্টমনেশন হচ্ছে ভেরিয়েবলের নাম সবসময় Lower Case লেটার দিয়ে করতে হবে।
- প্রথম অক্ষরের পর যেকোনো Letter, Underscore (), Number ব্যবহার করা যাবে। যেমন: number5, triangle_area ইত্যাদি।
- ভেরিয়েবল Underscore () ছাড়া অন্য কোনো বিশেষ চিহ্ন (!,@,#,%,&,*,<,>,?,{,},[,]) ব্যবহার করা যাবেন। যেমন: A#, S%, bitbox@.angel! ইত্যাদি ব্যবহার করা যাবেন।
- ভেরিয়েবল নামের মধ্যে কোনো খালি/ফাঁকা আয়ত্তা (Space) রাখা যাবেন। যেমন: bitbox it, First id, area value ইত্যাদি।

Python ডেটা টাইপ (Data Type): পাইথনে ভ্যারিয়েবল ব্যবহার করার পূর্বে তা ডিক্রিয়ার করতে হয় না বা ভ্যারিয়েবলের টাইপ বলে দিয়ে হয় না। আমরা জানি যে, পাইথন অবজেক্ট ভরিয়েস্টেড প্রোগ্রামিং স্যারেজের এর অভ্যেক্ট ভ্যারিয়েবলই হচ্ছে এক একটা অবজেক্ট। যেমনিতে আমরা যে ডেটা রাখব, তা অনেক ধরনের হতে পারে। যেমন: যে কোন নামের, বিভিন্ন লিস্ট, যে কোন টেক্সট ইত্যাদি। পাইথনে পার্টিটি স্ট্যাভার্ট ডেটা টাইপ রয়েছে। সেগুলো হচ্ছে:

- Number
- String
- List
- Tuple
- Dictionary

মাল্টিপ্ল এসাইনমেন্ট (Multiple Assignment): পাইথনে এক সাথে একের অধিক ভ্যারিয়েবলে একই ডেটা এসাইন করতে পারি। যেমন: ICT, IT, CSE = 5.00

যেখানে ICT, IT, CSE নামক ভিনটিতে ভ্যারিয়েবলে প্রতিটাতে 5.00 ভ্যালুটি এসাইন হবে।

আবার প্রয়োজনে একই সাথে একাধিক ভ্যারিয়েবলে একাধিক মান এক সাথে এসাইন করতে পারি। যেমন:

name, pi, number="bitBox Publications", 3.1416, 999

যেখানে আমরা ভিনটে ভ্যারিয়েবলে এক সাথে ভিনটিতে আলাদা আলাদা ভ্যালু এসাইন করেছি। যেখানে pi = 3.1416, name ="bitBox Publications", এবং number = 999

✓ **নামের (Number):** পাইথন দুই ধরনের নামার সাপোর্ট করে, ইটিজার এবং ক্লেটি প্রেস্টে।

ইটিজার (integer) ব্যবহার করার জন্য: myInt = 10

উপরে আমরা myInt নামক একটা ভ্যারিয়েবল নিয়েছি এবং যার মধ্যে 10 রয়েছে। ভ্যারিয়েবলে কোন মান/ভ্যালু/ডেটা রাখাকে বলে এসাইন/Asgin করা।

ক্লেটি প্রেস্ট ব্যবহার করার জন্য: myFloat = 10.6

বা (Or):

myFloat = float(10)

print(myFloat)

যদি একটা ইটিজার ভ্যারিয়েবলের মান পরিবর্তন করে তাতে একটা ক্লেটি প্রেস্ট/স্যারিফ ভ্যালু রাখি, তাহলে তা অটোমেটিকেলি ক্লেটি প্রেস্ট ভ্যারিয়েবলে কনভার্ট হয়ে যাবে। যেমন: myInt = 10.0

পাইথনে সহজেই আমরা যোগ, বিয়োগ, গুণ, ভাগ করে ফেলতে পারি। ফেলতে কিনা প্রোগ্রাম করতে আমাদের অনেক কাজে আসবে।

বেগ এর কাজ:

value1 = 35

value2 = 25

result = value1 + value2

print(result)

যার আউটপুট হবে: 60

বিয়োগ এর কাজ:

value1 = 35

value2 = 25

result = value1 - value2

print(result)

যার আউটপুট হবে: 35 - 25

print result

যার আউটপুট হবে: 10

ভাগ এর কাজ:

value1 = 35

value2 = 7

value3 = value1 * value2

print(value3)

যার আউটপুট হবে: 75

ভাগ এর কাজ:

value1 = 35

value2 = 7

value3 = value1 / value2

print(value3)

যার আউটপুট হবে: 5

প্রিমাইভার / ভাগশেষ এর কাজ:

value1 = 35

value2 = 8

value3 = value1 % value2

print(value3)

যার আউটপুট হবে: 3

String (স্ট্রিং): যে কোন লেখা, ড্যাটাই হচ্ছে এক একটা String।

যেমন:

myString = "Hello bitBox Group!"

print(myString)

এটি একটা স্ট্রিং। স্ট্রিং আমরা ভালু কোটেশন বা সিলেক্সেল ভেতর লিখতে পারি। উপরে লিখেই ভালু কোটেশনে। কিন্তু নিচের মত করেও লিখতে পারি।

myString = 'Hello world!'

print(myString)

সিলেক্সেল কোটেশনের ভেতরে লিখলে একটা সমস্যা হয়ে যাবে, আমরা এপস্ট্রিপস/apostrophes লিখতে পারব না। যেমন: নিচের কোড টুকু লিখলে প্রোগ্রামে ভুল দেখবে:

myString = 'It's Friday, Harun'

print(myString)

এটা বান হবেনো। সিলেক্সেল কোটেশনের মধ্যে লিখা হচ্ছে, যার মধ্যে আবার সিলেক্সেল কোটেশন ব্যবহার করা হয়েছে। এ জন্যই স্ট্রিং তলোকে ভালু কোটের মধ্যে লিখব।

এখন name নামক একটা ভ্যারিয়েবলের মান পরিবর্তন করে তাতে একটা ক্লেটি প্রেস্ট/স্যারিফ ভ্যালু রাখি, তাহলে তা অটোমেটিকেলি ক্লেটি প্রেস্ট ভ্যারিয়েবলে কনভার্ট হয়ে যাবে। যেমন: myInt = 10.0

name = "Saiful"

print("Hello " + name)

উপরে মূলত আমরা দুইটা স্ট্রিং এক সাথে প্রিণ্ট করা হচ্ছে। Hello এবং Saiful। আর দুইটা স্ট্রিং এক সাথে করাকে বলে String Concatenation।

Concatenation: আউটপুটে দুই বা ততোধিক ভেরিয়েবল বা স্ট্রিং পরপর সংযুক্ত করার জন্য ব্যবহৃত তিনিই Concatenation বলে। আরেকটা উদারণ খেয়াল করলে আরো ভালু করে বুঝা যাবে।

print("Hello" + "World")

or

string1 = "Hello"

string2 = "World!"

print(string1 + string2)

or

a = 10

print("The integer is = " + a)

শাখার অপারেটর (Python Operator)

উপরের আলোচনার আমরা একটা ভ্যারিয়েবলে কোন ভ্যালু এসাইন করতে (=) সমান সমান চিহ্ন হচ্ছে একটা অপারেটর, যাকে বলা হয় Equal to অপারেটর। এছাড়াও এর পূর্বে যোগ, বিয়োগ, গুণ, ভাগ ইত্যাদি স্যারিফ এর কাজ হচ্ছে তাগ শেষ বের করা। যেমন:

value1 = 35
value2 = 7
value3 = value1 * value2
print(value3)
অপারেটরের মধ্যে যে ভ্যারিয়েবল বা যে ভেটার উপর কাজ করে, তাকে আমরা বলি অপারেট (Operand)। যেমন: 35 + 25 এ 35 এবং 25 হচ্ছে অপারেট, এবং + হচ্ছে অপারেটর। কিন্তু কিছু অপারেটরের জন্য একটা অপারেট লাগে। কিছু কিছু অপারেটরের জন্য লাগে দুইবা তার অপারেট। এগুলোকে বলা হয় দ্বিপদী অপারেটর।

এগুলোকে বলা হয় একটু লক্ষ্য করি।
প্রিন্ট(19%3)

19 কে 3 দিয়ে ভাগ করলে ভাগশেষ 1 থাকব কথা। তাই এই প্রোগ্রামের আউটপুট হচ্ছে 1।

এক্সপোনেন্ট অপারেটর (Exponent Operators): যদি আমরা লিখি 10^3 তাহলে আমরা পাবো 30. একটা এস্টেরিঙ্গ (*) হচ্ছে সাধারণ গুণ করার অপারেটর। কিন্তু যদি লিখি 10^{*3} মানে দুইটা এস্টেরিঙ্গ ব্যবহার করি, তাহলে এর মান হচ্ছে 10 to the power 3 বা 10^3 এবং যার মান হবে 1000. অর্থাৎ a^{*n} মানে a^n । [এখানে a হচ্ছে বেইজ বা Base, n হচ্ছে Power বা ঘাত।] নিচের প্রোগ্রাম একটু লক্ষ্য করি।
প্রিন্ট(10^{*3})

বেইজ বা Base 10 এর Power বা ঘাত 3. তাই এর আউটপুট মুন না হয়ে অর্থাৎ 30 না হয়ে 1000 হবে।
ভুলোর ডিভ

ভাজকের ক্ষেত্রে নশমিকের পরের সংখ্যা বান দিয়ে পূর্ণ সংখ্যা রিটার্ন করে। আবার যদি ভাজ্য অথবা ভাজক শব্দাতুক হয়, তাহলে পরবর্তী শব্দাতুক সংখ্যা রিটার্ন করবে। যেমন: `print(-11/3)` ক্ষেত্রে রিটার্ন করবে -4.

এসাইনমেন্ট অপারেটর (Assignment Operators): ভ্যারিয়েবলে কোন ভ্যালু এসাইন করা হচ্ছে এসাইনমেন্ট অপারেটরের কাজ। উপরের উপরে আমরা কিভাবে একটা ভ্যারিয়েবলে এসাইন করা হয়, তা দেখেছি। আবার আগে তখন একটা মাত্র এসাইনমেন্ট অপারেটর Equal to (=) এসাইনমেন্ট অপারেটরের ব্যবহার করেছি। এটি ছাড়াও আরো অনেক ক্ষেত্রে Assignment operator রয়েছে। যেমন:

1. $+=$ (Plus equal to)
2. $-=$ (Minus equal to)
3. $*=$ (Product equal to)
4. $/=$ (Division equal to)
5. $\%=$ (Mode equal to)
6. $**=$ (Exponent equal to)
7. $//=$ (Floorequal to) ইত্যাদি।

এই অপারেটর এর $+=$ (Plus equal to), $-=$ (Minus equal to), $*=$ (Product equal to), $\%=$ (Mode equal to) এর ব্যবহার অন্যান্য Language এর মতই। বাকিগুলো আলোচনা করা হচ্ছে।

1. $=$ (Exponent equal to) Operator:** এটা নিচের মতো করে লেখা হচ্ছে:

`Expression1 **= Expression2`, যা (`Expression1 = Expression1 ** Expression2`) এর সমান।

ব্যাখ্যা: মনে করি, $x=2$, $y=5$.

$x=2$

$y=5$

$x**=y$

`print(x)`

উপরের প্রোগ্রামটা রান করে দেখলে আউটপুট পাবো: 32

2. $//=$ (Floor equal to) Operator: এটা নিচের মতো করে লেখা হচ্ছে:

`Expression1 //= Expression2`, যা (`Expression1 = Expression1 // Expression2`) এর সমান।

আমরা জানি দুইটা ভাগ চিহ্ন (/) হচ্ছে ফ্রেজ ডিভিশন। যা ভাগফল হিসেবে ভগ্নাংশের কাছাকাছি একটা পূর্ণ সংখ্যা রিটার্ন করে।

ব্যাখ্যা: মনে করি, $x=7$, $y=2$.

$x=7$

$y=2$

$x//=y$

`print(x)`

উপরের প্রোগ্রামটা রান করে দেখলে আউটপুট পাবো: 3

Logical Operator: পাইথন প্রোগ্রামিং এ তিনটি Logical Operator রয়েছে। যেমন:

Operator	Meaning
and	Logical AND
or	Logical OR
not	Logical NOT

AND operator: মনে করি, x,y,z তিনটি চলক। এখন $(x < y)$ and $(y < z)$ হচ্ছে একটি এক্সপ্রেশন। এখন এই পুরো এক্সপ্রেশন এর মান সত্য হবে যদি $(x < y)$ এবং $(y < z)$ সত্য হয়। $(x < y)$ এবং $(y < z)$ এর মধ্যে কোন একটি মিথ্যা হলে $(x < y)$ and $(y < z)$ এর মান মিথ্যে হবে।

OR operator: মনে করি, x,y,z তিনটি চলক। এখন $(x < y)$ OR $(y < z)$ হচ্ছে একটি এক্সপ্রেশন। এখন এই পুরো এক্সপ্রেশনের মান সত্য হবে যদি $(x < y)$ অথবা $(y < z)$ এর যে কোন একটি সত্য হয়। $(x < y)$ এবং $(y < z)$ দুটি একসাথে মিথ্যা হলে $(x < y)$ OR $(y < z)$ এর মান মিথ্যে হবে।

NOT operator: কোন একটা সত্য ভ্যালুকে মিথ্যে অথবা একটা মিথ্যে ভ্যালুকে সত্য করার জন্য NOT অপারেটরের ব্যবহার করা হয়।

Comparison (Relational) Operators: কোন কিছুর তুলনা করার জন্য Comparison অপারেটরের ব্যবহার করা হয়। এটাকে Relational অপারেটরও বলা হয়। পাইথন প্রোগ্রামিং এর Comparison অপারেটর তালো হলো:

Operator	Meaning
$=$	Equal to
$!=$	Not Equal to
$<$	Less than
$<=$	Less than or equal to
$>$	Greater than
$>=$	Greater than or equal to

Relational এবং Logical Operators এর কয়েকটি উদাহরণ নিচে দেওয়া হল: মনে করি x , y , z তিনটি চলক। x এর মান 5, y এর মান 6 এবং z এর মান 7.

Expression	ব্যাখ্যা	মান
$X < y$	TRUE	1
$X == 5$	TRUE	1
$y == 4$	FALSE	0
$(X+y) > z$	TRUE	1
$(X+y) <= z$	FALSE	0
$X != y$	TRUE	1
$(X < y) \& \& (y == 6)$	TRUE	1
$(X < y) \& \& (z != y)$	TRUE	1
$(X > y) (z != y)$	TRUE	1
$(X > y) \& \& (z != y)$	FALSE	0
$(X < y) \& \& (z == y)$	FALSE	0
$(X < y) (z == y)$	TRUE	1

অর্থাৎ TRUE এর মান জন্য মান 1। এবং FALSE এর মান 0।

Bitwise Operators: পাইথনে ৬ ধরনের বিটওয়াইজ অপারেটর দিয়ে আসা হয়ে আছে। বিটওয়াইজ অপারেটরের তালো হলো:

1. & Binary AND

2. | Binary OR
3. ^ Binary XOR
4. ~ Binary Ones Complement
5. << Binary Left Shift
6. >> Binary Right Shift

আমরা পূর্ব থেকেই জানি যে, কম্পিউটারের সকল ডেটা মেমরিতে সংরক্ষিত থাকে। আর ডেটা তালো সংরক্ষিত থাকে বিট আকারে। অর্থাৎ 0 অথবা 1 আকারে। কোন ভ্যারিয়েবলের বাইনারি রূপ দেখতে চাইলে আমরা bin() ফাংশন ব্যবহার করতে পারি।

9 এর বাইনারি রূপ হচ্ছে 1001। আর দশ এর বাইনারি রূপ হচ্ছে 1010। মনেকরি, $a = 9$, $b = 10$ এরপর এ দুইটার উপর আমরা বিভিন্ন বিটওয়াইজ অপারেশন করতে পারি। যেমন:

Binary AND:
 $a = 9$, $b = 10$
 $c = a \& b$
`print(bin(c))`

Binary OR:
 $a = 9$, $b = 10$
 $c = a | b$
`print(bin(c))`

Binary XOR:
 $a = 9$, $b = 10$
 $c = a ^ b$
`print(bin(c))`

BinaryOne's Complement:
 $a = 9$, $c = \sim a$
`print(bin(c))`

Binary Left Shift:
 $a = 9$
 $c = a << 2;$
`print(bin(c))`

Binary right Shift:
 $a = 9$
 $c = a >> 2;$
`print(bin(c))`

Membership Operators: মেমোরিশিপ অপারেটরের দিয়ে কোন লিস্টে কোন আইটেম আছে কি না, তা যাচাই করা হয়।

1. in
2. not in

`list = [1, 2, 3, 4, 5, 7, 9]`

এখানে উপরে একটা লিস্ট তৈরি করেছি। এখন আমরা নিচে প্রোগ্রামের মাধ্যমে যাচাই করবো যে লিস্টের মধ্যে আমাদের যাচাইকৃত আইটেম বা ভ্যালুটি আছে নাকি নাই?

`list = [1, 2, 3, 4, 5, 7, 9]`

if (2 in list):

 print("2 exist in the list")

else:

 print("2 is not exist in the list")

এছাড়াও অন্য একটি আইটেম লিস্টে না থাকলেও আমরা তা যাচাই করতে পারি। যেমন:

`list = [1, 2, 3, 4, 5, 7, 9]`

if (6 not in list):

 print("6 is not exist in the list")

Identity Operators: আইডেন্টিটি অপারেটরের দিয়ে দুটি অবজেক্টের মেমরি লোকেশন একই কিনা, তা যাচাই করা হয়। পাইথনে দুটি আইডেন্টিটি অপারেটর রয়েছে। যেমন:

1. is
2. is not

এখানে, যদি একই ভ্যালু দুইটা ভ্যারিয়েবলের রাখি, মেমরি সেইভ করার জন্য এ দুইটা ভ্যারিয়েবলে এ ভ্যালুটির লোকেশনে পদ্ধতি করে। যেমন:

```
a = 10
b = 10
if (a is b):
    print ("a and b have same identity")
else:
    print ("a and b do not have same identity")
```

উপরের প্রোগ্রামটা রান করে দেখলে আউটপুট পাবো:

a and b have same identity

এখন আবার দুইটাকে আলাদা ভ্যালু সেট করে রান করে দেখলো কি আউটপুট দিচ্ছে। যেমন:

```
a = 10
b = 13
if (a is not b):
    print ("a and b have different identity")
else:
    print ("a and b do have same identity")
```

উপরের প্রোগ্রামটা রান করে দেখলে আউটপুট পাবো: a and b have different identity

Object in Python: পাইথন প্রোগ্রাম এ ডাটা টাইপ

Data Type Conversion (ডাটা টাইপ কনভেরশন)

Python এ একধরনের ডাটা টাইপ থেকে আরেকটি ডাটা টাইপে রূপান্বয় করার জন্য কিছু বিট-ইন ফাংশন আছে, যেগুলো আউটপুট হিসেবে পরিবর্তিত মানসহ নতুন অবজেক্ট তৈরি করে। নিচে কিছু ফাংশন ও তাদের অর্থ দেয়া হল। যেমন:

ফাংশন (Function)	বর্ণনা
<code>int(x [,base])</code>	x কে একটি পূর্ণসংখ্যায় পরিবর্তন করে, এবং base লিখাটি বেস কে নির্দেশ (specify) করে যদি x একটি স্ট্রিং হয়।
<code>long(x [,base])</code>	x কে একটি সীর্জ পূর্ণসংখ্যায় পরিবর্তন করে, এবং base লিখাটি বেস কে নির্দেশ (specify) করে যদি x একটি স্ট্রিং হয়।
<code>float(x)</code>	x কে একটি floating-point সংখ্যায় পরিবর্তন করে।
<code>complex(real [,imag])</code>	জটিল সংখ্যা তৈরি করে।
<code>str(x)</code>	x কে একটি string representation এ পরিবর্তন করে।
<code>repr(x)</code>	x কে একটি expression string এ পরিবর্তন করে।
<code>eval(str)</code>	একটি string কে অভাস্যোটি করে এবং একটি নতুন অবজেক্ট সৃষ্টি করে।
<code>tuple(s)</code>	s কে একটি tuple এ পরিবর্তন করে।
<code>list(s)</code>	s কে list এ পরিবর্তন করে।
<code>set(s)</code>	s কে set এ পরিবর্তন করে।
<code>dict(d)</code>	একটি dictionary তৈরি করে। তবে, d কে অবশ্যই (key,value) ফর্ম্যাট এর tuples হতে হবে।
<code>frozenset(s)</code>	s কে frozen set এ পরিবর্তন করে।
<code>chr(x)</code>	একটি integer কে একটি character এ পরিবর্তন করে।
<code>unichr(x)</code>	একটি integer কে একটি Unicode character এ পরিবর্তন করে।
<code>ord(x)</code>	একটি single character কে এর পূর্ণসাংখ্যিক (integer) মানে প্রকাশ করে।
<code>hex(x)</code>	একটি integer কে একটি hexadecimal string এ পরিবর্তন করে।
<code>oct(x)</code>	একটি integer কে একটি octal string এ পরিবর্তন করে।

পাইথন ইনপুট, আউটপুট এবং ইম্পোর্ট বিট-ইন ফাংশন

Python Programming এর ক্ষেত্রে **input**: Python এ user কর্তৃক input নেওয়ার জন্য `input()` ব্যবহার করা হয়। `input()` ব্যবহার করে দেকেন ধরনের ডাটা ইনপুট নেওয়া যায়। `input` নিয়ে তা একটা ভ্যারিয়েবলে এসাইন করার জন্য নিচের মত করে লেখা হয়:

যেমন:

```
var = input()
```

যেমন:

```
var = input()
```

print (var)

একজন user input হিসেবে কি ধরনের ডাটা প্রদান করবে তা ফাংশনের ভেতরে Double quotation (" ") এর মধ্যে হিসেবে আকারে উল্লেখ করে দেওয়া যায়। যেমন:

```
Guide_name = input("Enter Your Guide Name: ")
print("You Entered: ", Guide_name)
```

লাইন দুটি রান করলে (i) `var = input("Enter Your Guide Name: ")` লাইনটি রান হবে, অর্থাৎ var এ `input` হিসেবে Enter Your Guide Name: এরপর bitBox DUET CSE Admission Guide টাইপ করতে হবে।
(ii) এরপর `print("You Entered: ", Guide_name)` লাইনটি রান হবে। Output টি হবে নিচেরপ্রতি।

Output:

```
Enter Your Guide Name: bitBox DUET CSE
Admission Guide
You Entered: bitBox DUET CSE Admission
Guide
```

বিদ্রোহ: Python এ `input` সবসময় স্ট্রিং হিসেবে বিবেচনা করে।
ধরন একজন ব্যবহারকারী একটি সংখ্যার বর্ণ নির্ণয় করতে চাচ্ছে সেক্ষেত্রে যে নাহারটি ইনপুট নেওয়া হবে তাকে প্রথমে এ নাহারটিকে ইন্টিজারে কনভার্ট করে নিতে হবে। অন্যথায়, Traceback করবে।
কেবল পাইথন সকল ইনপুট স্ট্রিং হিসেবে ধারণ করবে।
যেমন:

```
number = input("Enter Any Number: ")
number= int(number)
square = number * number
print ("Square of Given Number is: ", square)
```

Output:

```
Enter Any Number: 5
Square of Given Number is: 25
```

ইনপুট নেওয়ার তরঙ্গেই নাহারটিকে ইন্টিজারে কনভার্ট করে নিতে পারি।
যেমন:

```
number = int(input("Enter Any Number: "))
square = number * number
print ("Square of Given Number is: ", square)
```

Python Programming এর ক্ষেত্রে **Output**: পাইথন বিট-ইন(নিয়ন্ত্রণ) ফাংশন `print()` ব্যবহার করে আউটপুট কার্য সম্পাদন করতে হয়। স্টার্ট আউটপুটের জন্য যথাক্রমে `print()` ফাংশন ব্যবহৃত হয়। স্টার্ট আউটপুট ডিভাইসে (যেমন: জিনে বা পর্সন) আউটপুট নেওয়ার জন্য আমরা `print()` ফাংশন ব্যবহার করি। যেমন:

```
>>> print("Hello World!!")
Hello World!!
```

এবং

```
>>> print(10+7)
17
>>>
```

এবং from কীওয়ার্ড ব্যবহার করে কোনো মডিউলের নির্দিষ্ট

কোনো সংজ্ঞাকেও ইম্পোর্ট (import) করতে পারি। যেমন:

```
>>> from math import pi
>>> pi
Output: 3.1415926535
```

যখন একটি মডিউলকে ইম্পোর্ট করা হয় তখন এটি `sys.path` এ নির্ধারিত বিভিন্ন স্থানে খোজ করে। নিচে আমার পিসির ডিরেক্টরি লোকেশন এর তালিকা তুলে ধরা হলো:

```
>>> import sys
>>> sys.path
['', 'C:\Python\Python37-32\python37.zip',
'C:\Python\Python37-32\DLLs',
'C:\Python\Python37-32\lib',
'C:\Python\Python37-32',
'C:\Python\Python37-32\lib\site-packages']
```

কন্ট্রোল স্টেটমেন্ট (Control Statement)**Conditional Statement:**

Python এ Control Statement তলোর কাজ অন্যান্য language এর মতই। শুধুমাত্র Syntax এ পার্থক্য রয়েছে। তা একটি একটি করে তুলে ধরা হলো:

1. **Conditional (if):** if কন্ডিশন সত্য হলে এর ত্রুকে যতগুলো লাইন থাকবে সবগুলো execute হবে। অন্যথায়, if এর ত্রুক execute হবে না।

Syntax :

```
if (condition):
    block
```

Example:

```
a = 23
b = 21
if (a>b):
    print("a is Greater than b")
```

Output:

a is Greater than b

2. **Conditional (if-else):** if কন্ডিশন সত্য হলে এর ত্রুকে যতগুলো লাইন থাকবে সবগুলো execute হবে। অন্যথায়, else এর ত্রুক execute হবে।

Syntax :

```
if (condition):
    block-1
else
    block-2
```

Example:

```
a = 21
b = 23
if (a>b):
    print("a is Greater than b")
else:
    print("a is Less than b")
```

উপরের উদাহরণে `math` মডিউলে থাকা সকল সংজ্ঞা (কোড) আমাদের কোপেও বিদ্যমান অর্থাৎ আমরা এর সকল কোড ব্যবহার করতে পারি।

Output: a is Less than b

০. **Conditional (elif বা else if):** if কভিশন সত্য হলে এর তবে ব্যবহার করবে সবগুলো execute হবে। অন্যথার, else if এর কভিশন সত্য হলে এর তবে execute হবে। if বা else if এর কেন্দ্রিত সত্য না হলে else এর তবে execute হবে।
এখনে n-সংখ্যক else if block থাকতে পারে।

Syntax :

```
if (condition):
    block-1
else if (condition):
    block-2
:
else
    block-n
```

Example:

```
a = int(input("Enter 1st Number: "))
b = int(input("Enter 2nd number: "))
if (a>b):
    print("Yes, a is Greater than b.")
elif(a<b):
    print("No, a is Less than b.")
elif(a==b):
    print("Wow! Both are Equal Number.")
```

Output:

```
Enter 1st Number: 33
Enter 2nd number: 33
Wow! Both are Equal Number.
```

Nested if-else (নেটোচ ইফ-এলস): যখন একটা if else এর ভেতরের আরেকটা if else ব্যবহার করা যাবে তখন তাকে কলা হবে।
Nested if-else (নেটোচ ইফ-এলস) : যেমন:

Code:

```
a = int(input("Enter Any Number: "))
if(a > 0):
    if(a%2==1):
        print("Your Entered Number is Positive and
It's a Odd Number.")
    else:
        print("Your Entered Number is Positive and
It's an Even Number.")
else:
    print("Oh!! You Entered a Negative Number.")
```

Output:

```
Enter Any Number: 69
Entered Number is Positive and It's a Odd Number
```

Again, Enter Different Input for different Output:
Enter Any Number: 420

Your Entered Number is Positive and It's an Even Number.

Again, Enter Different Input for different Output:
Enter Any Number: -42069
Oh!! You Entered a Negative Number.

Looping Statement:

১. **Python while loop (পাইথন হোয়াইল):** একই স্টেটমেন্ট বার বার রান করানোর জন্য while loop ব্যবহার করা হয়। একটি কভিশন দেওয়া হয়, যদি কভিশনটি সত্য হয়, তাহলে while লুপের ভেতরে থাকা কোড গুলো রান হবে। আর যদি কভিশনটি মিথ্যে হয়, তাহলে while লুপের ভেতরের কোড গুলো রান হবে না অর্থাৎ আউটপুট আসবেন।

Note: স্থির নয় এমন ঘেরানো ভালুকে পাইথন True হিসাবে সন্তুষ্ট করে। None এক 0 কে False হিসাবে গণ্য করে।

Syntax:

```
while expression:
    block of code
```

expression কলতে একটা কভিশন বা শর্ত দেওয়াকে বুঝায়। যেটা হয় সত্য হবে না হয় মিথ্যা। যতক্ষণ পর্যন্ত এই কভিশনটি সত্য হবে, ততক্ষণ পর্যন্ত while লুপটি চলবে। যেমন:

Code:

```
x = 0
while x < 3:
    print(x)
    x=x+1
print("Well Done!!")
```

Output:

```
0
1
2
```

Well Done!!

else সহ while লুপ: for লুপের ভেতরে while লুপেরও অতিরিক্ত else block থাকতে পারে। একেজে একটি while লুপের কভিশন False হলে else অশ্ব সম্পূর্ণ হয়। while লুপ বন্ধ করার জন্য break স্টেটমেন্ট ব্যবহার করা হয়। এটি else অশ্বকেও এড়িয়ে যায়। সুতরাং লুপের else অশ্ব কেবল তখনই সম্পূর্ণ হবে যখন কোনো break স্টেটমেন্ট থাকে না এবং কভিশন False হবে। যেমন:

Code:

```
x = 0
while x < 5:
    print("Inside while loop")
    x = x + 1
else:
    print("\nIt's Inside of else ")
```

Output:

```
Inside while loop
Inside while loop
Inside while loop
```

Inside while loop
Inside while loop

It's Inside of else

Python for loop (পাইথন ফর লুপ): কোন অনুক্রমের বা ধারাবাহিক (Sequence) সব গুলো আইটেম এর মধ্যে লুপ চালানোর জন্য for লুপ ব্যবহার করা হয়। এই লুপে variable বা সরাসরি string, array নিয়ে লুপ চালানো যায়।

Syntax:

```
for initialization_variable in data_variable/message:
    block of line
```

যেমন: bitBox এই শব্দটির মধ্যে যত গুলো লেটার আছে, তার মধ্যে লুপ চালাবো, এবং লেটার গুলো একটা একটা করে প্রিন্ট করে আউটপুট দেবো।

```
for letter in 'bitBox':
    print ("Current Letter: ", letter)
```

Output:

```
Current Letter: b
Current Letter: i
Current Letter: t
Current Letter: B
Current Letter: o
Current Letter: x
```

এখনে, letter হল initialization_variable এবং 'bitBox' হল message বা string। প্রতিটি letter এর জন্য লুপটি একবার করে দ্বৰাৰে। যেহেতু 'bitBox' এ ৬টি letter আছে সেহেতু লুপটি ৬ বার দ্বৰাৰে। এবং এর ভেতরের block মোট ৬ বার execute কৰবে।

Nested for loop (নেটোচ ফর লুপ): একটা for লুপের ভেতর আরেকটা for লুপ ব্যবহার করাকে নেটোচ for লুপ বলে। একটা প্যাটার্ন প্রিন্ট করার উদাহরণ দেখলে বিষয়টা আরো ক্লিয়ার হবে। যেমন:

*

**

উপরের প্যাটার্নটি আউটপুট হিসেবে দেখতে চাই়। তার জন্য একটা প্রয়োগ for লুপ ব্যবহার করে।

Code:

```
for i in range(1,6):
    for j in range(i):
        print("*",end=" ")
    print(" ")
```

Python break and continue (পাইথন ব্ৰেক এবং কন্টিনিউ):
break (ব্ৰেক): নিমিট শর্ত সাপেক্ষে লুপ থেকে বাহির হওয়া বা লুপ থেকে কোর জন্য break keyword ব্যবহার কৰা হয়। যেমন:

Code:

for letter in "bitBox":

```
    if letter == "B":
        break
    print ("Current Letter: ", letter)
    print("\nOh No!! Loop is End.")
```

Output:

```
Current Letter: b
Current Letter: i
Current Letter: t
```

Oh No!! Loop is End.

উপরের প্রয়োগ bit পৰ্যন্ত আউটপুট দিবে। যখনই B পাৰে, তখন লুপ থেকে বেৰ হয়ে যাবে। অর্থাৎ if এর শর্ত ব্যবহৰ সত্য হয়, তখন 'break' execute হয় এবং লুপটি end হয়ে যাব।

Continue (কন্টিনিউ): শর্ত সাপেক্ষে লুপের ভেতরের লাইনগুলো নিমিট সংস্থাকৰণ skip কৰা জন্য যে keyword ব্যবহার কৰা হয়, তাহলে Continue। যেমন:

Code:

```
for letter in "bitBox":
    if letter == "B":
        continue
    print ("Current Letter: ", letter)
print("\nOh No!! Loop is End.")
```

Output:

```
Current Letter: b
Current Letter: i
Current Letter: t
Current Letter: o
Current Letter: x
```

Oh No!! Loop is End.

break এর ক্ষেত্ৰে যখন letter == "B" পাৰে তখনই লুপটি শেষ কৰে দেৱ। কিন্তু continue এর ক্ষেত্ৰে B বাদে সবগুলোই letter প্রিন্ট কৰে।

Function in Python Programming (পাইথন প্ৰোগ্ৰামিং এ ফাংশন)

function is Python Programming (পাইথন প্ৰোগ্ৰামিং এ ফাংশন): function হল একটি প্ৰয়োগ সেগমেন্ট, যা কিছু সুনিশ্চিত কজৰ কৰে থাকে। ফাংশন হচ্ছে পুনৰাবৃত্ত কৰাতের একটি block। প্ৰয়োগ আমোৰ সাধাৰণত দুই ধৰণের ফাংশন ব্যবহাৰ কৰে থাকি।

Built In Function: যে সকল ফাংশন ব্যবহাৰকাৰীক তৈৰি কৰতে হয় না প্ৰয়োগ আলো থেকে সেট কৰা থাকে, সেগুলো বিন্ট-ইন-ফাংশন।
যেমন: print(), input() ইতাদি।

User Defined Function: যা ব্যবহাৰকাৰী নিজেৰ প্ৰয়োজনে নিমিট কৰে ব্যবহাৰ কৰা জন্য তৈৰি কৰে থাকে।

Function এৰ basic Syntax:
def function-name(Parameters):

```

statements
return
পিস্টারের বাক্য: এখনে,
1. def নিয়ে ফাংশন তৈ করা হয়।
2. function-name হচ্ছে ফাংশনের নাম। যে নাম নিয়ে
ফাংশনকে কল করা হব বা পুনরায় ব্যবহার করা হয়।
3. Parameters এর সাথে value পাস করা হয়।
4. ফাংশনটি কি রিটার্ন করবে তাই return এ উল্লেখ করে নিতে
হয়। তবে ফাংশনে রিটার্ন করা বাধাতাত্ত্বিক নয়।
হেমন: আবরা bitBox নামে একটা ফাংশন সিখ, যেটাকে কল করলে
DUET CSE Master Copy হিস্ট করবে।
def bitBox():
    print("DUET CSE Master Copy")
Output: DUET CSE Master Copy
একটা ফাংশনকে বত বাব ইচ্ছা ততবর কল করা যায়। এটাই হচ্ছে
ফাংশনের কল সুবিধা। যেমন, উপরের ফাংশনটিকে আবরা হণি ও বাব
কল করলে আউটপুট ও বাব DUET CSE Master Copy হিস্ট করে। ব্যবহার print("DUET CSE Admission Guide") সেখা
প্রয়োজন হয় না।
def bitBox():
    print("DUET CSE Admission Guide")
bitBox()
bitBox()
bitBox()
Output:
DUET CSE Admission Guide
DUET CSE Admission Guide
DUET CSE Admission Guide

```

Parameter of function (ফাংশনের প্যারামিটার):

Parameter বা Argument হল একবারের ভেরিয়েবল, যার সাথে
function এ value পাস করা হয়। Parameter ব্যবহার করে দুটি
সংখ্যা মোগের জন্য একটি উদাহরণ নিন্তে দেওয়া হল:

```

Code:
def add(x,y):
    return (x+y)
print("Summation is: ", add(7,5))
Output: Summation is: 12

```

উপরের ফাংশনে দুটি parameter ব্যবহার করা হয়েছে। যখন
ফাংশনটি কল করা হব তখন parameter দুটির value উল্লেখ করে
দেওয়া হয়। অর্থাৎ, add(7,5) সেখা হত তখন x=7 ও y=5 হবল
করে। এবং কাশেনের ভেরিয়েবল সাইনেস্টেলে execute করে। 7 ও 5 কে
যোগ করে 12 return করবে। অনুরূপভাবে, দুটি সংখ্যার মোগের জন্য
ফাংশনটিকে বতবর ইচ্ছা ততবর ব্যবহার করা যায়।

Default Parameter (ডিফল্ট প্যারামিটার): কোন ফাংশন ডিফাইন
করল সবৰ তাৰ ডিফল্ট প্যারামিটার সেট করে দেওয়া যাব। ডিফল্ট
প্যারামিটার সেট করে দেওয়াৰ অৰ্থ হচ্ছে যদি ফাংশন কল কৰাৰ সময় এই
প্যারামিটারেৰ অবিচারে জন্য কোন ভালু সেট কৰে দেওয়া না হয়।

তাহলে ফাংশন ডিফল্ট ভালুটি ব্যবহাৰ কৰবে। একটা উদাহৰণ দেখলে
সহজে বুঢ়া যাবে। হেমন:

```

Code:
def max(x, y=0):
    if (x > y):
        return format(x) + " is Greater than " +
format(y)
    elif (x < y):
        return format(x) + " is Less than " + format(y)
    elif (x == y):
        return format(x) + " and " + format(y) + " Oh!!
Both are Same Value."

```

```

print(max(7, 5))
print(max(9))
print(max(6, 6))

```

Output:

7 is Greater than 5
9 is Greater than 0
6 and 6 Oh!! Both are Same Value.

উপরের উদাহৰণে প্রোগ্রামটি লক্ষ কৰলে দেখা যাবে যে, আবরা দুইটা
সংখ্যার মধ্যে বত হোট নিৰ্ণয়ের জন্য একটা ফাংশন নিয়েছি। যখন
ফাংশনে ছিটীয় প্যারামিটাৰ হিসেবে কোন ভালু পাঠালো হয়লি, তখন
ডিফল্ট ভালু 0 ব্যবহাৰ কৰা হয়েছে।

ফাংশন থেকে একাধিক ভালু রিটার্ন কৰা: পূৰ্বৰে উদাহৰণে আবরা দেখছি
যে ফাংশন থেকে কিভাবে একটা ভালু রিটার্ন কৰা যাব। পাইথন
প্রোগ্ৰাম এৰ কেবলে ফাংশন থেকে একাধিক ভালু ও রিটার্ন কৰা যাব।
নিচে প্রোগ্রামটি লক্ষ কৰি। হেমন:

```

def binary(x,y):
    return bin(x),bin(y)
print(binary(9,10))

```

এখনে একটা ফাংশন সেখা হয়েছে, যেখনে দুইটা সংখ্যা পাঠালো এই
সংখ্যা দুইটিৰ বাইনারি ভালু রিটার্ন কৰবে। যদিও দুইটা একসাবে টাপল
আকারে হিস্ট কৰবে। প্রয়োজনে প্রতিটা রিটার্ন ভালু আলাদা আলাদা
কৰেস কৰতে পাৰি। হেমন:

```

Code:
def binary(x,y):
    return bin(x),bin(y)
a,b=(binary(10,15))
print(a)
print(b)

```

Output:

0b1010
0b1111

এখনে রিটার্ন ভালু অসো a এবং b তে আলাদা আলাদা ভাবে এসাইন
হৈব। এৱ্যলু যে কোন ভালু সেট কৰে দেখতে পাৰি।

শেষবাবেৰ মতো আৱেকটা ফাংশন সেখি, যা একটা নথৰ জোড় না
বিজোড় আ দেখাবে। নথৰকাৰী থেকে একটা নথৰ ইনপুট নিব।

তাৰপৰ তা ফাংশনে পাস কৰব। ফাংশন আমাদেৰ বলে দিবে নথৰটি
জোড় না বিজোড়। হেমন:

```

Code:
def checkNumber(n):
    if (n % 2 == 0):
        print("Your Entered Number is Even.")
    else:
        print("Your Entered Number is Odd.")
number = int(input("Enter a Number for
Check: "))
checkNumber(number)

```

Output:
Enter a Number for Check: 58
Your Entered Number is Even.

Again:
Enter a Number for Check: 99
Your Entered Number is Odd.

রিকাৰ্সন (Recursion): যখন কোন ফাংশন নিজেই নিজেকে কল কৰে
তখন তাকে রিকাৰ্সন কলা হয়। আবৰা এই ফাংশনটাকে কলা হয় রিকাৰ্সন
(Recursive) ফাংশন। হেমন:

```

def counter(number):
    print(number)
    number += 1
    counter(number)
counter(1)

```

• Write a recursive function to find the factorial.
Code:

```

def fact(n):
    if (n==0):
        return 1
    else:
        return n*fact(n-1)

```

• Write a recursive function to find the nth
fibonacci number.
Code:

```

def fib(n):
    if (n==1):
        return 0
    elif(n==2):
        return 1
    else:
        return fib(n-1)+fib(n-2)

```

• Python Program to Find if a Number is
Prime or Not Prime Using Recursion.

```

Code:
def check(n, i=2):

```

```

if(n==i):
    return 0
else:
    if (n % i == 0):
        return 1
    else:
        return check(n, i+1)

```

Output:
p=check(5)
if p==1:
 print('not prime')
else:
 print('prime.')

Output: prime.
p=check(6)
if p==1:
 print('not prime')
else:
 print('prime.')
Output: not prime

এক লাইনেৰ ফাংশন - ল্যাব্ডা (lambda): lambda অপারেটৰ
ব্যবহাৰ কৰে পাইথনে এক লাইনেৰ ফাংশন দেখা যাব। lambda এৰ
পৰ শেল দিয়ে আন্তৰিকে নিতে হয়। তাৰপৰ কোলন : চিহ্ন দিয়ে
আৱিধিবেটি একান্তেন্স নিতে হয়। ফাংশনটাকে একটা নাম দেখাৰ জন্য
যেকোন ভালুয়েবলে আসাইন কৰা যাব। হেমন:
sum = lambda x, y : x + y
print("Summation is: ", sum(25, 15))
print("Summation is: ",(lambda x, y : x + y)(25,
15))

Output:
Summation is: 40
Summation is: 40
উদাহৰণেৰ শেষ লাইনে ফাংশনটাকে কোন ভালুয়েবলে আসাইন না
কৰেই যোগফল নিৰ্ণয়েৰ কাজ কৰা হয়েছে।

Class and Object (ক্লাস এবং অবজেক্ট)
Class (ক্লাস): পাইথন হচ্ছে একটি অবজেক্ট এক্রিয়েটেড প্রোগ্ৰামিং
ল্যাঙ্গুেজ। ক্লাস হচ্ছে মূল নকশা আৰ সেই ক্লাসেৰ এক বা একাধিক
অবজেক্ট তৈৰি কৰা যাব, যাৰা এই ক্লাসেৰ সব বৈশিষ্ট্য ধাৰণ কৰবে।

Object (অবজেক্ট) বা ইনস্ট্যান্স (instance): Object (অবজেক্ট)
বা ইনস্ট্যান্স (instance) হচ্ছে তেটা কালেকশন বা ভালুয়েবল এবং
যেখন এমত এৰ সমষ্টি। আবৰ যেখন হচ্ছে একটা ফাংশন, যেগুলো এই তেটা বা
ভালুয়েবলেৰ উপৰ কাজ কৰে। একটি ক্লাসেৰ এক বা একাধিক ইল্ট্যাল
তৈৰি কৰা যাব। এই অবজেক্ট বা ইল্ট্যাল দিয়ে এই ক্লাসেৰ সমষ্ট
কোড়ৰেৰ বা ইল্ট্যালকে কৰা যাব।
হেমন: একটি গাড়িৰ ডিজাইন হচ্ছে ক্লাস আৰ সেই ডিজাইন অনুসৰণ
কৰে যত গাড়ি তৈৰি কৰা হবে, সেই গাড়িগুলো হচ্ছে এই ক্লাসেৰ
অবজেক্ট। ডিজাইনকৃত গাড়িৰ এই বৈশিষ্ট্য গুলোকে বলা হব এই তেটা
আণ্ড্রিভিট (data attribute) আৰ যেসৰ কাজ কৰতে পাৰবে,

সেভলোকে বলা হয় মেথড (method). মূলত মেথড কলো হচ্ছে ক্লাসের ভেতরে তৈরি করা ফাংশন, তবে ক্লাস ও অবজেক্টের ক্ষেত্রে তাদেরকে আমরা মেথড বলি। উদাহরণস্বরূপ আমরা একটা সহজ ক্লাস তৈরি করবো bitBox নামে।

```
class bitBox:
    def saybitBox(self):
        print("DUET CSE Admission Guide")
add = bitBox()
add.saybitBox()
```

Output: DUET CSE Admission Guide

এখানে class bitBox দিয়ে ক্লাসের নাম দেওয়া হয়েছে। এরপর এর ভেতরে ক্লাসের ভেটা এবং মেথড কলো দেওয়া হয়েছে। আমদের bitBox ক্লাসে একটি মাত্র মেথড রয়েছে saybitBox নামে। যাকে কল করলে DUET CSE Admission Guide প্রিণ্ট করবে। add = bitBox() এখানে bitBox ক্লাসের একটি ইনস্ট্যান্স তৈরি করে নিয়েছি, add নামে। এরপর add.saybitBox() দিয়ে bitBox ক্লাসের saybitBox মেথড টাকে কল করেছি। তা DUET CSE Admission Guide প্রিণ্ট করেছে।

প্রশ্ন: পাইথনে মধ্যে সেলফ (self) কি?

উত্তর: self অসলে একটা কনসেন্সিয়াল নাম। আমরা চাইলে এটাকে বনলে দিতে পারি। self হল ক্লাস ইনস্ট্যান্সের একটা রেফারেন্স। ইনস্ট্যান্স ভ্যারিয়েবল ডিক্রিয়ার করার সময় self এর রেফারেন্স ব্যবহার করা হয়। ক্লাস ভ্যারিয়েবল আর ইনস্ট্যান্স ভ্যারিয়েবলের ভিত্তি পার্থক্য হল ক্লাস ভ্যারিয়েবল ক্লাসের সব ইনস্ট্যান্সের ভিত্তেই শেয়ার্ড বা শেয়ারেবল (Shared or Shareable), অন্যদিকে ইনস্ট্যান্স ভ্যারিয়েবল প্রতিটা ইনস্ট্যান্সে ইউনিক (Unique)।

Python Data Structure (পাইথন ভেটা স্ট্রাকচার)

List (লিস্ট)

লিস্ট শব্দের বাংলা অর্থ হচ্ছে তালিকা। লিস্ট হল আইটেমের একটি তালিকা। Python এ লিস্টের টাইপ নির্দিষ্ট করে দেওয়ার প্রয়োজন পড়ে না। অর্থাৎ, একটি লিস্ট যেকোন ধরনের ভাটা রাখা যায়। লিস্ট ডিক্রিয়ার করতে হয় Bracket ([]) এ আবক্ষ ও কমা (Commas) দিয়ে আলাদা আলাদা করে। যেমন:

```
list=['bitBox','CSE','Master','Copy', 69, 420, 69.99]
tinylist=[786, 'bitBox', 'Legend']
print (list)           // সম্পূর্ণ লিস্ট প্রিণ্ট করার জন্য
print (list[0])        // লিস্টের প্রথম আইটেম প্রিণ্ট করার জন্য
print (list[1:5])      // লিস্টের ২য় আইটেম থেকে ৫ম আইটেম
প্রিণ্ট করার জন্য
print (list [2:1])     // ৩য় আইটেম থেকে তৃতীয় করে পূর্বের
আইটেম প্রিণ্ট করার জন্য
print (tinylist * 2)   // লিস্টের আইটেম ২ বার প্রিণ্ট করার জন্য
print (list + tinylist) // দুইটা লিস্ট একত্রে
(Concatenated) প্রিণ্ট করার জন্য
```

Output:

```
['bitBox', 'CSE', 'Master', 'Copy', 69, 420, 69.99]
bitBox
['CSE', 'Master', 'Copy', 69]
```

```
[]

[786, 'bitBox', 'Legend', 786, 'bitBox', 'Legend']
['bitBox', 'CSE', 'Master', 'Copy', 69, 786, 69.99, 786,
'bitBox', 'Legend']
```

লিস্ট নতুন আইটেম সংযোজন করা: লিস্টের শেষে নতুন উপাদান যোগ করার জন্য append() মেথড ব্যবহার হয়। যেমন:

```
>>> saarc = ["Bangladesh", "India", "Sri Lanka",
"Pakistan", "Nepal", "Bhutan"]
>>> saarc.append("Afghanistan")
>>> print(saarc)
['Bangladesh', 'India', 'Sri Lanka', 'Pakistan', 'Nepal',
'Bhutan', 'Afghanistan']
```

উপরের উদাহরণে saarc লিস্টে প্রথমে “Afghanistan” আইটেমটি ছিলোনা, যা append() মেথড এর মাধ্যমে যুক্ত করা হয়েছে। যার আটটুপুর্ণ প্রিণ্ট করাতে দেখতে পেরেছি।

লিস্টের আইটেমগুলো একটি নিদিষ্ট ক্রমে(অর্ডার অনুসারে) সাজানো: আমরা যদি চাই, কোনো লিস্টের আইটেমগুলো একটি নিদিষ্ট ক্রমে(অর্ডার অনুসারে) সাজাতে, তখন sort() মেথড ব্যবহার করতে পারি। যেমন:

```
>>> saarc.sort()
>>> print(saarc)
['Afghanistan', 'Bangladesh', 'Bhutan', 'India', 'Nepal',
'Pakistan', 'Sri Lanka']
>>> li = [1, 3, 7, 2, 4, 6, 1]
>>> li.sort()
>>> print(li)
[1, 1, 2, 3, 4, 6, 7]
>>>
```

লিস্টের আইটেমগুলো উন্টার্সি ক্রমে সাজানো: লিস্টের উপাদানগুলো উন্টার্সি ক্রমে সাজানোর জন্য reverse() মেথড ব্যবহার করা হয়। যেমন:

```
>>> li = [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> li.reverse()
>>> print(li)
[9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> li = ["mango", "banana", "orange", "potato",
"watermelon"]
>>> li.reverse()
>>> print(li)
['watermelon', 'potato', 'orange', 'banana', 'mango']
>>>
```

লিস্টের কোনো আইটেম বাদ দেওয়া: যদি লিস্টের কোনো আইটেম বাদ দিতে চাই, তখন remove() মেথড ব্যবহার করা হয়। যেই উপাদানটি বাদ দিতে চাই, সেটি remove() এর ভেতরে আর্গুমেন্ট আকারে পাঠাতে হয়। আর যদি এই আইটেমটি লিস্টে না থাকে তাহলে এর মেসেজ আসবে। যেমন:

```
>>> fruits
['apple', 'mango', 'coconut', 'banana', 'orange']
```

```
>>> fruits.remove("coconut")
>>> fruits
['apple', 'mango', 'banana', 'orange']
>>> fruits.remove("pineapple")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x not in list
>>>
```

লিস্টের আইটেম কাউন্ট করা: কোনো আইটেম একটি লিস্টে কতবার আছে, তা দেখার জন্য count() মেথড ব্যবহার করা হয়। যেমন:

```
>>> li = [1, 2, 3, 3, 4, 5, 6]
>>> li.count(3)
2
>>> li.count(5)
1
>>> li.count(10)
0
```

লিস্ট থেকে কোন আইটেম মুছে ফেলা: লিস্ট থেকে কোনো আইটেম মুছে ফেলতে, কিংবা সম্পূর্ণ লিস্ট মুছে del() ফাংশন ব্যবহার করা হয়। যেমন:

```
>>> li
[1, 2, 3, 3, 4, 5, 6]
>>> del(li[0])
>>> li
[3, 3, 4, 5, 6]
>>> del(li)
>>> li
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'li' is not defined
```

Tuple (Tuple)

টাপল হচ্ছে পাইথনের একটি ভেটা স্ট্রাকচার। যা অনেকটা লিস্টের মতোই। লিস্ট ও টাপলের মধ্যে পার্থক্য হল লিস্টে ক্ষেত্রে তৃতীয় বক্সী বা (Square Bracket/Third Bracket) ব্যবহার করতে হয়, আর টাপলের ক্ষেত্রে প্রথম বক্সী (একে ইংরেজিতে বলে parentheses) ব্যবহার করতে হয়। যেমন:

```
tuple=('bitBox','CSE','Master','Copy', 69, 20, 69.99)
tinytuple=(786,'bitBox', 'Legend')
print (tuple)           // সম্পূর্ণ টাপল প্রিণ্ট করার জন্য
print (tuple[0])        // টাপলের প্রথম আইটেম প্রিণ্ট করার জন্য
print (tuple[1:5])      // টাপলের ২য় আইটেম থেকে ৫ম আইটেম
প্রিণ্ট করার জন্য
print (tuple [2:1])     // ৩য় আইটেম থেকে তৃতীয় করে পূর্বের
আইটেম প্রিণ্ট করার জন্য
print (tinytuple * 2)   // টাপলের আইটেম ২ বার প্রিণ্ট করার জন্য
print (tuple + tinytuple) // দুইটা টাপল একত্রে
(Concatenated) প্রিণ্ট করার জন্য
```

Output:

```
('bitBox', 'CSE', 'Master', 'Copy', 69, 786, 69.99)
bitBox
('CSE', 'Master', 'Copy', 69)
```

একটু লক্ষ করে দেখি হে, প্রথম Bracket না নিলে কি?

```
>>> x = 1, 2, 3
>>> type(x)
<class 'tuple'>
এখানে আমরা প্রথম বক্সী ব্যবহার করিনি, তবুও পাইথন x কে একটি টাপল হিসেবেই গ্রহণ করে। এভাবে কি একটি আইটেমের টাপল তৈরি করা সম্ভব? চলো সেই প্রশ্নের উত্তর খুঁজি। যেমন:
```

```
>>> x = 1
>>> type(x)
<class 'int'>
>>> x = 1,
>>> type(x)
<class 'tuple'>
আইটেম বিহীন খালি টাপল তৈরি: যদি কখনো খালি টাপল তৈরি করার প্রয়োজন হয়, তখন এভাবে করতে পারি। যেমন:
```

```
>>> t = ()
>>> type(x)
<class 'tuple'>
```

ইন্ডেক্স ব্যবহার করে টাপল তৈরি: টাপলের প্রতিটি আইটেম ইনডেক্স ব্যবহার করে তৈরি করা যায়, এবং এই ইনডেক্সও ক্ষিতি () থেকে তরুণ হবে। যেমন:

```
>>> tpl = (1, 2, 3)
>>> tpl[0]
1
>>> tpl[1]
2
>>> tpl[2]
3
>>> tpl[3]
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: tuple index out of range
```

এখন, টাপল কিন্তু মিউটেবল (mutable) নয়। অর্থাৎ, এখানে যে tpl নামে একটি টাপল তৈরি করা হয়েছে যার তিনটি উপাদান যথাক্রমে 1, 2 ও 3, এখন কিন্তু আমরা চাইলে এটা আর পরিবর্তন করতে পারবো না, যেটি লিস্টের ক্ষেত্রে সম্ভব। অর্থাৎ আমরা চাইলে লিস্টের ক্ষেত্রে আইটেম পরিবর্তন করতে পারবো কিন্তু টাপলের ক্ষেত্রে না। যেমন:

```
>>> li = [1, 2, 3]
>>> li
[1, 2, 3]
>>> li[0] = 5
```

```
>>> li
[5, 2, 3]
>>> tpl = (1, 2, 3)
>>> tpl
(1, 2, 3)
>>> tpl[0] = 5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

পাইথন প্রোগ্রামিং ডিকশনারি(Dictionary)

পাইথন প্রোগ্রামিং এর ক্ষেত্রে Key Value জোড়ার জোড়ার থাকে। যেকোনো ভাটা টাইপ Key হতে পারে। তবে বেশিরভাগ ক্ষেত্রে সাধারণত নামার বা স্ট্রিং (Number or String) হিসেবে ব্যবহৃত হয়। আবার যেকোন সংখ্যা বা অবজেক্টই রেকর্ড হিসেবে ব্যবহৃত হতে পারে। ডিকশনারিকে Second Bracket এর সাহায্যে প্রকাশ করা হয় এবং Third Bracket/Square Bracket এর সাহায্যে ডিকশনারিতে Value হিসেবে এসাইন করা হয়। একটা উদাহরণ দেখলে আমরা ডিকশনারিত সম্পূর্ণ বিবরণটা বুঝতে পারবো। যেমন:

```
dict={}
dict['bitBox'] = "This is bitBox"
dict['name'] = "This is
DUET_CSE_Admission_Guide"
dict[4] = "Edition"

tinydict = {'name': 'bitBox', 'code': 4, 'dept': 'CSE'}

print(dict['bitBox']) // 'bitBox' কীর জন্য ভ্যালু প্রিন্ট করা :
print(dict['name']) // 'name' কীর জন্য ভ্যালু প্রিন্ট করা :
print(tinydict) // সম্পূর্ণ ডিকশনারি প্রিন্ট করার জন্য :
print(tinydict.keys()) // সকল কী প্রিন্ট করার জন্য :
print(tinydict.values()) // সকল ভ্যালু প্রিন্ট করার জন্য :
```

Output:
This is bitBox
This is DUET_CSE_Admission_Guide
Edition
{'name': 'bitBox', 'code': 4, 'dept': 'CSE'}
dict_keys(['name', 'code', 'dept'])
dict_values(['bitBox', 4, 'CSE'])

যদি রাখতে চান যে, ডিকশনারিটে ক্রম, সিরিয়াল বা অর্ডার টিক রাখতে চান তবে কোনো বাধা বা শর্ত নেই।

পাইথন সেট (Python Set)

সেট (Set) হচ্ছে ইউনিক আইটেমের সংগ্রহ। ইহাকে দ্বিতীয় বর্ণনা। দ্বারা প্রকাশ করা হয় এবং এর মধ্যে আইটেমগুলোকে কমা দ্বারা পৃথক করা হয়। সেট এর আইটেম বা উপাদান সমূহের ক্রম, অর্ডার টিক থাকে না। যেমন:

```
>>> a={2,4,5,1,8,6}
>>> print("a=",a)
a= {1, 2, 4, 5, 6, 8}
>>> print(type(a))
<class 'set'>
>>>
```

বীজগাছিতের মত পাইথনেও দুটি সেট এর মধ্যে ইউনিয়ন এবং ইন্টারসেকশন করা যায়। এবং সেটের ভ্যালুসমূহ ইউনিক হয়ে থাকে। যেমন:

```
>>> a={1,2,4,4,7,7,9,8}
>>> a
{1, 2, 4, 7, 8, 9}
>>> a={1,2,2,3,3,3,5,5}
>>> a
{1, 2, 3, 5}
```

সেটের উপাদানসমূহের ক্রম বা অর্ডার টিক না থাকায় ইডেক্সিং এর কোনো অর্থ নাই। সুতরাং সেট এর ক্ষেত্রে স্লাইসিং অপারেটর (Square Bracket or Third Bracket) [] কাজ করে না। যেমন:

```
>>> a={2,4,5,1,8,6};
>>> a[1]
```

Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
>>>

File in Python

File এর মাধ্যমে secondary device হতে ভাটা data access করা যায়।

Opening a File: কাজের উপর ভিত্তি করে প্রথমেই একটি File কে open করতে হয়।

ক্রান্ত নিয়ম:
hndle = open('filename', mode)
এখানে, hndle, manipulate কৃত ফাইল ধারণ করে।
filename, যে ফাইল নিয়ে কাজ করতে চাই তার নাম।
mode, এটি অপশনাল, 'r' রিচ করার জন্য এবং 'w' রাইট করার জন্য।

Example: fhand = open('bitBox.txt', 'r')

File Handle as a Sequence:

- প্রতিটি লাইন ধারাবাহিকভাবে string এর একটি sequence হিসেবে পড়ার জন্য file এর open() handle ব্যবহার করা হয়।
- for statement ব্যবহার করে ধারাবাহিকভাবে file টিকে iterate করতে পারি।

কিছু বৈশিষ্ট্য আরেকটি উত্তরসূরি ক্লাসের মধ্যে নিয়ে আসাকে ইনহেরিট্যাস বলে। যেমন:

```
class Monster:
    def __init__(self, name, color):
        self.name = name
        self.color = color
```

```
def attack(self):
    print("I am Attacking Mode...")
```

```
class Fogthing(Monster):
    def make_sound(self):
        print("Geeeeeee!!!!!!\n")
```

```
class Mournsnake(Monster):
    def make_sound(self):
        print("Hiiiiissssshhhh...\n")
```

```
fogthing = Fogthing("Fogthing", "Yellow")
fogthing.attack()
fogthing.make_sound()
```

```
mournsnake = Mournsnake("Mournsnake", "Red")
mournsnake.attack()
mournsnake.make_sound()
```

Output:
I am Attacking Mode...
Geeeeeee!!!!!!

I am Attacking Mode...

Hiiiiissssshhhh...
[নোট: কোন ক্লাসকে ইনহেরিট করার জন্য এ ক্লাসের নামটি নতুন ক্লাসের নামের পর Bracket'র মধ্যে লিখতে হয়। এখানে Monster হচ্ছে সুপারক্লাস এবং Fogthing, Mournsnake হচ্ছে সাবক্লাস।]

Polymorphism (পলিমোফিজম): বইয়ের অবজেক্ট অরিয়েন্টেড অঙ্গে আলোচনা করা আছে।

Data Abstraction (ভাটা আব্স্ট্রাকশন): বইয়ের অবজেক্ট অরিয়েন্টেড অঙ্গে আলোচনা করা আছে।

Encapsulation (এনক্যাপসুলেশন): বইয়ের অবজেক্ট অরিয়েন্টেড অঙ্গে আলোচনা করা আছে।

Properties (প্রোপার্টিস): কোন একটি মেথডের উপর property ডেকোরেটর ব্যবহার করে প্রোপার্টি ডিফাইন করা হয়। এর একটা বহুল ব্যবহার করা যায় কোন ইলেক্ট্রোল অ্যাট্রিবিউটকে রিচ-অনলি বানানোর মধ্যে কিছু ফাংশনালিটি ও বৈশিষ্ট্য শেয়ার করার একটা পদ্ধতি হচ্ছে ইনহেরিট্যাস। অর্থাৎ, একটা ক্লাসকে ইনহেরিট (অনুসরণ) করে তার ইনহেরিট্যাস অধিকারী একটা পরিকার হওয়া যাবে। যেমন:

```

class Pizza:
    def __init__(self, toppings):
        self.toppings = toppings

    @property
    def pineapple_allowed(self):
        return False

pizza = Pizza(["cheese", "tomato"])
print(pizza.pineapple_allowed)
pizza.pineapple_allowed = True

Output:
False
Traceback (most recent call last):
  File
    "C:/Users/ABC/Desktop/Python_Practice/list.py",
line 11, in <module>
    pizza.pineapple_allowed = True
AttributeError: can't set attribute

```

setter/getter ফাংশন ব্যবহার করেও প্রোপার্টি ডিফাইন করা যায়। setter ফাংশন ব্যবহার করে প্রোপার্টির ভ্যালু সেট করা যায়। আর getter ফাংশন ব্যবহার করে এই প্রোপার্টির ভ্যালু অ্যাক্সেস করা যায়। setter ডিফাইন করার জন্য প্রোপার্টির নাম এবং একটি ডট চিহ্ন দিয়ে setter কিওয়ার্ড লিখে একটি ডেকোরেটর হিসেবে নির্দেশ করতে হয়। getter ডিফাইন করার ফোরেও একই নিরাম। যেমন:

```

class Pizza:
    def __init__(self, toppings):
        self.toppings = toppings
        self._pineapple_allowed = False

    @property
    def pineapple_allowed(self):
        return self._pineapple_allowed

    @pineapple_allowed.setter
    def pineapple_allowed(self, value):
        if value:
            password = input("Enter the Password: ")
            if password == "Brownfish1":
                self._pineapple_allowed = value
            else:
                raise ValueError("Alert! Unauthorized!")

pizza = Pizza(["cheese", "tomato"])
print(pizza.pineapple_allowed)
pizza.pineapple_allowed = True
print(pizza.pineapple_allowed)

```

Output:

Enter the Password: Brownfish1

True

উপরের উদাহরণে, প্রথমে 7 নাম্বার লাইনে `pineapple_allowed` মেথডকে একটি প্রোপার্টি ডিফাইন করা হয়েছে। যার ফলে, 20 নাম্বার লাইনে এই প্রোপার্টির ভ্যালু অ্যাক্সেস (Access) করতে গেলে বজ্ঞ `pineapple_allowed` মেথডটি কল হয় এবং যা পক্ষান্তরে `self._pineapple_allowed = False` এর উপর ভিত্তি করে `False` বিটার্ন করে। এরপর, 9 নাম্বার লাইন, `@pineapple_allowed.setter` ডেকোরেটর ব্যবহার করে এই প্রোপার্টির জন্য একটি setter মেথড ডিফাইন করা হয়েছে। আর তাই, 20 নাম্বার লাইনে `pizza.pineapple_allowed = True` স্টেটমেন্ট এক্সেকিউট হবার সময় আসলে `pineapple_allowed` সেটার মেথডটি কল হচ্ছে। এই মেথডটি 14 নাম্বার লাইনে, `_pineapple_allowed` নামের প্রাইভেট ভ্যারিয়েবলের মান বদলে দেয় যার প্রমাণ পাওয়া যাবে 21 নাম্বার লাইনে নতুন করে `print(pizza.pineapple_allowed)` ডিস্ট্রিবিউটর মাধ্যমে।

Magic Method (ম্যাজিক মেথড): পাইথনে কিছু বিশেষ ধরনের বিল্ট ইন মেথড আছে যেগুলোকে ম্যাজিক মেথড বলা হয়। এগুলো চেনার খুব সহজ উপায় হচ্ছে এসের নামের দুই পাশেই দুটি করে আভারকোর সিল্ব থাকে। যেমন: `__init__()`, `__iter__()`, `__next__` ইত্যাদি। কোন ক্লাসে এই ম্যাজিক মেথড ব্যবহার করলে এবং প্রোগ্রামটিতে সেই ক্লাসের ইন্স্টান্স তৈরির সময় এই ম্যাজিক মেথডটি ব্যবহৃত্যা তাবেই কল হয় যাতে করে এর মাধ্যমে কিছু সেটআপ রিলেটেড কাজ করে নেয়া যায়। আগে-পরে দুইটা আভারকোর থাকলে তাকে ভাভার (dunder) বলে। আর পঢ়ার সময় আভারকোর আভারকোর ইনিট আভারকোর আভারকোর এভাবে না বলে (dunder init) নামে পড়া হয়।

কিছু কমন অপারেটরের ম্যাজিক মেথড।		তুলনা করার অপারেটর তালোর জন্য পাইথনে ম্যাজিক মেথড।	
সামান্যতর মেথড		ব্যাসিক মেথড	
<code>__sub__</code>	হচ্ছে - এর জন্য	<code>__lt__</code>	হচ্ছে < এর জন্য
<code>__mul__</code>	হচ্ছে * এর জন্য	<code>__le__</code>	হচ্ছে <= এর জন্য
<code>__truediv__</code>	হচ্ছে / এর জন্য	<code>__eq__</code>	হচ্ছে == এর জন্য
<code>__floordiv__</code>	হচ্ছে // এর জন্য	<code>__ne__</code>	হচ্ছে != এর জন্য
<code>__mod__</code>	হচ্ছে % এর জন্য	<code>__gt__</code>	হচ্ছে > এর জন্য
<code>__pow__</code>	হচ্ছে ** এর জন্য	<code>__ge__</code>	হচ্ছে >= এর জন্য
<code>__and__</code>	হচ্ছে & এর জন্য	এছাড়াও আরও অনেক ম্যাজিক মেথড আছে পাইথনে। যেমন: <code>__len__</code> , <code>__getitem__</code> , <code>__setitem__</code> , <code>__delitem__</code> , <code>__iter__</code> , <code>contains</code> ইত্যাদি।	
<code>__xor__</code>	হচ্ছে ^ এর জন্য		
<code>__or__</code>	হচ্ছে এর জন্য		

Program-1: যেকোনো মেসেজ প্রিন্ট করার জন্য প্রোগ্রাম লিখুন।

Code: Without new line.

```
print("bitBox")
print("DUET CSE Admission Guide")
print("Welcome to Python Programming")
```

Output:

```
bitBox
DUET CSE Admission Guide
Welcome to Python Programming
```

Note: ইনপুট নেওয়ার জন্য ইন্পুটের এর ক্ষেত্রে int এবং স্টেটিং প্রক্রিয়া এর ক্ষেত্রে float ব্যবহার করবেন।

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scripts\python.
Enter the Value of 1st Number: 5.5
Enter the Value of 2nd Number: 10.25
Enter the Value of 3rd Number: 45.75
The Total Summation is: 61.5
```

Program-4: কোনো স্ট্রিং ইনপুট হিসেবে নিয়ে সেটি প্রিন্ট করে দেখানোর জন্য প্রোগ্রাম লিখুন।

Code:

```
String= input("Which is the Best Guide for DUET CSE Admission Test?\n")
print("The Best Answer is: ", String)
String= input("Why bitBox Guide is the Best?\n")
print("The Best Answer is: ", String)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scripts\python.exe C:
Which is the Best Guide for DUET CSE Admission Test?
bitBox DUET CSE Admission Guide
The Best Answer is: bitBox DUET CSE Admission Guide
Why bitBox Guide is the Best?
Because Its Written in Very Easy Way !!
The Best Answer is: Because Its Written in Very Easy Way !!
```

Program-5: দুই বা ততোধিক সংখ্যা বা মেসেজকে নিয়ে তাদেরকে একত্র (Concatenated) প্রিন্ট করার জন্য প্রোগ্রাম লিখুন।

Code:

```
Number1= input("Enter 1st Number: ")
Number2= input("Enter 2nd Number: ")
result= Number1+Number2
print("The Concatenated Result is: ", result)
```

Output:

```
Enter 1st Number: 33
Enter 2nd Number: 55
The Concatenated Result is: 3355
দুইটার অধিক সংখ্যা মিলে।
```

Code:

```
Number1= input("Enter 1st Number: ")
Number2= input("Enter 2nd Number: ")
Number3= input("Enter 3rd Number: ")
Number4= input("Enter 4th Number: ")
result= Number1+Number2+Number3+Number4
print("The Concatenated Result is: ", result)
```

Program-3: দুই বা ততোধিক সশ্বিক সংখ্যার (Floating point Number) যোগফল নির্ণয়ের জন্য পাইথন প্রোগ্রাম লিখুন।

Code:

```
Number1= float(input("Enter the Value of 1st Number: "))
Number2= float(input("Enter the Value of 2nd Number: "))
Number3= float(input("Enter the Value of 3rd Number: "))
addition= Number1+Number2+Number3
print("The Total Summation is: ", addition)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Script
Enter 1st Number: 1
Enter 2nd Number: 22
Enter 3rd Number: 333
Enter 4th Number: 5555
The Concatenated Result is: 1223335555
তারিখ অধিক মেসেজ নিয়ে
```

Code:

```
String1= input("Enter 1st String: ")
String2= input("Enter 2nd String: ")
String3= input("Enter 3rd String: ")
String4= input("Enter 4th String: ")
String5= input("Enter 5th String: ")
result=String1+String2+String3+String4+String5
print("The Concatenated Result is:",result)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scripts\python.exe C
Enter 1st String: bitBox
Enter 2nd String: DUST
Enter 3rd String: CSE
Enter 4th String: Admission
Enter 5th String: Guide
The Concatenated Result is: bitBoxDUSTCSEAdmissionGuide
```

Program-6: দুইটি সংখ্যার বিয়োগফল, গুণফল, ভাগফল ও ভাগশেষ নির্ণয়ের জন্য প্রোগ্রাম লিখুন।

Code: দুইটি সংখ্যার বিয়োগফল

```
Number1=int(input("Enter the Value of 1st Number: "))
Number2=int(input("Enter the Value of 2nd Number: "))
Sub= Number1-Number2
print("The Final Subtraction Result is: ", Sub)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scripts\
Enter the Value of 1st Number: 420
Enter the Value of 2nd Number: 69
The Final Subtraction Result is: 351
```

Code: দুইটি সংখ্যার গুণফল

```
Number1=int(input("Enter the Value of 1st Number: "))
Number2=int(input("Enter the Value of 2nd Number: "))
mul= Number1*Number2
print("The Final Multification Result is: ", mul)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Script
Enter the Value of 1st Number: 42
Enter the Value of 2nd Number: 10
The Final Multification Result is: 420
```

Code: দুইটি সংখ্যার ভাগফল

```
Number1=int(input("Enter the Value of 1st Number: "))
Number2=int(input("Enter the Value of 2nd Number: "))
div=float(Number1)/float(Number2)
print ("The Final Divided Result is: ", div)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv
Enter the Value of 1st Number: 420
Enter the Value of 2nd Number: 10
The Final Divided Result is: 42.0
```

Code: দুইটি সংখ্যার ভাগশেষ

```
Number1=int(input("Enter the Value of 1st Number: "))
Number2=int(input("Enter the Value of 2nd Number: "))
rem=int(Number1)% int(Number2)
print ("The Final Remainder Result is: ", rem)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Sc
Enter the Value of 1st Number: 29
Enter the Value of 2nd Number: 5
The Final Remainder Result is: 4
```

Program-7: Print ASCII Value of a character.**Code:**

```
c = 'b'
print("The ASCII Value of Character " + c + " is: ",
ord(c)) # print the ASCII value of assigned character in c
Output: The ASCII Value of Character 'b' is: 98
```

তেরিয়েল ব্যবহার করে কিন্তু পাইথন প্রোগ্রাম আউটপুট

Program-1: n সংখ্যার সংখ্যার যোগফল নির্ণয়ের প্রোগ্রাম লিখুন।**Code:**

```
number= int(input("Enter a Number: "))
if number<0:
    print("Enter a Positive Number: ")
else:
    sum=0
    while(number>0):
        sum+=number
        number -= 1
    print("The Summation is: ",sum)
```

Output:

```
C:\Users\ABC\PycharmProjects\lis
Enter a Number: 10
The Summation is: 10
The Summation is: 19
The Summation is: 27
The Summation is: 34
The Summation is: 40
The Summation is: 45
The Summation is: 49
The Summation is: 52
The Summation is: 54
The Summation is: 55
```

Program-2: কত টোলো সংখ্যা ইনপুট নিয়ে তাদের গড় নির্ণয়ের প্রোগ্রাম লিখুন।

Code:

```
numbers=[0,1,2,3,4]
numbers[0]=input("Please, Enter 1st Number: ")
numbers[1]=input("Please, Enter 2nd Number: ")
numbers[2]=input("Please, Enter 3rd Number: ")
numbers[3]=input("Please, Enter 4th Number: ")
numbers[4]=input("Please, Enter 5th Number: ")
avg=(int(numbers[0])+int(numbers[1])+int(numbers[2])+int(numbers[3])+int(numbers[4]))/5
print ("The Average Value is: ",avg)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\
Please, Enter 1st Number: 5
Please, Enter 2nd Number: 29
Please, Enter 3rd Number: 55
Please, Enter 4th Number: 67
Please, Enter 5th Number: 23
The Average Value is: 35.8
```

Code: Using for Loop:

```
print("Enter the Value of n: ", end="")
n = int(input())
print("Enter " +str(n)+" Numbers: ")
numbers = []
for i in range(n):
    numbers.insert(i, int(input()))
sum = 0
for i in range(n):
    sum = sum+numbers[i]
avg = sum/n
print("\nThe Average Value is: ", avg)
```

Python Programming**Output:**

```
Enter the Value of n: 5
Enter 5 Numbers:
5
29
55
67
23
```

The Average Value is: 35.8

Code: Using While Loop:

```
print("Enter the Value of n: ", end="")
n = int(input())
print("Enter " +str(n)+" Numbers: ")
numbers = []
i = 0
while i<n:
    numbers.append(int(input()))
    i = i+1
sum = 0
i = 0
while i<n:
    sum = sum+numbers[i]
    i = i+1
avg = sum/n
print("\nThe Average Value is: ", avg)
```

Output:

```
C:\Users\ABC\PycharmProjects\li
Enter the Value of n: 5
Enter 5 Numbers:
5
29
55
67
23
```

The Average Value is = 35.8

Program-3: সেলসিয়াস তাপমাত্রাকে ফারেনহাইট তাপমাত্রার নির্ণয়ের প্রোগ্রাম লিখুন।

Conversion formula: Fahrenheit=((Celsius * (9/5))+32 = (Celsius * 1.8)+32

Code:

```
celsius = float(input("Enter temperature in Celsius
Value: "))

fahrenheit = (celsius * 1.8) + 32
print("%0.1f Celsius is Equal to %0.1f Degree
Fahrenheit." %(celsius, fahrenheit))
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scripts\python
Enter temperature in Celsius Value: 38
38.0 Celsius is Equal to 100.4 Degree Fahrenheit.
```

Same as Celsius:

Formula: Celsius = (Fahrenheit - 32) / 1.8

Program-4: কিলোমিটারকে মাইলে প্রকাশ করার জন্য প্রয়োজন নির্ধারণ।
Conversion formula: আবরা জনি, ১ কিলোমিটার = 0.621371 মাইল

Code:

```
kilometers = float(input("Enter the Value in Kilometers: "))
conversion_factor = 0.621371
miles = kilometers * conversion_factor

print("%0.2f Kilometers is Equal to= %0.2f Miles"
%(kilometers,miles))
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Script
Enter the Value in Kilometers: 5
5.00 Kilometers is Equal to= 3.11 Miles
```

Same as: kilometers = miles / conversion_factor [Try yourself]

Program-5: ফুট থেকে ইঞ্চি, গজ (Yards) & মাইলে প্রকাশ করার প্রয়োজন নির্ধারণ।

Conversion formula:

1 foot (ft)	12 Inches(in)
1 yard (yd)	3 feet (ft)
1 yard (yd)	36 Inches (in)
1 mile (mi)	1760 yards (yd)
1 mile (mi)	5280 feet (ft)

Code:

```
dis_ft = int(input("Enter the Distance in Feet: "))
dis_inches = dis_ft * 12
dis_yards = dis_ft / 3.0
dis_miles = dis_ft / 5280.0

print("The Distance in Inches is %.2f Inches." %
dis_inches)
print("The Distance in Yards is %.2f Yards." %
dis_yards)
print("The Distance in Miles is %.2f Miles." %
dis_miles)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scripts\python
Enter the Distance in Feet: 1200
The Distance in Inches is 14400.00 Inches.
The Distance in Yards is 400.00 Yards.
The Distance in Miles is 0.23 Miles.
```

Program-6: আয়তক্ষেত্রের ক্ষেত্রফল ও পরিসীমা বের করার প্রয়োজন নির্ধারণ।

Formula: Area of Rectangle = length * breadth;
Perimeter of Rectangle = 2 (length + breadth)

Code:

```
length = float(input("Enter Length of the Rectangle: "))
breadth = float(input("Enter Breadth of the Rectangle: "))

area = length * breadth
perimeter = 2 * (length + breadth)
```

```
print("\nArea of the Rectangle is= ", area)
print("Perimeter of the Rectangle is = ", perimeter)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Script
Enter Length of the Rectangle: 50
Enter Breadth of the Rectangle: 30

Area of the Rectangle is= 1500.0
Perimeter of the Rectangle is = 160.0
```

Program-7: একটি বৃত্তের ব্যাস, পরিধি এবং ক্ষেত্রফল বের করার প্রয়োজন নির্ধারণ।

Formula: Diameter of a Circle = $2r = 2 * \text{radius}$,
Circumference of a Circle = $2\pi r = 2 * \pi * \text{radius}$ and
Area of a circle are: $A = \pi r^2 = \pi * \text{radius} * \text{radius}$

Code:

```
import math
radius = float(input("Please Enter the Radius of a Circle: "))

diameter = 2 * radius
circumference = 2 * math.pi * radius
area = math.pi * radius * radius
```

```
print("\nThe Diameter of a Circle is= %.2f" %
diameter)
print("The Circumference of a Circle is= %.2f" %
circumference)
print("The Area of a Circle is = %.2f" % area)
```

Output:

```
Please Enter the Radius of a Circle: 7.5
The Diameter of a Circle is= 15.00
The Circumference of a Circle is= 47.12
The Area of a Circle is = 176.71
```

Program-8: একটি বর্গক্ষেত্রের ক্ষেত্রফল ও পরিসীমা বের করার প্রয়োজন নির্ধারণ।

Formula: Area of Square = Side * Side;
Perimeter of Square = 4 * Side

Code:

```
side = int (input ("Enter the Side of a Square: "))
Square_area = side*side
Square_perimeter = 4*side
print("Area of a Square is: %.2f" %Square_area)
print("Perimeter of a Square is: %.2f" %
%Square_perimeter)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\S
Enter the Side of a Square: 12
Area of a Square is: 144.00
Perimeter of a Square is: 48.00
```

Program-9: একটি গোলকের ক্ষেত্রফল ও আয়তন বের করার প্রয়োজন নির্ধারণ।

Formula: The area of a sphere is: $A = 4\pi r^2$; The volume of the sphere is: $V = \frac{4}{3}\pi r^3$

Code:

```
import math
radius=float(input("Enter the Radius of a Sphere: "))

sphere_area=4*math.pi*pow(radius,2)
sphere_volume=(4/3)*math.pi*pow(radius,3)

print("\nSurface Area of the Sphere is= %.2f" %
%sphere_area)
print("Volume of the Sphere is= %.2f" %
%sphere_volume)
```

Note: import math করার না করে তার সাথে pi এর মান স্বাস্থি এলাইন (PI 3.1416) করে দেওয়া যাব। তখন math.pi নিখে তার pi নির্ধারণ হবে প্রয়োজন নথে।

Output:

```
Enter the Radius of a Sphere: 7
Surface Area of the Sphere is= 615.75
Volume of the Sphere is= 1436.76
```

Program-10: সমকোণী ত্রিভুজের (Right Angle) ক্ষেত্রফল, অঞ্চল বাহ (অতিকৃজ) ও পরিসীমা নির্ধারণ প্রয়োজন নির্ধারণ।

Formula: সমকোণী ত্রিভুজের (Right Angle) ক্ষেত্রফল = $\frac{1}{2}$ (Base * Height)
পিথাগোরাসের উপপাদ্য অনুসারে পাই (অতিকৃজ): $c^2 = a^2 + b^2$; পরিসীমা = $a + b + c$

Code:

```
import math

base = float(input("Please Enter the Value of Base: "))
height = float(input("Please Enter the Value of Height: "))

Area = 0.5 * base * height
c = math.sqrt((base*base) + (height*height))
Perimeter = base + height + c
```

```
print("\nArea of a Right Angled triangle is: %.2f" %
%Area)
print("Other Side of Right Angled triangle is: %.2f" %
%c)
print("Perimeter of Right Angled triangle is: %.2f" %
%Perimeter)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scripts\p
Please Enter the Value of Base: 3
Please Enter the Value of Height: 4
```

```
Area of a Right Angled triangle is: 6.00
Other Side of Right Angled triangle is: 5.00
Perimeter of Right Angled triangle is: 12.00
```

পাইথন অপারেটের ব্যবহার করে কিছু ব্যাসিক প্রয়োজন

Program-1: দিনকে বছর, মাস, সপ্তাহ ও দিনে কনভার্ট করার প্রয়োজন নির্ধারণ।

Code: days=int(input("Please Enter Day: 395
Days to Years: 1
Days to Weeks: 56
Days to Months: 13

```
years =(int)(days / 365)
months =(int)(days / 30)
weeks =(int)(days / 7)
print("Days to
Years:",years)
print("Days to
Weeks:",weeks)
print("Days to
Months:",months)
```

Program-2: একারিথেটিক অপারেটর ব্যবহার করে কিছু আউটপুট দেখানো হলো।

Code:

```
x = 25
y = 5

print('x + y =',x+y)
print('x - y =',x-y)

print('x * y =',x*y)
print('x / y =',x/y)
print('x % y =',x%y)
print('x // y =',x//y)
print('x ** y =',x**y)
```

Output:

```
C:\Users\ABC\Pycha
x + y = 30
x - y = 20
x * y = 125
x / y = 5.0
x % y = 0
x // y = 5
x ** y = 9765625
```

Program-3: কোনো সংখ্যা জোড় (Even) নাকি বিজোড় (Odd) তা বের করার প্রয়োগ লিখুন।

Code:

```
num = int(input("Enter a number: "))
if (num % 2) == 0:
    print("{} is Even Number".format(num))
else:
    print("{} is Odd Number".format(num))
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv
Enter a number: 11
11 is Odd Number
```

Program-4: সূইচ নথার Swap (অদল-বদল) করার জন্য প্রয়োগ লিখুন।

Code:

```
a = input("Enter the Value of a: ")
b = input("Enter the Value of b: ")
```

```
temp = a
a = b
b = temp
```

```
print("\nValue of a after Swapping: {}".format(a))
print("The Value of b after Swapping: {}".format(b))
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\S
Enter the Value of a: 55
Enter the Value of b: 33
```

```
The Value of a after Swapping: 33
The Value of b after Swapping: 55
```

Program-5: কম্পিউজন অপারেটর ব্যবহার করে কিছু আউটপুট দেখানো হলো।

Code:

```
x = 25
y = 23

print('x > y is',x>y)
print('x < y is',x<y)
print('x == y is',x==y)
print('x != y is',x!=y)
print('x >= y is',x>=y)
print('x <= y is',x<=y)
```

Output:

```
C:\Users\ABC\Pyc
x > y is True
x < y is False
x == y is False
x != y is True
x >= y is True
x <= y is False
```

Program-6: It's an Output Program.

Code:

```
a = 4.5
b = 2
print (a/b)
Output: 2.0
```

Program-7: কতিশনাল অপারেটর ব্যবহার করে কিছু আউটপুট দেখানো হলো।

Code:

```
a = True
b = False
c = False
```

```
if a or b and c:
    print("DUET Admission")
else:
    print("IT Job Preparation")
```

Output: DUET Admission

Program-8: কতিশনাল অপারেটর ব্যবহার করে কিছু আউটপুট দেখানো হলো।

Code:

```
a = True
b = False
c = False
```

```
if not a or b:
    print(1)
elif not a or not b and c:
    print(2)
elif not a or b or not b and a:
    print(3)
else:
    print(4)
```

Output: 3

ব্যাখ্যা: In Python the precedence order is first NOT then AND and in last OR. So the if condition and second elif condition evaluates to False while third

elif condition is evaluated to be True resulting in 3 as output.

Program-9: It's an Output Program.

Code:

```
count = 1
def doThis():
    global count
    for i in (1, 2, 3):
        count += 1
doThis()
print (count)
```

Output: 4

ব্যাখ্যা: The variable count declared outside the function is global variable and also the count variable being referenced in the function is the same global variable defined outside of the function. So, the changes made to variable in the function are reflected to the original variable. So, the output of the program is 4.

Program-10: বিটওয়্যাইজ অপারেটর ব্যবহার করে কিছু আউটপুট দেখানো হলো।

Code:

```
a = 10
b = 6

print("a & b is: ",a&b)          # Print bitwise AND operation
print("a | b is: ",a|b)          # Print bitwise OR operation
print("\n~a is: ", ~a)           # Print bitwise NOT operation
print("a ^ b is: ",a^b)          # print bitwise XOR operation
print("\na >> 2 is: ", a>>2)   # print bitwise right shift operation
print("a << 2 is: ",a<<2)      # print bitwise left shift operation
```

Output:

```
C:\Users\ABC\PycharmProje
a & b is: 2
a | b is: 14
~a is: -11
a ^ b is: 12
a >> 2 is: 2
a << 2 is: 40
```

Program-11: অপারেটর প্রেসিডেন্সি ব্যবহার করে কিছু আউটপুট দেখানো হলো।

Code:

```
Expression = 11 + 22 * 33 # Precedence of '+' & '*'
print(Expression)
name = "bitBox" # Precedence of 'or' & 'and'
Editon = 4
if name == "bitBox" or name == "Publications" and Editon >= 6:
    print("Wow! Welcome to DUET Admission.")
else:
    print("Oh! Welcome to bitBox IT Job Aid Group.")
```

Output:

```
737
Wow! Welcome to DUET Admission.
```

Again: Change a Little bit in below code: Using and Instead of or.

Code:

```
Expression = 11 + 22 * 33 # Precedence of '+' & '*'
print(Expression)
name = "bitBox" # Precedence of 'or' & 'and'
Editon = 4
```

if name == "bitBox" and name == "Publications" and Editon >= 6:

```
    print("Wow! Welcome to DUET Admission.")
else:
    print("Oh! Welcome to bitBox IT Job Aid Group.")
```

Output:

```
737
Oh! Welcome to bitBox IT Job Aid Group.
```

Program-12: অপারেটর এসেসিগ্রেডেটিভি ব্যবহার করে কিছু আউটপুট দেখানো হলো।

Code:

```
print("100 / 10 * 10 the Result is: ",100 / 10 * 10)
# Left-right associativity;
# Note: 100 / 10 * 10 is calculated as (100 / 10) * 10
# and not as 100 / (10 * 10)
print("5 - 2 + 3 the Result is: ",5 - 2 + 3)
```

Left-right associativity; (Note: 5 - 2 + 3 is calculated as (5 - 2) + 3 and not as 5 - (2 + 3))
print("5 - (2 + 3) the Result is: ",5 - (2 + 3))

```
# left-right associativity
print("2 ** 3 ** 2 the Result is: ", 2 ** 3 ** 2)
# right-left associativity; (Note: 2 ** 3 ** 2 is
calculated as 2 ** (3 ** 2) and not as (2 ** 3) ** 2
Output:
C:\Users\ABC\PycharmProjects\list\venv\Scr
100 / 10 * 10 the Result is: 10.0
5 - 2 + 3 the Result is: 6
5 - (2 + 3) the Result is: 0
2 ** 3 ** 2 the Result is: 512
```

বিজ্ঞ প্রক্রিয়া স্টেটমেন্ট ব্যবহার করে কিছু বাসিন্দা প্রোগ্ৰাম লিখুন।

Program-1: তিনটি সংখ্যার মধ্যে বৃহত্তম সংখ্যা বের করার প্রোগ্ৰাম লিখুন।

Code:

```
Number1 = float(input("Enter first Number: "))
Number2 = float(input("Enter second Number: "))
Number3 = float(input("Enter third Number: "))

if (Number1 >= Number2) and (Number1 >= Number3):
    largest = Number1
elif (Number2 >= Number1) and (Number2 >= Number3):
    largest = Number2
else:
    largest = Number3

print("The Largest Number is", largest)
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scr
Enter first Number: 33
Enter second Number: 55
Enter third Number: 44
The Largest Number is 55.0
```

Program-2: কোনো সংখ্যা পজিটিভ নাকি নেগেটিভ আ বের করার প্রোগ্ৰাম লিখুন।

Code:

```
num = float(input("Enter any number: "))
if num > 0:
    print("Your Entered Number is Positive.")
elif num == 0:
    print("Oh!! Your Entered Number Zero.")
else:
    print("Your Entered Number is Negative.")
```

Output:

```
Enter any number: 55
Your Entered Number is Positive.
```

আরেকটা ইনপুট দিয়ে চেক করলে:

Enter any number: -55
Your Entered Number is Negative.

Program-3: শতসাপেক্ষ বিষমবাহু ত্রিভুজের অর্ধ-পরিসীমা এবং ক্ষেত্ৰফল লিখুন।
Formula: অর্ধ-পরিসীমা (s) = $(a+b+c)/2$; ক্ষেত্ৰফল (Area) = $\sqrt{(s-a)(s-b)(s-c)}$

Code:

```
import math
a=int(input("Enter the Value of Side1(a): "))
b=int(input("Enter the Value of Side2(b): "))
c=int(input("Enter the Value of Side2(c): "))
if((a+b)>c and (b+c)>a and (c+a)>b):
    s=(a+b+c)/2
    Area=math.sqrt(s*(s-a)*(s-b)*(s-c))
    print("\nThe Perimeter of Triangle is: ",s)
    print("Area of the Traiangle is: %0.3f" %Area)
else:
    print("\nThe Triangle is Not Possible.")
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scr
Enter the Value of Side1(a): 12
Enter the Value of Side2(b): 13
Enter the Value of Side2(c): 14

The Perimeter of Triangle is: 19.5
Area of the Traiangle is: 72.308
```

Again Output:

```
C:\Users\ABC\PycharmProjects\list\venv\Scr
Enter the Value of Side1(a): 12
Enter the Value of Side2(b): 13
Enter the Value of Side2(c): 25

The Triangle is Not Possible.
```

Program-4: কোনো ইংৰেজি বহু লিপি ইয়াৰ কিনা তা চেক কৰার জন্য প্রোগ্ৰাম লিখুন।

Code:

```
n= int(input("Enter Any Year for Check: "))
if(n%4==0):
    if(n%100==0):
        if(n%400==0):
            print("Wow!! It is a Leap Year.")
        else:
            print("Oh!! It is Not a Leap Year.")
    else:
        print("Wow!! It is a Leap Year.")
else:
    print("Wow!! It is a Leap Year.")
else:
    print("Oh!! It is Not a Leap Year.")
```

Output:

```
Enter Any Year for Check: 2020
Wow!! It is a Leap Year.
```

Again Check:

```
Enter Any Year for Check: 2021
Oh!! It is Not a Leap Year.
```

Program-5: একটি নামৰ প্রাইম কিনা তা চেক কৰার জন্য প্রোগ্ৰাম লিখুন।

Code:

```
number = int(input("Enter Any Number: "))
flag = False
if number > 1:
    for i in range(2, number):
        if (number % i) == 0:
            flag = True
            break
    if flag:
        print(number, "is Not a Prime Number.")
    else:
        print(number, "is a Prime Number.")
```

Output:

```
Enter Any Number: 17
17 is a Prime Number.
Again Check:
Enter Any Number: 20
20 is Not a Prime Number.
```

Program-6: একটি সংখ্যার ফ্যাক্টোরিয়াল মান বের কৰার জন্য প্রোগ্ৰাম লিখুন।

Code:

```
number = int(input("Enter Any Number: "))
factorial = 1
if number < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif number == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1,number + 1):
        factorial = factorial*i
    print("The Factorial of",number,"is: ",factorial)
```

Output:

```
Enter Any Number: 7
The Factorial of 7 is: 5040
```

Program-7: ফিবোনাক্সি (Fibonacci) সিরিজ দেখার জন্য প্রোগ্ৰাম লিখুন।

Code:

nterms = int(input("How Many terms You Want? "))
number1, number2 = 0, 1
count = 0
if nterms <= 0:
 print("Please Enter a Positive Integer: ")
elif nterms == 1:
 print("Fibonacci Sequence Upto",nterms,:")
 print(number1)
else:

```
print("Fibonacci Fibonacci Sequence is:")
while count < nterms:
    print(number1, end="," )
    nth = number1 + number2
    number1 = number2
    number2 = nth
    count += 1
```

Output:

```
How Many terms You Want? 1
Fibonacci Sequence Upto 1:
0
```

Again Check:

```
How Many terms You Want? 15
Fibonacci Fibonacci Sequence is:
0,1,1,2,3,5,8,13,21,34,55,89,144,233,377
```

Program-8: Find the Sum of First N Natural Numbers. Or Find Sum of the Series: $1+2+3+4+5+\dots+n^{\text{th}}$

Code:

```
n=int(input("Enter the Number of Terms: "))
sum = 0
while(n > 0):
    sum=sum+n
    n=n-1
print("The Summation of nth Natural Numbers is: ",sum)
```

Output:

```
Enter the Number of Terms: 10
The Summation of nth Natural Numbers is: 55
```

Program-9: Find the Sum of the Series: $1 + 1/2 + 1/3 + \dots + 1/N$

Code:

```
n=int(input("Enter the Number of terms: "))
sum=0
for i in range(1,n+1):
    sum=sum+(1/i)
print("The Summation of Series is: ",round(sum,2))
```

Output:

Enter the Number of terms: 10
The Summation of Series is: 2.93

Program-10: Find the Sum of the Series: $1 + 1/1! + 1/2! + \dots + 1/n!$ **Code:**

```
import math
n=int(input("Enter the Number of terms: "))
sum=1
for i in range(1,n+1):
    sum=sum+(1/math.factorial(i))
print("The Summation of Series is: ",round(sum,2))
```

Output:

Enter the Number of terms: 5
The Summation of Series is: 2.72

Program-11: একটি নাম্বার পারফেক্ট কিনা তা দেখার জন্য প্রোগ্ৰাম লিখুন।**Code:**

```
num = int(input("Enter Any Number: "))
sum = 0
for i in range(1, num):
    if(num % i == 0):
        sum = sum + i
if (sum == num):
    print("Entered Number is a Perfect Number!")
else:
    print("Entered Number is not a Perfect Number!")
```

Output:

Enter Any Number: 6
Entered Number is a Perfect Number!
Again Check:
Enter Any Number: 15
Entered Number is not a Perfect Number!

Program-12: একটি নাম্বার সংজীব নাম্বার কিনা তা দেখার জন্য প্রোগ্ৰাম লিখুন।**Code:**

```
sum=0
num=int(input("Enter Any Number: "))
temp=num
while(num):
    i=1
    f=1
    rem=num%10
    while(i<=rem):
        f=f*i
        i=i+1
    sum=sum+f
    num=num//10
```

```
num=num//10
if(sum==temp):
    print("Entered Number is a Strong Number!!")
else:
    print("Entered Number is not a Strong Number!!")
```

Output:

Enter Any Number: 145
Entered Number is a Strong Number!!
Again Check:
Enter Any Number: 320
Entered Number is not a Strong Number!!

Program-13: Find the LCM of Two Numbers. (সূত্ৰ সংক্ষেপে গ.সা.গ বেৱে কৰাৰ জন্য প্ৰোগ্ৰাম লিখুন।)**Code:**

```
a=int(input("Enter the First Number: "))
b=int(input("Enter the Second Number: "))
if(a>b):
    min1=a
else:
    min1=b
while(1):
    if(min1%a==0 and min1%b==0):
        print("Least Common Multiple (LCM) is:",min1)
        break
    min1=min1+1
```

Output:

Enter the First Number: 15
Enter the Second Number: 25
Least Common Multiple (LCM) is: 75

Program-14: Find the GCD of Two Numbers. (সূত্ৰ সংক্ষেপে গ.সা.গ বেৱে কৰাৰ জন্য প্ৰোগ্ৰাম লিখুন।)**Code:**

```
def computeGCD(x, y):
    while(y):
        x, y = y, x % y
    return x
a = 60
b = 48
```

```
print ("The gcd of 60 and 48 is : ",end="")
print (computeGCD(60,48))
```

Output:

Enter the First Number: 60
Enter the Second Number: 40
The gcd of 60 and 48 is : 12

Program-15: Program to check Armstrong Number. Formula: Input: 153; Output: Yes 153 is an Armstrong number: $1^3 + 5^3 + 3^3 = 153$

Input: 120; Output: No
120 is not a Armstrong number: $1^3 + 2^3 + 0^3 = 9$

Code:

```
num = int(input("Enter any Number: "))
sum = 0
```

```
temp = num # find the sum of the cube of each digit
while temp > 0:
    digit = temp % 10
    sum += digit ** 3
    temp //= 10

if num == sum: # display the result
    print(num,"is an Armstrong Number.")
else:
    print(num,"is Not an Armstrong Number.")
```

Output:

Enter any Number: 153
153 is an Armstrong Number.

Again Check:

Enter any Number: 120
120 is Not an Armstrong Number.

Program-16: বিদ্যুত সমীকৰণের মূলধৰ্য বেৱে কৰাৰ জন্য প্ৰোগ্ৰাম লিখুন।

Formula: $ax^2 + bx + c = 0$, where a, b and c are real numbers and $a \neq 0$

If discriminant > 0 , then Two Distinct Real Roots exists for this equation

$$\text{Root1} = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$\text{Root2} = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

If discriminant $= 0$, Two Equal and Real Roots exists.

$$\text{Root1} = \text{Root2} = \frac{-b}{2a}$$

and if discriminant < 0 , Two Distinct Complex Roots exists.

$$\text{Root1} = \frac{-b + i\sqrt{-(b^2 - 4ac)}}{2a}$$

$$\text{Root2} = \frac{-b - i\sqrt{-(b^2 - 4ac)}}{2a}$$

Code:

```
import math
```

```
a = int(input("Enter the Value of a: "))
b = int(input("Enter the Value of b: "))
c = int(input("Enter the Value of c: "))
```

```
d = (b * b) - (4 * a * c)
```

```
if(d > 0):
```

```
root1 = (-b + math.sqrt(d)) / (2 * a)
root2 = (-b - math.sqrt(d)) / (2 * a)
print("Two Distinct Real Roots Exists: \nroot1 = %.2f and root2 = %.2f" %(root1, root2))
```

```
elif(d == 0):
```

```
root1 = root2 = -b / (2 * a)
print("Two Equal and Real Roots Exists: \nroot1 = %.2f and root2 = %.2f" %(root1, root2))
```

```
elif(d < 0):
```

```
root1 = root2 = -b / (2 * a)
imaginary = math.sqrt(-d) / (2 * a)
print("Two Distinct Complex Roots Exists: \nroot1 = %.2f+%.2fj and root2 = %.2f-%.2fj" %(root1, imaginary, root2, imaginary))
```

Output:

Enter the Value of a: 25
Enter the Value of b: 30
Enter the Value of c: -15
Two Distinct Real Roots Exists:
root1 = -29.02 and root2 = -30.98

Again Check:

Enter the Value of a: 8
Enter the Value of b: 16
Enter the Value of c: 8
Two Equal and Real Roots Exists:
root1 = -1.00 and root2 = -1.00

Again Check:

Enter the Value of a: 10
Enter the Value of b: 12
Enter the Value of c: 18
Two Distinct Complex Roots Exists:
root1 = -0.60+1.20 and root2 = -0.60-1.20

Python Programming

Program-17: বিদ্যুৎ বিল ক্যালকুলেশন করার জন্য প্রোগ্রাম লিখুন।
Code:

```
units = int(input("Enter Number of Units You Consumed: "))

if(units < 50):
    amount = units * 3.75
    surcharge = 25
elif(units <= 100):
    amount = 150 + ((units - 50) * 3.50)
    surcharge = 50
elif(units <= 200):
    amount = 150 + 162.50 + ((units - 100) * 3.00)
    surcharge = 75
else:
    amount = 150 + 162.50 + 526 + ((units - 200) * 5.00)
    surcharge = 100

total = amount + surcharge
print("The Total Electricity Bill is: %.2f taka" %total)
```

Output:

```
Enter Number of Units You Consumed: 50
The Total Electricity Bill is: 200.00 taka
Again Check:
Enter Number of Units You Consumed: 100
The Total Electricity Bill is: 375.00 taka
Again Check:
Enter Number of Units You Consumed: 200
The Total Electricity Bill is: 687.50 taka
Again Check:
Enter Number of Units You Consumed: 350
The Total Electricity Bill is: 1688.50 taka
```

Program-18: Find Sum and Average of N Natural Numbers using For Loop.

Code:

```
number = int(input("Enter Any Number of terms: "))
total = 0

for value in range(1, number + 1):
    total = total + value
average = total / number

print("Summation of Natural Numbers from 1 to {} = {}".format(number, total))
print("Average of Natural Numbers from 1 to {} = {}".format(number, average))
```

Output:

```
Enter Any Number of terms: 20
Summation of Natural Numbers from 1 to 20 = 210
```

Average of Natural Numbers from 1 to 20 = 10.5

Program-19: Print Floyd's Triangle using For Loop.

Code:

```
rows = int(input("Please Enter the Number of Rows: "))
number = 1

print("The Final Floyd's Triangle is:")
for i in range(1, rows + 1):
    for j in range(1, i + 1):
        print(number, end=' ')
        number = number + 1
    print()
```

Output:

```
C:\Users\ABC\PycharmProjects\list\venv>
Please Enter the Number of Rows: 5
The Final Floyd's Triangle is:
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

Program-20: Python Program to calculate Sum of Series $1^2+2^2+3^2+\dots+n^2$

Formula: $(n(n+1)(2n+1))/6$

Code:

```
number = int(input("Enter any Positive Number: "))
total = 0
total = (number * (number + 1) * (2 * number + 1)) / 6
```

```
for i in range(1, number + 1):
    if(i != number):
        print("%d^2 + %d, end=' ')
    else:
        print("{}^2 = {}".format(i, total))
```

Output:

```
Enter any Positive Number: 5
1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55.0
```

Program for Using Function and Some Outputs Program

* Write a program to print nth Catalan number.

```
def catalan_number(num):
    if num <=1:
        return 1
    res_num = 0
    for i in range(num):
```

Python Programming

```
res_num += catalan_number(i) *
catalan_number(num-i-1)
return res_num
```

```
for n in range(10):
    print(catalan_number(n))
```

Output: 1 1 2 5 14 42 132 429 1430 4862

Program-1: Write a program which can compute the factorial of a given numbers. The results should be printed in a comma-separated sequence on a single line.

Code:

```
def fact(x):
    if x == 0:
        return 1
    return x * fact(x - 1)
```

```
x=int(input("Enter any Number: "))
print("The Factorial Value is: ",fact(x))
```

Output:

```
Enter any Number: 5
The Factorial Value is: 120
```

Program-2: With a given integral number n, write a program to generate a dictionary that contains (i, i^i) such that is an integral number between 1 and n (both included), and then the program should print the dictionary.

Code:

```
n=int(input("Enter any Integer You Want: "))
d=dict()
for i in range(1,n+1):
    d[i]=i**i
print("The Dictionary is: ",d)
```

Output:

```
Enter any Integer You Want: 5
The Dictionary is: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Program-3: Define a function which can compute the sum of two numbers.

Code:

```
def SumFunction(number1, number2):
    return number1+number2
```

```
print("The Summation is: ",SumFunction(7,9))
```

Output:

```
The Summation is: 16
```

Program-4: Define a function that can convert a integer into a string and print it in console.

Code:

```
def printValue(n):
    print("String Value is: ", str(n))
printValue(7)
```

Output: String Value is: 7

Program-5: Define a function that can receive two integral numbers in string form and compute their sum and then print it in console.

Code:

```
def printValue(s1,s2):
    print("The Summation is: ",int(s1)+int(s2))
printValue("7","8")
```

Output: The Summation is: 15

Program-6: Define a function that can accept two strings as input and concatenate them and then print it in console.

Code:

```
def printValue(s1,s2):
    print("The Concatenated Value is: ",s1+s2)
printValue("7","8") #Output Should be: 34
```

Output: The Concatenated Value is: 78

Program-7: Define a function that can accept an integer number as input and print the "It is an even number" if the number is even, otherwise print "It is an odd number".

Code:

```
def checkValue(num):
    if num%2 == 0:
        print("It is an Even Number.")
    else:
        print("It is an Odd Number.")

checkValue(420) #If checkValue(69) is given, Output Shoud be Odd
```

Output: It is an Even Number.

Program-8: With a given tuple (1,2,3,4,5,6,7,8,9,10,11,12), write a program to print the first half values in one line and the last half values in one line.

Code:

```
tp=(1,2,3,4,5,6,7,8,9,10,11,12)
tp1=tp[:6]
tp2=tp[6:]
print(tp1)
print(tp2)
```

Output:

```
(1, 2, 3, 4, 5, 6)
(7, 8, 9, 10, 11, 12)
```

Program-9: Find Simple Interest Using Function.

Formula: Simple Interest (I)= pnr [Here, P=Principal, n=time duration, r=interest rate]

Code:

```
def si(p, n, r):
    print("Enter the Principal value: ", p)
    print("The Time Period is: ", n)
    print("The Rate of Interest is: ", r)

    Simple_Interest = (p * n * r) / 100

    print("The Simple Interest is: ", Simple_Interest)
    return Simple_Interest

si(5000, 3, 8)
```

Output:

```
Enter the Principal value: 5000
The Time Period is: 3
The Rate of Interest is: 8
The Simple Interest is: 1200.0
```

Program-10: Calculate Sum and Average of N Natural Numbers using Functions.

Code:

```
def sum_and_avg_of_natural_numbers(num):
    if (num == 0):
        return num
    else:
        return (num * (num + 1)) / 2

number = int(input("Please Enter any Number: "))

total = sum_and_avg_of_natural_numbers(number)
```

average = total / number

```
print("The Sum of Natural Numbers from 1 to {0} = {1}".format(number, total))
print("Average of Natural Numbers from 1 to {0} = {1}".format(number, average))
```

Output:

```
Please Enter any Number: 10
The Sum of Natural Numbers from 1 to 10 = 55.0
Average of Natural Numbers from 1 to 10 = 5.5
```

Digital Electronics and Logic Design

[Syllabus: BPSC CS:

Digital System: Number system: binary, octal, hexadecimal and BCD. Data representation.

Logic gates and Boolean algebra: Combinational circuits. Circuit design using logic gates.

Circuit and expression minimization: Karnaugh map and Quine-McCluskey. Basic flip-flops (FF), Design of half and full adder. Basic counters and register. Basic decoders, encoders, multiplexers and demultiplexers. ADC and DAC circuits. PLA design, Pulse mode and fundamental mode logic, Pulse & switching units, Newtrivibrations,

Digital LC: DTL, TTL, III, CMOS MOS gates, Memory system, LED, LCD applications of Op-Amps. Cooperators.

NTRCA CS: Number systems and codes, Boolean algebra, De Morgan's theorems, logic gates and their truth tables, combinational logic circuits, karnaugh map method, decoder, encoder, multiplexed, and demultiplexer, Flip flop, Asynchronous and synchronous counters, D/A converter circuitry, A/D converter circuitry.]