Ehindistudy.com

Python क्या है? - What is Python in Hindi?

पाइथन)**Python)** एक High Level कंप्यूटर Programming Language है जिसका इस्तेमाल **AI**, **मशीन Learning**, Data Analytic, Application Program आदि में किया जाता है यह एक .Object Oriented Language है.

पाइथन एक फ्री और Open Source Language है इसके कोड का इस्तेमाल किसी भी **Operating System** में किया जा सकता है.

पाइथन का इतिहास (History of Python in Hindi)

पाइथन भाषा को 1991 में Guido Van Rossum ने बनाया था. इसका सबसे पहला Release Version 0.9.0 था. इसमें Classes, List, String आदि Concept का इस्तेमाल किया गया था.

वैसे Guido Van Rossum ने पाइथन पर 1980 से ही काम करना शुरू कर दिया था. Python की शुरुवात National, Institute For Mathematic And Computer Science Netherlands में हुई थी.

धीरे – धीरे पाइथन का इस्तेमाल बढ़ने लगा और यह एक Popular Language बन गयी. Python के समय समय पर Version अपडेट होते हैं जिसके बारे में हम आगे जानेंगे.

बहुत सारे लोगों को लगता है कि पाइथन का नाम अजगर सांप के आधार पर रखा गया है पर यह गलत है. पाइथन Language का नाम एक ब्रिटिश टीवी शो “Monty’s Python Flying Circus” के आधार पर रखा गया है.

पाइथन के संस्करण (Python Version in Hindi)

पाइथन के समय – समय पर नए – नए Version आते रहते हैं. पाइथन के अभी तक 3 Version Launch हो चुके हैं. इसका Latest Version 3.9 है. पाइथन के अभी तक सभी लांच किये गए Version और उनके Launch Year को नीचे सारणी में दिया गया है.

Python Version	Launch Year
Python 1.0	**1994**
Python 1.2	1995
Python 1.3	1995
Python 1.4	1996
Python 1.5	1997
Python 1.6	2000
Python 2.0	**2000**
Python 2.1	2001
Python 2.2	2001
Python 2.3	2003
Python 2.4	2004
Python 2.5	2006
Python 2.6	2008
Python 2.7	2010
Python 3.0	**2008**
Python 3.1	2009

Python Version	Launch Year
Python 3.2	2011
Python 3.3	2012
Python 3.4	2014
Python 3.5	2015
Python 3.6	2016
Python 3.7	2018
Python 3.8	2019
Python 3.9	2020

पाइथन की विशेषताएं (Feature of Python in Hindi)

पाइथन Language की अनेक विशेशताएँ हैं. जिनके बारे में एक Python सीखने वाले व्यक्ति को पता होना जरुरी है.

1 – Easy to Code (कोड के लिए आसान)

पाइथन का Syntax बहुत आसान होता है. इसका Code लिखना भी बहुत आसान है. Python की कोडिंग अन्य Programming Language की तुलना में आसान है जैसे कि **C Language**, **C++ Language**, **Java** आदि.

2 – Free and Open Source (नि: शुल्क और खुला स्रोत)

पाइथन एक Open Source Programming Language है. इसका इस्तेमाल आप Free में कर सकते हैं. इसके Source Code को डाउनलोड कर सकते हैं, इस्तेमाल कर सकते है और दूसरों के साथ शेयर भी कर सकते हैं.

3 – Object-Oriented Language (ऑब्जेक्ट उन्मुख भाषा)

पाइथन की एक महत्वपूर्ण विशेषता यह है कि Object-Oriented Programming Language है. यह Classes, Object Encapsulation जैसे Concept को Support करता है.

4 – Portable Language (पोर्टेबल भाषा)

पाइथन एक पोर्टेबल Language है. जैसे अगर आपके पास पाइथन का कोड Window में है तो आप Same Code का इस्तेमाल Linux, Unix, Mac आदि में भी कर सकते हैं इसे बदलने की आवश्यकता नहीं है.

5 – High-Level Language (उच्च स्तरीय भाषा)

पाइथन एक उच्च स्तरीय प्रोग्रामिंग भाषा है. जब पाइथन के प्रोग्राम को लिखते हैं तो इसके System Architecture को याद रखने की जरुरत नहीं होती है और न ही इसकी मेमोरी को मैनेज करने की जरुरत होती है.

6 – Integrated Language (एकीकृत भाषा)

पाइथन एक Integrated Language है. इसके कोड को अन्य भाषओं जैसे कि C Language, C++ Language के साथ Integrateकर सकते हैं.

7 – Extensible Feature (विस्तारणीय सुविधा)

पाइथन एक Extensible Language है. पाइथन के कोड को अन्य भाषओं के साथ भी लिख सकते हैं और उन्हें Compile भी कर सकते हैं.

8 – Graphic User Interface (ग्राफिक यूजर इंटरफेस)

पाइथन में Graphic User Interface बनाने के लिए बहुत सारे Module उपलब्ध हैं. जैसे कि PyQt4, PyQt5 आदि.

पाइथन के उपयोग (Uses of Python in Hindi)

अभी तक आप अच्छे से समझ गए होंगे कि **Python** क्या है अब इसके उपयोगीं के बारे में जानते हैं.

पाइथन एक लोकप्रिय Language है. दुनिया की बड़ी – बड़ी Tech Company पाइथन का इस्तेमाल करती है. पाइथन का इस्तेमाल निम्न कामों में किया जाता है –

- **Artificial Intelligence (AI) And Machine Learning –** पाइथन एक सरल भाषा है इसका इस्तेमाल बहुत सारी मशीन Learning और AI Project में किया जाता है. मशीन Learning के लिए पाइथन एक लोकप्रिय भाषा है.
- **Programming Application** – पाइथन का इस्तेमाल करके अनेक प्रकार के एप्लीकेशन को Program किया जा सकता है.
- **Data Analytic** में भी पाइथन भाषा का इस्तेमाल किया जाता है.
- **Data Visualization** – पाइथन भाषा की Flexibility के कारण इसका उपयोग Data Visualization में भी किया जाता है.
- **Web Development**– Web Development में पाइथन का इस्तेमाल Backend को मजबूती देने के लिए किया जाता है.
- **Desktop Application** – विभिन्न प्रकार के Desktop एप्लीकेशन बनाने के लिए भी Python का इस्तेमाल होता है.

पाइथन के फायदे और लाभ (Advantage of Python Language in Hindi)

- पाइथन एक उच्च स्तरीय प्रोग्रामिंग भाषा है.
- इसके कोड को लिखना, पढना और समझना आसान है.
- पाइथन का इस्तेमाल आप फ्री में कर सकते हैं.
- पाइथन एक Open Source प्रोग्रामिंग भाषा है.
- पाइथन एक गतिशील प्रकार की Language है यह चर के प्रकार को तब तक नहीं समझ पाती है जब तक कि Code को Run नहीं किया जाता है.
- यह एक Portable Language है. पाइथन के Same Source Code में बिना किसी बदलाव के किसी भी Operating System में इस्तेमाल किया जा सकता है.

पाइथन के नुकसान (Disadvantage of Python Language)

जहाँ एक ओर पाइथन के बहुत सारे फायदे हैं वहीँ दूसरी ओर इसके कुछ नुकसान भी है. पाइथन के कुछ नुकसान निम्न प्रकार से हैं –

- गतिशील Language होने के कारण पाइथन की Speed धीमी होती है.
- पाइथन Language बहुत अधिक Memory का इस्तेमाल करती है.
- पाइथन को मुख्यतः Server Side Programming Language के रूप में इस्तेमाल किया जाता है. इसलिए हम पाइथन को Client Side और Mobile Application में नहीं देख सकते हैं.
- चूँकि पाइथन गतिशील रूप से Type की जाने वाली भाषा है इसलिए इसमें चर का डेटा प्रकार कभी भी बदल सकता है जिससे Runtime में गलती होने की सम्भावन बही रहती है.

पाइथन लैंग्वेज कैसे सीखें (Learn Python Language In Hindi)

जैसा कि हमने लेख में जाना कि पाइथन एक High Level Programming Language है और इसका इस्तेमाल बड़े – बड़े कामों में होता है, इसलिए पाइथन को सीखना आपके लिए फायदेमंद साबित हो सकता है.

पाइथन के कोड को लिखना भी आसान है और सीखना भी. आप आसानी से 1 – 2 महीनों में Python Language को सीख सकते हैं. पाइथन को आप ऑनलाइन फ्री में सीख सकते हैं और Offline Coaching लेकर भी सीख सकते हैं.

वेरिएबल क्या है?

Variable जब create किया जाता है तब interpreter द्वारा value को store करने के लिए memory location आरक्षित की जाती है |

Variable पर कोई भी data type की value store की जा सकती है | जैसे कि, Number, string, list, tuple, dictionary

Assigning Value to Variable

Python में declaration की जरुरत नहीं होती है | जब variable पर value assign होती है तब automatically declaration होता है |

declaration न होने के कारण Python में variable की default value नहीं होती है |

For Example,

```
a = 5       #Number

b = "Hello" #string

c = [2, 5, 9] #list


print(a, b, c)
```

Output :

```
5 Hello [2, 5, 9]
```

Changing Variable's Value

Python में variable की value change या re-assign की जा सकती है |

Source Code :

```
a = 5

print(a)

a = "Hello"

print(a)

a = [4, 5, 8]
```

```
print(a)
```

Output :

```
5

Hello

[4, 5, 8]
```

Assigning Single Value to Multiple Variables

Python में एक ही value एक से ज्यादा variables पर assign की जा सकती है |

Source Code :

```
a = b = c = d = "Hello"

print(a)

print(b)

print(c)

print(d)
```

Output :

```
Hello

Hello
```

```
Hello

Hello
```

Assigning Value to Variable according to order

Python में क्रमनुसार variable पर value store की जाती है |

Example पर एक ही memory location multiple variables और उनकी values assign की गयी है |

Source Code :

```
a, b, c = 1, 'H', [1, 2]

print(a)

print(b)

print(c)
```

Output :

```
1

H

[1, 2]
```

Variables Concatenation

Python में एक ही data types के variables concatenate किय जा सकते है |

Example पर str() function का इस्तेमाल object को integer से string में convert करने के लिए किया गया है |

Source Code :

```
a = 1

b = 2

print(a + b)

print(str(a) + str(b))

c = "Hello"

print(str(a) + c)
```

Output :

```
3

12

1Hello
```

Python - Types of Variable

Python में variable के दो प्रकार है |

1. Local Variables

2. Global Variables

1. Local Variables

Local Variables; functions के अन्दर होते है | उनकी visibility सिर्फ function के अन्दर होती है, जब वो function के बाहर आते है तब destroy हो जाते है |

Source Code :

```
def func():
    a = 5 #local variable
    print(a)


func()
print(a)
```

Output :

```
5
Traceback (most recent call last):
    print(a)
NameError: name 'a' is not defined
```

2. Global Variables

Global Variables; function के बाहर होते है | उनकी visibility function के अन्दर और बाहर होती है | उनका scope पूरे program पर होता है |

Source Code :

```
a = 10 #global variable

def func():

    print(a)


func()

print(a)
```

Output :

```
10

10
```

Example पर local और global ये दोनों variables declared किये गए है | function के बाहर का variable lobal है और अन्दर का variable local है | global variable का scope function के अन्दर और बाहर होता है लेकिन function के अन्दर अलग से variable declaration होने के कारण func() call करते ही variable की value change हो जाती है |

Source Code :

```
a = 10 #global variable

def func():

    a = 5 #local variable

    print(a)


func() #print local

print(a) #print global
```

Output :

```
5

10
```

PYTHON DATA TYPES in hindi

आज हम इस पोस्ट में python data types के बारें में पढेंगे तो चलिए शुरू करते है:-

Data जो की हमारे सिस्टम की memory मे स्टोर होती है वे कई प्रकार कई होती है जैसे की Number , Char आदि | जिन्हे हम डाटा टाइप कहते है | हर programming language मे डाटा टाइप होते है |

इस प्रकार पाइथन मे भी कुछ डाटा टाइप (python data types) है जो की इस प्रकार है |

- **Boolean**
- **Number**
- **String**
- **List**

- **Tuples**
- **Dictionaries**

Boolean

Boolean सभी प्रोग्रामिंग लैंग्वेज का एक कॉमन डाटा टाइप है | बूलियन दो वैल्यू को return करता है|

- True
- False

उदहारण :-

>>> a= (“learn python”)
>>> len(a)
आउटपुट :- 12
>>> len(a)==12
आउटपुट :-True
>>> len(a)!=12
आउटपुट :-False
len () एक method है | जिसका उपयोग String की length पता करने के लिए किया जाता है |

Number

पाइथन numbers डाटा टाइप numeric value को स्टोर करता है |
पाइथन 2.x मे 4 (चार) built-in-data_type (int, long, float, complex) थे | जो की number data type को represent करते है | लेकिन पाइथन 3.x मे long type को हटा कर उसे int type मे extend करके इसकी length को unlimited कर दिया गया |
Number data type भी तीन प्रकार के होते है |

- int data type
- float data type
- complex data type

1:- int data type

int data type मे सिंपल न्यूमेरिक वैल्यू होती है | ये वैल्यू नेगेटिव तथा पॉजिटिव दोनों प्रकार की होती है |

उदहारण :-

a= 14582

print("positive int no .= ", a)

आउटपुट :- *positive int no . =* ***14582***

b= -1345

print("negative int no .= ", b)

आउटपुट :- *negative int no . =* ***-1345***

2:- Float data type

Float data type मे दशमलव वैल्यू होती है | ये भी नेगेटिव तथा पॉजिटिव दोनों प्रकार की होती है |

उदहारण :-

a= 2525.5455

print("positive float no . =", a)

आउटपुट :- *positive float no . = 2525.5455*

b= -2165.13155

print("negative float no . =", b)

आउटपुट :- *negative float no . = -2165.13155*

3:- complex data type

complex data type मे दो part होते है | पहला real part दूसरा imaginary part होता है |

उदहारण :-

a=3+5j

print ("complex no = ", a)

आउटपुट :- *complex no = (3+5j)*

3=real part & 5j = imaginary part

String

पाइथन string characters या text का sequence होता है | यह textual डाटा को handle करता है | string को single(' ') या double (" ") quotes के अंदर लिख़ते है |

पाइथन string की forward indexing 0 ,1 ,2 ,3 ..... से शुरू होती है जबकि backward indexing -1 , -2 , -3 ..... से शुरू होती है |

उदहारण :-

```
a= "ehindistudy"
b=' hello python'
print(a)
print(b)
```

आउटपुट :- *ehindistudy*
Hello python

List

list पाइथन का एक महत्वपूर्ण डाटा टाइप है |इसमें बहुत सारी values को रखा जाता है | प्रत्येक वैल्यू को comma (,) से seperate किया जाता है | तथा सारी values को एक square bracket ([ ]) के अंदर रखा जाता है |

उदहारण :-

```
>>> list = ['amit', 25, 25.322, "kathait", "R"]
>>> print (list)
```

आउटपुट :- ['amit', 25, 25.322, 'kathait', 'R']

list की indexing string की indexing के similar होती है | लिस्ट की indexing भी 0 से start होती है |

Updating list (लिस्ट को अपडेट करना) :-

list को आप आसानी से update कर सकते है| नई वैल्यू को लिस्ट मे add कर सकते है | या फिर पुरानी वैल्यू को बदल कर के उसकी जगह नई वैल्यू रख सकते हो|

उदहारण :-

```
>>> list = ['amit', 25, 25.322, "kathait", "R"]
```

>>>print ("Value available at index 2 = ", list[2])
आउटपुट :- *Value available at index 2 = 25.322*
>>> list [2]=1997
>>>print ("New Value available at index 2 = ", list[2])
आउटपुट :- New Value available at index 2 = 1997

Tuples

Tuples भी list की तरह होता है जो की बहुत सारी अलग अलग वैल्यू को hold करता है इसमें भी प्रत्येक वैल्यू को comma (,) से separate किया जाता है | तथा सारी values को parentheses ( ) के अंदर रखा जाता है | यह एक immutable डाटा टाइप है | अर्थात
tuples की वैल्यू को चेंज नहीं कर सकते |

उदहारण :-

>>> tuple = ("python", 25, 12.23 ,"a")
>>> print (tuple)
आउटपुट :- *('python', 25, 12.23, 'a')*
>>> print ("tuple [0] = ", tuple [0])
आउटपुट :- *tuple [0] = python*
>>> print ("tuple [1:3] = ", tuple [1:3])
आउटपुट :- *tuple [1:3] = (25, 12.23)*

Updating tuple:–

Tuples immutable होता है जिसका मतलब ये होता है की tuples को अपडेट तथा इसकी वैल्यू को change नहीं कर सकते |

उदहारण :-

>>> tuple = ("hello", 635.35, 13 ,"d",1997)
>>>tuple [1] = 2000
Error msg :- type error

Dictionaries

list और tuples की तरह Dictionaries भी एक महत्वपूर्ण python data types है. | तथा यह एक mutable डाटा टाइप है |

इसमें key तथा value का जोड़ा (pairs) होता है | प्रत्येक key और उसके value को colon (:) से seperate तथा key और value के जोड़े को comma (,) से seperate किया जाता है | तथा सारी key और values को एक curly braces { } के अंदर रखा जाता है |

उदहारण :-

>>> dict1 = {'Name': 'ehindi', 'Age': 25, 'branch': 'IT'}
>>> print ("dict1['Name'] = ", dict1['Name'])
>>> print ("dict1['Age'] = ", dict1['Age'])

आउटपुट :- *dict1['Name'] = ehindi*
dict1['Age'] = 25

Updating Dictionaries:-

Dictionaries को update कर सकते है | इसमें हम नई entry को add कर सकते है तथा साथ मे वैल्यू को भी change कर सकते है |

उदहारण :-

>>> dict2 = {'Name': 'ehindi', 'Age': 20, 'Class': 'IT'}
>>> dict2 ['Age'] = 25
>>> dict2 ['School'] = "XYZ"
>>> print (dict2)

आउटपुट :- *{'Name': ' ehindi ', 'Age': 25, 'Class': 'IT', 'School': 'XYZ'}*

python operators in hindi:-

आज हम इस पोस्ट में python operators के बारें में पढेंगे और इसके types के बारें में जानेंगे तो चलिए शुरू करते है:-

operator एक symbol है जो कि एक ऑपरेशन को प्रस्तुत करता है. और वह वैल्यू जिस पर ऑपरेटर कार्य (operate) करते है उन्हें operands कहते है. Operators का प्रयोग प्रोग्राम में data तथा variables को manipulate करने के लिए किया जाता है.
अर्थात् operators वो symbols होते है जिनका प्रयोग प्रोग्राम में गणितीय या लॉजिकल ऑपरेशन करने के लिए किया जाता है.

python operators दो प्रकार के होते है:- unary operators तथा binary operators.

unary operator केवल एक operand पर operate होता है. जबकि binary operator दो operands पर operate होता है.

उदाहरण के लिए:- 4+6, जहाँ 4 तथा 6 operands है जबकि + एक ऑपरेटर है.
types of python operators in hindi:-

python operators निम्नलिखित होते है:-

1:- arithmetic operator
2:- relational (comparison) operator
3:- logical operator
4:- bitwise operator
5:- assignment operator
6:- identity operator
7:- membership operator
1:- arithmetic operators:-

arithmetic operators का प्रयोग गणितीय कार्यों को करने के लिए किया जाता है जैसे:- addition, substraction, division आदि.

यहाँ पर हमने उदाहरण के लिए वेरिएबल x का मान 20 तथा y का मान 30 लिया है.

operators	description	उदाहरण
+ (addition)	यह दो operands को जोड़ता है.	x + y = 50
– (substraction)	यह दो operands को घटाता है.	x – y = -10
* (multipication)	यह दो operands को गुणा करता है	x * y = 600
/ (division)	यह right वाले operands से left वाले operands को divide करता है	y / x = 1.5
% (modulus)	यह right वाले operand से left वाले operand को divide करता है तथा यह शेषफल return करता है.	y * x = 0
// (floor division)	यह right वाले operands से left वाले operands को divide करता है तथा भागफल return करता है. यह दशमलव वाले हिस्से को छोड़ देता है	जैसे:- 13//3 = 4 तथा 15//2 = 7
** (exponential)	इसमें right वाला operand, left वाले operand की power (घात) बन जाता है.	x ** y (x to the power y) माना x = 2, y =4 तो x ** y = 16

2:- relational (comparison) operators:-

रिलेशनल ऑपरेटर का प्रयोग दो values को compare करने के लिए किया जाता है. यह condition के अनुसार या तो true return करता है या false.
यहाँ पर हमने उदाहरण के लिए वेरिएबल x की वैल्यू 20 तथा y की वैल्यू 30 दी है.

operators	description	उदाहरण
==	यदि दोनों operands की वैल्यू समान होती है तो condition सत्य होती है.	(x == y) सत्य नहीं है.
!=	यदि दोनों operands की वैल्यू समान नहीं होती है तो condition सत्य होती है.	(x != y) सत्य है
< >	यदि दोनों operands की वैल्यू समान नहीं होती है तो condition सत्य होती है. यह != की तरह है.	(x < > y) सत्य है
>	यदि left operand की वैल्यू right operand से बड़ी होती है तो condition सत्य होती है.	(x > y) सत्य नही है.
<	यदि right operand की वैल्यू left operand से बड़ी होती है तो condition सत्य होती है.	(x < y) सत्य है.
>=	यदि left operand की वैल्यू right operand से बड़ी होती है या बराबर होती है तो condition सत्य होती है	(x >= y) सत्य नही है.
<=	यदि right operand की वैल्यू left operand से बड़ी होती है या छोटी होती है तो condition सत्य होती है	(x <= y) सत्य है.

3:- logical operators:-

लॉजिकल ऑपरेटर जो है वह Boolean expressions को compare करता है तथा Boolean result रिटर्न करता है.

operators	description	उदाहरण
and	यदि दोनों operands सत्य होते है तो condition सत्य होती है.	6>3 and 5<10 सत्य है
or	यदि दो operands में से एक भी सत्य होता है तो condition सत्य होती है.	6>3 or 5>10 सत्य है
not	यदि operands की वैल्यू गलत होती है तो condition सत्य होती है.	not(5 < 2) सत्य है

4:- bitwise operators:-

bitwise python operators जो है वह bits पर कार्य करते है तथा bit by bit ऑपरेशन को परफॉर्म करते है.

उदाहरण के लिए माना x = 10 तथा y = 4

तो बाइनरी फॉरमेट में x = 0000 1010 तथा y = 0000 0100

operators	meaning	उदाहरण
&	bitwise AND	x & y = 0 (0000 0000)
|	bitwise OR	x | y = 14 (0000 1110)
~	bitwise NOT	~ x = -11 (1111 0101)
^	bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

5:- assignment operators:-

python में assignment operators का प्रयोग variables को वैल्यू assign करने के लिए किया जाता है.

operators	description	उदाहरण	same as
=	यह बायीं तरफ के expression को वैल्यू assign करता है.	x = 5	x = 5
+=	यह right operand को left operand से जोड़ता है तथा जो result आता है उसे left operand को assign करता है.	x += 5	x = x + 5
-=	यह right operand को left operand में घटाता है तथा जो result आता है उसे left operand को assign करता है.	x -= 5	x = x – 5
*=	यह right operand को left operand में गुणा करता है तथा जो result आता है उसे left operand को assign करता है.	x *= 5	x = x * 5
/=	यह left operand को right operand से divide करता है तथा जो result आता है उसे left operand को assign करता है.	x /= 5	x = x / 5
%=	यह दो operands का modulus लेता है और उसे left operand को assign करता है.	x %= 5	x = x % 5
**=	यह operands में power (घात) को परफॉर्म करता है तथा उसे left operand को assign करता है.	x **= 5	x = x ** 5
//=	यह operators में floor division को परफॉर्म करता है	x //= 5	x = x

	तथा उसे left operand को assign करता है.		// 5

6:- identity operators:-

identity python operators दो ऑब्जेक्ट्स के मैमोरी लोकेशन को compare करता है.

operators	description	उदाहरण
is	यदि दोनों operands एक ही जैसी identity साझा करते है तो condition सत्य होगी अन्यथा असत्य होगी.	x is y
is not	यदि दोनों operands एक ही जैसी identity साझा नहीं करते है तो condition सत्य होगी अन्यथा असत्य होगी.	x is not y

7:- membership operators:-

membership python operators का प्रयोग sequence में membership को टेस्ट करने के लिए किया जाता है. sequence एक list, एक string या एक tuple हो सकती है. python में दो मेम्बरशिप ऑपरेटर है जो नीचे दिए गये है:-

operators	description	उदाहरण
in	यदि ऑपरेटर के दोनों तरफ के वेरिएबल एक sequence में होते है तो condition सत्य होगी अन्यथा असत्य होगी.	x in y
not in	यदि ऑपरेटर के दोनों तरफ के वेरिएबल एक sequence में नहीं होते है तो condition सत्य होगी अन्यथा असत्य होगी.	x not in y

Python Loops क्या है?

सभी प्रोग्रामिंग लैंग्वेज (जैसे की c++ , java आदि) मे हमे loops की सुविधा प्रदान की जाती है उसी प्रकार पाइथन लैंग्वेज मे भी loops का उपयोग किया जाता है |

Loops का उपयोग हम तब करते है जब हमे एक कोड को बार बार execute करना पड़ता है |

जैसे की हमे किसी कोड या फंक्शन को 200 से ज्यादा बार execute करना है तो हमे इस कोड को उतनी ही बार कॉल करना पड़ेगा या लिखना पड़ेगा जितनी बार कोड को execute करना है इस प्रोसेस मे हमे कोड को लिख़ने मे ज्यादा समय लग जाता है | इस प्रॉब्लम को दूर करने के लिए हम लूप्स का प्रयोग करते है | लूप्स के द्वारा हम जितनी बार चाहे उतनी बार प्रोग्राम को execute करा सकते है | इससे हमे कोड को बार बार कॉल करने की जरूरत नहीं पड़ती |
loops मे हम condition का उपयोग करते है जिससे की हम लूप्स को start और end करा सकते है | लूप्स मे अगर हमारी कंडीशन true होती है तो code / loops execute होता रहेगा जैसे की कंडीशन false होती है कोड terminate हो जाता है

types of loops in python

पाइथन मे तीन प्रकार के लूप्स होते है |

1. while loop
2. for loop
3. nested

while loop in python :-

while लूप को हम आसानी से define कर सकते है | ये एक simple लूप है |

syntax :-

```
while condition

  statements
```

उदहारण :- यदि हमे hello word को 10 बार प्रिंट करवाना है दो इसके दो तरीके हो सकते है |
1. पहला या तो हम 10 बार hello word को लिख कर प्रिंट कराये | जिससे की प्रोग्राम काफी बड़ा हो जायेगा |
२. दूसरा हम loop का प्रयोग करके hello word को print करे | जिससे प्रोग्राम काफी सरल हो जायेगा | हम अब देख़ते है की लूप द्वारा हम कैसे hello word को 10 बार print करेंगे |

```
i= 1

while i<=10:

   print ("hello word")

   i=i+1
```

step 1:- सबसे पहले हमने एक variable **i= 1** initialise किया जिससे की हमे यह समझने मे आसानी होगी की while loop कैसे काम करेगा

step 2:- फिर हम while कीवर्ड के साथ condition (**while i<=10:**) लिख़ते है | अगर हमे लूप 1० बार चलाना हो हम (**while i<=10:**) लिखेंगे और अगर 100 बार चलना हो तो (**while i<=100:**) लिखेंगे |

step 3:- फिर हम statement लिखेंगे जो हमे प्रिंट करवानी है |

step 4:- फिर i की value को increment करेंगे जिससे की i की value चेंज होती रहेगी |

जब तक कंडीशन **i<=10** false नहीं होती तब तक लूप चलता रहेगा और हेलो वर्ड प्रिंट होता रहेगा |

for loop in python

पाइथन मे for loop का syntax बाकि ओर प्रोग्रामिंग लैंग्वेज से काफी अलग होता है | for loop का प्रयोग किसी sequence (जैसे की list, tuples, dictionaries और set आदिजैसे की list, tuples, dictionaries और set आदि) को iterate करने के लिए किया जाता है |

syntax :-

```
for < Variable_Name > in < Sequence_Name >

   statements
```

उदहारण :- ऊपर दिए गए उदहारण को हम for loop के सहायता से 10 बार प्रिंट कराएँगे

```
for i in range (10):

   print ("hello word !")
```

step 1:- सबसे पहले हम for कीवर्ड को लिख़ते है |
step 2:- फिर वेरिएबल का नाम लिख़ते है जो की sequence को iterate करेगा |
step 3:- फिर in ऑपरेटर को लिख़ते है |
step 4:- उसके बाद हम sequence का नाम लिखेंगे जिसको iterate करना है |
step 5:- फिर हम statement लिखेंगे जो हमे प्रिंट करवानी है

break *and* continue *keyword*

break keyword का प्रयोग हम लूप को ख़त्म करने लिए करते है |

example:-

```
for i in range (1 , 11):
   if i==5:
      break
   print (i)
```

output:-

```
1
2
3
4
```

ऊपर दिए गए example मे जब i == 5 होगा तो लूप ब्रेक हो जायेगा ओर हमे ऊपर दिया गया आउटपुट मिलेगा |

continue keyword , loopके current iteration को stop करता है ओर फिर continue आगे बढ़ता है |

example :-

```
for i in range (1 , 11):
```

```
    if i==5:
        continue
    print (i)
```

output :-

1 2 3 4 6 7 8 9 10

ऊपर दिए गए example मे जब i == 5 होगा तो current iteration stop हो जायेगा ओर हमे ऊपर दिया गया आउटपुट मिलेगा |

nested loop in python

पाइथन मे भी हम बाकि प्रोग्रामिंग लैंग्वेज की तरह एक लूप के अंदर दूसरा लूप define कर सकते है |

example :-

```
for i in range(1, 4):
    for j in range(i):
        print(i, end=' ')
    print()
```

output :-

1

2 2

```
3 3 3
```

Control Statements in hindi

पाइथन प्रोग्रामिंग लैंग्वेज मे तीन प्रकार के control Statement होते है |

1. if statement
2. if_else statement
3. if_elif_else statement

control statement का अर्थ होता है की जब तक प्रोग्राम मे दिया गया कोई condition सत्य (true) नहीं होगा तब तक ये अपने statement को execute नहीं करता |

1. if Statement

if statement का प्रयोग बहुत सारी प्रोग्रामिंग लैंग्वेज जैसे की c, c++, java आदि के साथ साथ पाइथन मे भी किया जाता है|

if statement मे जब कोई condition true होगा तो ही statement execute होगा अथवा नहीं |

```
# SYNTAX

if condition:

    statement(s)

age =  int(input("enter age ------- "))

if age >= 15:

    print ("you are above 15 ")
```

```
output 1 :-

enter age ------- 16      # condition is true

you are above 15



output 2 :-

enter age ------- 12      #condition is not true
```

ऊपर दिए गए प्रोग्राम मे जब condition true हुई तो हमे output मिला जबकि condition true न होने पे हमे कोई भी आऊट्पुट नहीं मिला |

2. if_else Statement

if_else statement मे जब कोई condition true होगा तो if block का statement execute होगा | और जब कोई condition false होगा तो else block का statement execute होगा |

```
#SYNTAX

if condition :

    statement (s)

else :

    statement (s)

email = "abc@gmail.com"

passwd = "123456"
```

```
em = input ("enter email------  ")

pa = input ("enter your passwd ---- ")

if email==em and passwd ==pa :

    print (" log in. ")

else :

    print ("wrong email id ..")
```

 # output 1 :-

```
enter email------  abc@gmail.com

enter your passwd ---- 123456

 log in.
```

 # output 2 :-

```
enter email------ cba@gmail.com

enter your passwd ---- 123456

wrong email id ..
```

ऊपर दिए गए प्रोग्राम मे अगर दोनों condition true होगी तो ही if block का statement प्रिंट होगा | यदि दोनों कंडीशन मे से कोई भी एक condition या दोनों condition false होगी तो else block का statement प्रिंट होगा

3. if_elif_else Statement

if_elif_else statement मे अगर if block का condition true होगा तो if block का स्टेटमेंट प्रिंट होगा अगर | if ब्लॉक का कंडीशन false होगा तो फिर elif ब्लॉक check होगा elif true होने पे elif block का statement प्रिंट होगा | अगर if और elif दोनों false होंगे तो else block का statement प्रिंट होगा |

```
# SYNTAX


if condition :
    statement (s)
elif condition :
    statement (s)
else :
    statement (s)
a = int (input ("enter the no :----  "))
if 0<= a <100:
    print ("java")
elif 100<=a < 150:
    print ("python")
else :
```

```
    print ("c++")
```

output 1:-

enter any no :---- 111

python

output 2:-

enter any no :---- 30

java

output 3:-

enter any no :---- 210

c++

पाइथन में फंक्शन क्या है?

Python Function ये statements का एक समूह होता है | हर एक Program में एक function तो होता ही है | जहाँ पर Function की statements की जरुरत होती है वहाँ पर Function को call किया जाता है |

Python Functions Advantages

- Function में लिखा हुआ code बार बार लिखना नहीं पड़ता-|
- बड़े Program को छोटे छोटे-function में विभाजित किया जा सकता है |
- Function Programmer का समय, Program की space और memory बचाता है |
- अगर Program में कहा पर error आ जाए तो उसे आसानी से निकाला जा सकता है |
- जहाँ पर जरुरत हो वहाँ पर function को बार बार-call किया जा सकता है |

फंक्शन के प्रकार

1. In-Built/Predefined Function
2. User-Defined Function

1. In-Built/Predefined Function

Python में अभीतक print() नामक function बहुत ही बार इस्तेमाल किया है | print() function ये Python में in-built function है |

2. User-Defined Function

Python में user-defined function में दो प्रकार पड़ते है |

- Function Definition
- Function Call

2.1 Function Definition

Syntax for Function Definition

```
def function_name(parameter(s)):
      "function_docstring"
      function_body
      return statement
```

- **def :** Python में function को create करना हो तो शुरुआत में 'def' keyword का इस्तेमाल किया जाता है |
- **function_name :** def keyword के बाद function का नाम दिया जाता है | हर function का नाम उसके statement से related हो तो अच्छा रहता है |
- **(parameter(s)) :** Optional. parameters optional होते है parenthesis(()) नहीं होते है | function के name के बाद parenthesis(()) दिया जाता है | उन parenthesis में एक या एक से ज्यादा parameters दिए जाते है | parameters; optional होते है |
- **:** : parenthesis के बाद colon(:) देना अनिवार्य होता है | colon देने के बाद Python interpreter द्वारा automatic code indent होता है |
- **"function_docstring" :** Optional. यहाँ पर documentation string दिया जाता है |
- **function_body :** Optional. ये function की body होती है | जिसमे कुछ local variables और statements हो सकते है | body में दिए हुए variables को global भी बनाया जा सकता है |
- **return statement :** Optional. ये return statement होता है | return statement के लिए 'return' keyword का इस्तेमाल किया जाता है | ये statement से function end होता है | अगर return statement दिया नहीं जाता है तो default 'None' return होता है |

Example for Function Definition

Source Code :

```
#Function definition

def func(param):          #function name, parameter and
colon

    "This is a docstring" #docstring

    print(func.__doc__)   #function body

    return param          #return statement
```

Example पर 'func' ये function का नाम है |

उसके बाद parenthesis(()) में 'param' नाम का एक parameter और parenthesis के बाहर colon(:) दिया गया है |

function के indent code में पहले docstring दी गयी है |

उसके बाद function की body में class attribute की मदद से docstring को print किया गया है

और आखिर में return statement में 'param' इस parameter को print किया गया है |

2.1 Function Call

Function के call में सिर्फ function का नाम और अगर function को parameter हो तो parameter की value दी जाती है | जबतक function को call नहीं किया जाता तबतक function का code execute नहीं होता है |

Example for Function Call

Source Code :

```
#Function definition

def func(param):          #function name, parameter and
colon

    "This is a docstring" #docstring

    print(func.__doc__)   #function body

    return param          #return statement


print("Call 1")

print(func(5))            #Function call


print("Call 2")

print(func(10))           #Function call
```

Output :

```
Call 1
```

```
This is a docstring

5

Call 2

This is a docstring

10
```

Python में जब function को call करके जिस data type की value as a argument दी जाती है तब उसका data type decide हो जाता है |

Source Code :

```
def func(num, str):

    print(num)

    print(str)


print("Function Call1 :")

func("Hello", 2)

print("Function Call2 :")

func(2, "Hello")
```

Output :

```
Function Call1 :

Hello

2

Function Call2 :

2

Hello
```

Function Argument in Python

Python में चार प्रकार के function parameter होते है |

1. Default Argument
2. Required Argument
3. Keyword Argument
4. Variable Number of Argument

1. Default Argument

Default argument में definition पर argument के लिए default value '='(assignment operator) से set की जाती है |

Function Call पर जब definition पर assign किया हुआ argument नहीं दिया जाता तो default value pass की जाती है |

Source Code :

```
def DefArg(num=5, str="Hello"):

    print(num)

    print(str)


print("Call1")

DefArg()

print("Call2")

DefArg(8)

print("Call3")

DefArg(8, "Hii")
```

Output :

```
Call1

5

Hello

Call2

8

Hello

Call3
```

```
8

Hii
```

2. Required Argument

Required Argument में function call पर argument देना अनिवार्य होता है |

जब function definition पर argument दिया जाता है तब उसे function call पर उसकी value देना required होता है |

Source Code :

```
def ReqArg(num, str):
    print(num)
    print(str)


print("Call1")
ReqArg(5, "Hello World")


print("Call2")
ReqArg("Hello World", 5)
```

```

```

Output :

```
Call1

5

Hello World

Call2

Hello World

5
```

अगर function call पर argument दिया नहीं जाता है तो , TypeError का exception आ जाता है |

Source Code :

```
def ReqArg(num, str):

    print(num)

    print(str)


ReqArg()
```

Output :

```
    ReqArg()

TypeError: ReqArg() missing 2 required positional
arguments: 'num' and 'str'
```

अगर function पर दो arguments है | अगर call करते वक्त एक ही argument value दी जाती है तो तब भी 'TypeError' exception आ जाता है |

Source Code :

```
def ReqArg(num, str):

    print(num)

    print(str)


ReqArg(5)
```

Output :

```
    ReqArg(5)

TypeError: ReqArg() missing 1 required positional
argument: 'str'
```

3. Keyword Argument

अबतक function पर positional argument का इस्तेमाल किया गया है | Positional argument ये normal parameters होते है |

positional argument में जब values दी जाती है तब उसकी position की हिसाब से call पर value assign की जाती है |

Keyword Argument में function call पर दिया जाता है | Keyword Argument की मदद से arguments की positions को बदला जाता है |

Source Code :

```
def KeyArg(int, float, str):
    print("int :",int)
    print("float :",float)
    print("String :",str)


print("Call1")
KeyArg(int=10, float=1.5, str="H")
print("Call2")
KeyArg(float=2.8, int=20, str="G")
print("Call3")
KeyArg(str="R", float=8.6, int=5)
```

Output :

```
Call1

int : 10

float : 1.5

String : H

Call2

int : 20

float : 2.8

String : G

Call3

int : 5

float : 8.6

String : R
```

4. Variable Number of Arguments

किसी वक्त पर पता नहीं होता कि function पर कितने arguments pass करने है | ऐसे वक्त पर 'Number of Arguments' का महत्व काफी बढ़ जाता है |

जब number of argument का इस्तेमाल करना हो तो Function के definition पर argument से पहले '*'(asterisk) का इस्तेमाल किया जाता है |

'Function definition पर दिया हुआ argument 'tuple' जैसा होता है |

Source Code :

```
def MultiArg(*argTuple):


    print(argTuple)

    print(argTuple[0])

    print(argTuple[1])

    print(argTuple[2])


    for i in argTuple:

        print(i)


MultiArg(5, 10, "Hello")
```

Output :

```
(5, 10, 'Hello')

5

10
```

```
Hello

5

10

Hello
```

Anonymous or Lambda Function

जब Normal Function create किया जाता है तब function को विशिष्ट नाम दिया जाता है | लेकिन Anonymous या lambda Function का कोई नाम नहीं होता है |

Normal Function के लिए 'def' keyword का इस्तेमाल किया जाता है लेकिन Anonymous/Lambda Function के लिए 'lambda' keyword का इस्तेमाल किया जाता है | ये function काफी छोटा होता है |

Syntax for Anonymous/Lambda Function

```
lambda arg1,arg2,..,argN : expression
```

arg1,arg2,..,argN : lambda function में एक या एक से ज्यादा arguments हो सकते है |

expression : lambda function में एक ही expression होता है | expression; return भी होता है |

Example पर 'anony' नाम के function object पर anonymous function को assign किया गया है | उसके बाद function object का इस्तेमाल function के रूप में call करके और argument पर values pass की गयी है |

Source Code :

```
anony = lambda a, b : a < b


print(anony(5, 4))

print(anony(4, 5))
```

Output :

```
False

True
```

Another Example for Anonymous/Lambda Function

Source Code :

```
anony = lambda a, b : print(a < b)

anony(5, 4)

anony(4, 5)
```

Output :

```
False

True
```

Another Example for Anonymous/Lambda Function

Source Code :

```
anony = lambda a, b : print(a < b) Aanonymous Function


#same as


def anony(a, b):      #Named/Normal Function

    return a < b
```

What is Exception ?

Exceptions ये Python में Errors के प्रकार होते है | Exception के अलग-अलग प्रकार पड़ते है | हर अलग-अलग error के लिए अलग-अलग exception होता है |

जब Python के program पर error occured होता है तब कोई ना कोई exception raise होता है |

जब program पर किसी statement के लिए exception raise होता है तो उस statement के आगे की script exeception की वजह से execute नहीं होती है |

इन scripts को executes करने के लिए error/exeception को handle करना पड़ता है |

For Example,

Example पर तीन variables को defined करके print किया गया है | लेकिन जहा पर print statements है वहा पर अतिरिक्त undefined 'd' variable को print करने की अनुमति दी गयी है |

'd' variable defined न होने की वजह से exeception raise होता है | 'd' के print statement से पहले का print statement execute होता है लेकिन उसके बाद के statements print नहीं हो पाते है |

उन अगले statements को print करने के लिए exception को handle करना पड़ता है |

```
a = 10

b = 20

c = 30

print(a)

print(d) #exception raised

print(b)

print(c)
```

Output :

```
10

    print(d)

NameError: name 'd' is not defined
```

Example for Exception Handling

Example पर Exception को handle किया गया है |

```
a = 10

b = 20

c = 30


print(a)


try:

    print(d)

except(Exception):

    print("Variable is not defined")
```

```
print(b)

print(c)
```

Output :

```
10

Variable is not defined

20

30
```

try_except Statement(Exception Handling) in Python

Python में error को handle करना हो तो 'try_except' statement का इस्तेमाल करना पड़ता है | Python में जिस code पर error occured होने की संभावना होती है वो code 'try' block पर आता है और except block पर exception/error को handle करने के लिए कुछ code दिया जाता है |

Syntax for 'try_except' Statement

```
try:

      try_block_code

except Exception1:
```

```
      exception handling code 1
except Exception2:
      exception handling code 2
```

Syntax में देखे तो एक try block के लिए एक या एक से अधिक except clauses लिए जा सकते है |

Example for 'try_except' Statement

Example में try block पर जो code दिया है वो 'ZeroDivisionError' exception raise करता है इसीलिए ZeroDivisionError का ही except clause execute होगा |

Source Code :

```
try:
    a = 10
    a/0
except NameError:
    print("'a' is not defined")
except ZeroDivisionError:
    print("Divided By Zero Error")
```

Output :

```
Divided By Zero Error
```

'try_except' with 'else' Statement(Exception Handling) in Python

Syntax for 'try_except' Statement

```
try:
      try_block_code
except Exception1:
      exception handling code 1
except Exception2,Exception3:
      exception handling code 2
else:
      else_block_code
```

Syntax में देखे तो एक try block के लिए एक या एक से अधिक except clauses लिए जा सकते है |

जब try block code में दिया हुआ code कोई भी exception raise नहीं करता है तो else block code execute होता है |

Example में try block code कोई exception raised नहीं करता है इसीलिए else block code execute होगा |

Source Code :

```
try:
    a = 10
    a/4
except NameError:
    print("'a' is not defined")
except ZeroDivisionError:
    print("Divided By Zero Error")
else:
    print("No Exception raised")
```

Output :

```
No Exception raised
```

except clause with More than one Exception

Syntax :

```
try:
```

```
      try_block_code
except Exception1:
      exception handling code 1
except (Exception2, Exception3):
      exception handling code 2
```

Syntax में देखे तो एक try block के लिए एक या एक से अधिक except clauses लिए जा सकते है और हर except clause में एक या एक से अधिक exceptions दिए जा सकते है | लेकिन उन multiple exceptions को parenthesis(()) में close करना पड़ता है |

Example :

```
try:
    b = a/4
    print("Value of b is",b)
except ZeroDivisionError:
    print("'a' is not defined")
except (TypeError,NameError):
    print("Something is wrong")
else:
    print("No Exception raised")
```

Output :

```
Something is wrong
```

try_except_finally Block(Exception Handling) in Python

Syntax :

```
try:
      try_block_code
except Exception(s):
      exception_handling_code
finally:
      always_executed_code
```

try block का code जब कोई execption raised करता है और उसे handle नहीं किया जाता है तो finally block का code पहले execute होता है |

try block का code जब कोई execption raised करता है और उसे handle किया जाता है तो except block का code पहले execute होता है और बाद में finally block का code execute होता है |

अगर exception raised होता है या नहीं होता है finally block का code हमेशा execute होता है |

Source Code :

```
try:
    "5" + 4
except TypeError:
    print("Value must be a Number")
finally:
    print("Alway executed")
```

Output :

```
Value must be a Number
Alway executed
```

Raising Exceptions

Exception user द्वारा raise करने 'raise statement' का इस्तेमाल किया जाता है |

raise Statement program में कोई भी exception raised करने के लिए मजबूर कर देता है |

Syntax :

```
raise exception('message') #argument is optional
```

Source Code :

```
raise TypeError("HI")
```

Output :

```
  raise TypeError("HI")

TypeError: HI
```

Another Example for Raising Exception

Source Code :

```
try:
    raise ValueError('ValueError raised forcely')
except Exception as e:
    print ('Error :',e)
```

Output :

```
Error : ValueError raised forcely
```

पाइथन में Class और Object क्या है?

Python ये Object-Oriented Programming(Objects) Language है | इसके साथ-साथ ये Procedural-Oriented Programming(Functions) Language भी है |

ज्यादातर Programming Languages में OOP का concept होता है | Python के Object में variables, functions या methods होते है |
जैसे कि, अगर कोई प्राणी है तो उस प्राणी का behavior और properties जैसे कि, उसका भौकना, चलना, देखना , उसके शरीर की रचना को variables और functions या methods के जरिये लिखा जाता है |
हर एक से अधिक प्राणी को अलग-अलग नाम से उनके objects भी बनाये जाते है |

What is a Class ?

- Class के अन्दर कुछ functions या methods होते है और उससे निगडित कुछ variables दिए जाते है | उन functions और variables को एक ही class पर इकट्ठा किया जाता है और उन data को access करने के लिए उस class का object बनाया जाता है |
- Class अपने data को hold करने का काम करता है |
- Class ये Object की blueprint या layout होता है |
- एक Class के एक या एक से ज्यादा Object create किये जा सकते है |

Defining Class in Python

class को create करने के लिए पहले 'class' keyword का इस्तेमाल किया जाता है |

Syntax :

```
class MyClass:
    "I am a Docstring"
    #class_body
```

Syntax पर 'myClass' ये class का नाम है |
उसके बाद एक docstring लिया गया है | ये docstring class में optional होता है |
docstring ये class के बारे में कुछ जानकारी देने के लिए लिया जाता है |
उसके बाद class_body में कुछ statements के रूप में कुछ variables, functions या methods हो सकते है |

Example for Creating a Class

Example पर Class के नाम से ही Object का इस्तेमाल किया गया है |इस class object; का इस्तेमाल class के अलग-अलग attributes access करने के लिए किया जाता है | निचे docstring को access करने के लिए '__doc__' इस class attribute का इस्तेमाल किया गया है और उसके बाद func ये attribute function object को return करता है |

Source Code :

```
class MyClass:          # Class Name
    "I am a Docstring"
    def func(self):
        print("Hello World")


print(MyClass.__doc__)    # docstring attribute
print(MyClass.func)       # func attribute
```

Output :

```
I am a Docstring

<function MyClass.func at 0x02423270>
```

Creating an Object in Python

Object को variable के रूप में ही create किया जाता है | उस variable पर जैसे function को call किया जाता है वैसे ही उस class को function के रूप में variable पर store किया जाता है |

Syntax :

```
Object_Name = Class_Name()
```

Example for Creating a Class

Example पर MyClass इस Class का Object 'obj' बनाया गया है |

Source Code :

```
class MyClass:

    "I am a Docstring"

      var = 1

    def func(self):
```

```
        print("Hello World")


obj = MyClass()  #Class Object
```

Accessing Class Variable and Function

Source Code :

```
class MyClass:
    "I am a Docstring"
    var = 1
    def func(self):
        print("Hello World")


obj = MyClass() #Object


print("MyClass.var :",MyClass.var)  #1
#same as
print("obj.var :",obj.var)          #2


print("MyClass.func :",MyClass.func)#3
```

```
print("obj.func :",obj.func)          #4


MyClass.func(obj)                     #5

#same as

obj.func()                            #6
```

Output :

```
MyClass.var : 1

obj.var : 1

MyClass.func : <function MyClass.func at 0x02993270>

obj.func : <bound method MyClass.func of
<__main__.MyClass object at 0x0278AD50>>

Hello World

Hello World
```

Explaination of Above Example

- **Comment#1 :** यहाँ पर Class name वाले Object(MyClass) से class variable को access किया गया है |
- **Comment#2 :** यहाँ पर Class के बनाये हुए Object(obj) से class variable को access किया गया है |
- **Comment#3 :** यहाँ पर Class name वाले Object(MyClass) से func इस class function को access किया गया है लेकिन ये 'function object' को return करता है |

- **Comment#4 :** यहाँ पर Class के बनाये हुए Object(obj) से func इस class function को access किया गया है लेकिन ये 'method object' को return करता है |
- **Comment#5 :** यहाँ पर MyClass इस class के function पर जो definition पर 'self' इस parameter को pass किया गया है वैसे ही 'obj' इस class के object को argument पर pass किया गया है |
- **Comment#6 :** definition पर 'self' ये parameter pass किया गया है लेकिन यहाँ पर कोई भी argument को pass नहीं किया गया है इसका मतलब है कि जब बनाये हुए object द्वारा function को access किया जाता है तब खुद object ही argument पर pass हो जाता है | उसे अलग से देने की जरुरत होती है |

Class Attribute/Variable and Instance Attribute/Variable

Example पर class variable 'classVar' और instance variable 'inVar' लिया गया है |

class Variable अपने className या उसके object के जरिये access किये जा सकते है लेकिन instance variable को access करने के लिये class के object की जरुरत पड़ती है |

Instance Variable को constructor के अन्दर create किया जाता है |

Source Code :

```
class MyClass:

    classVar = 1 #classVar is a Class Variable


    def __init__(self, inVar):
```

```
        self.inVar = inVar   #inVar is a Instance
Variable


obj1 = MyClass("Ramesh")

obj2 = MyClass("Rahul")




print("Access Class Variable using Class Name :",
MyClass.classVar)

print("Access Class Variable using Class Object :",
obj1.classVar)

print(obj1.classVar, obj1.inVar) #Value of inVar is
Ramesh

print(obj2.classVar, obj2.inVar) #Value of inVar is
Rahul
```

Output :

```
Access Class Variable using Class Name : 1

Access Class Variable using Class Object : 1

1 Ramesh

1 Rahul
```

Changing Class Variable's and Instance Variable's Values

Class के variable की value change करने के लिए ClassName या ClassObject को इस्तेमाल किया जा सकता है लेकिन instance variable की value को change करना हो तो ClassObject का ही इस्तेमाल किया जाता है |

Source Code :

```
class MyClass:

    classVar = 1 #classVar is a Class Variable


    def __init__(self, inVar):

        self.inVar = inVar   #inVar is a Instance
Variable


obj1 = MyClass("Ramesh")

obj2 = MyClass("Rahul")


print("Before Changing :")

print(obj1.classVar, obj1.inVar)
```

```
MyClass.classVar = 2 #Changing Class Variable's Value
by ClassName

obj1.inVar = "Rakesh" #Changing Instance Variable's
Value



print("After Changing :")

print(obj1.classVar, obj1.inVar)
```

Output :

```
Before Changing :

1 Ramesh

After Changing :

2 Rakesh
```