



Introduction to Python

Dr. Sarwan Singh
NIELIT Chandigarh

Agenda

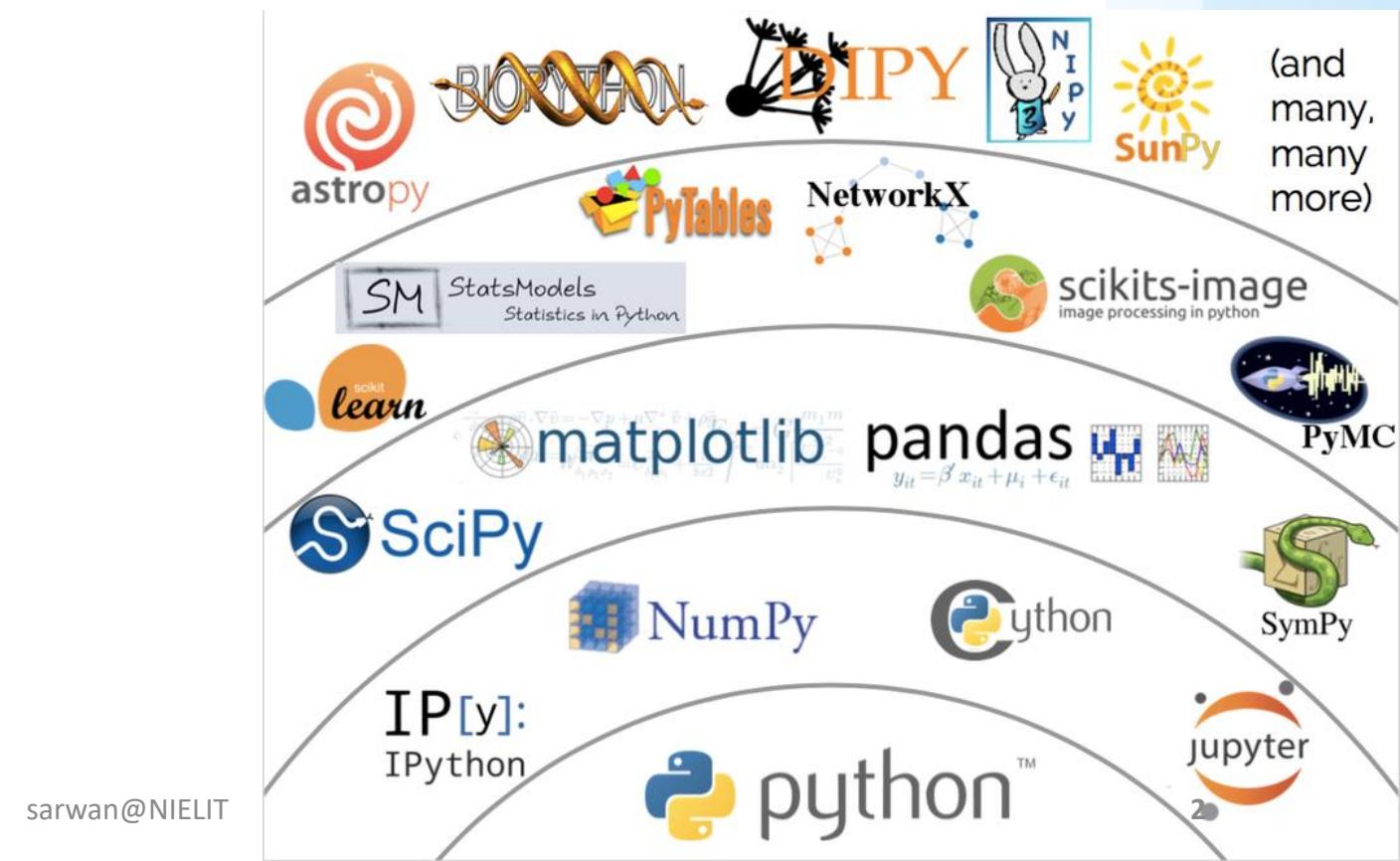
- Introduction
- History and usage
- Comparison
- companies using python
- Variables, Datatypes
- Keywords
- Looping constructs



Artificial Intelligence

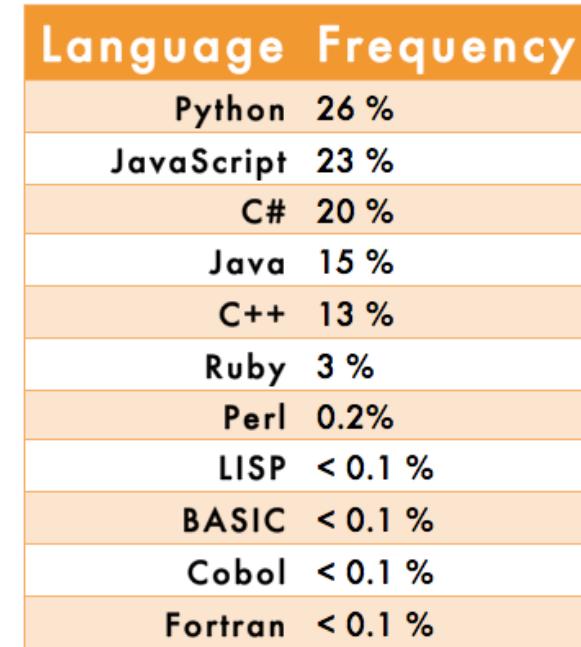
Machine Learning

Deep Learning



Python ... at first glance

- Python is an **interpreted, object-oriented, high-level** programming language with dynamic semantics.
- Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.
- *Python is portable: it runs on many Unix variants, on the Mac, and on PCs under MS-DOS, Windows, Windows NT, and OS/2.*
- *not to mention around 80 percent of the top computer science programs around the world teach Python as the introduction to the program.*



Source:masterbaboon



History of Python

- A scientist once said

"I have used a combination of Perl, Fortran, NCL, Matlab, R and others for routine research, but found out this general-purpose language, Python, can handle almost all in an efficient way from requesting data from remote online sites to statistics, and graphics."
- The programming language Python was conceived in the late 1980s, and its implementation was started in December 1989 by Guido van Rossum at CWI in the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Guido van Rossum was also reading the published scripts from “Monty Python's Flying Circus”, a BBC comedy series from the 1970s.
- Van Rossum thought he needed a name that was short, unique, and slightly mysterious, so he decided to call the language **Python**.



रा.ह.सू.प्रौ.सं
NIELIT



About Python

- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- Python source code is now available under the [GNU General Public License \(GPL\)](#).
- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- Python is a [scripting language](#) like PHP, Perl, Ruby and so much more.
- It can be used for [web programming \(django, Zope, Google App Engine, and much more\)](#).
- Can be used for [desktop applications](#) (Blender 3D, or even for games pygame).
- Python can also be translated into [binary code](#) like java.
- Python can be used for both [artificial intelligence](#) and [statistical analysis](#).
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

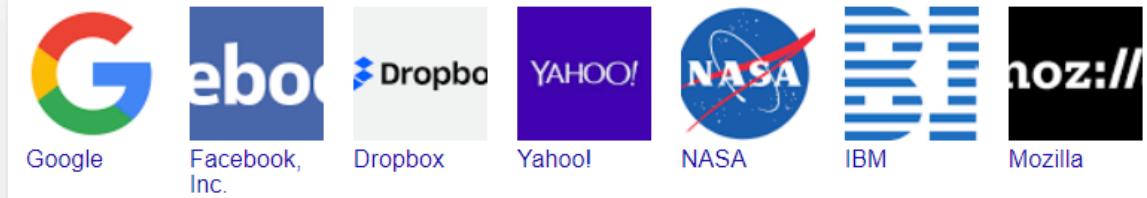


More about Python

- **Python** is developed under an OSI-approved open source **license**, making it freely usable and distributable, even for **commercial use**. **Python's license** is administered by the **Python Software Foundation**.
- **Python** is not an exception - its most popular implementation is called **C_{Python}** and is **written in C**
- The biggest difference between Java and Python is that **Java** is statically typed and **Python** is dynamically typed. *This makes Python very easy to write and not too bad to read, but difficult to analyze.*
- **IronPython** is the **Python** running on .NET
- **Jython** is the **Python** running on the Java Virtual Machine



Programming with Python



Some companies I know that use python are:

- Google (Youtube)
- Facebook (Tornado)
- Dropbox.
- Yahoo.
- NASA.
- IBM.
- Mozilla.
- Quora :D.

More items...

What top tier companies use Python? - Quora
<https://www.quora.com/What-top-tier-companies-use-Python>



Python Identifiers

- A Python identifier is a name used to identify a variable, function, class, module or other object.
- An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- Python does not allow punctuation characters such as @, \$, and % within identifiers.
- Python is a **case sensitive programming** language.



naming conventions for Python identifiers :

- Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
- Starting an identifier with a single leading underscore indicates that the identifier is private.
- Starting an identifier with two leading underscores indicates a strongly private identifier.
- If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.



Building Blocks

- Values and Variables
 - numeric values
 - variables
 - assignment
 - identifiers
 - reserved words
 - comments



Standard Data Types

- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types
 - Numbers
 - String
 - List
 - Tuple
 - Dictionary



Python Numbers

- Number data types store numeric values.
- Number objects are created when you assign a value to them. For example –
 - `var1 = 1`
 - `var2 = 10`
- You can also delete the reference to a number object by using the `del` statement. The syntax of the `del` statement is –
 - `del var1[,var2[,var3[....,varN]]]]`
 - You can delete a single object or multiple objects by using the `del` statement. For example –
 - `del var`
 - `del var_a, var_b`



Checking the type of datatype

```
In [1]: type('hello')
Out[1]: str

In [2]: type (10)
Out[2]: int

In [3]: type(20.5)
Out[3]: float

In [4]: a=10
a
Out[4]: 10

In [5]: del a

In [6]: a
-----
NameError                                 Traceback (most recent call last)
<ipython-input-6-60b725f10c9c> in <module>()
      1 a
NameError: name 'a' is not defined
```



Assigning Values to Variables

- Python variables **do not need explicit declaration** to reserve memory space.
- The declaration happens automatically when you assign a value to a variable.
- The equal sign (=) is used to assign values to variables.

- For example –

```
counter = 100      # An integer assignment  
miles   = 1000.0    # A floating point  
name    = "Nielit"   # A string  
print(counter)  
Print(miles)  
Print(name)
```

- Here, 100, 1000.0 and “Nielit” are the values assigned to *counter*, *miles*, and *name* variables, respectively.



Reserved words

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

Multiline/single line

In [7]: `a=10;b=20;c=30;
print(a,b,c)`

10 20 30

In [8]: `a\
=\\
50
print (a)`

50



Data type

Python supports four different numerical types –

- int (signed integers)
- long (long integers, they can also be represented in octal and hexadecimal)
- float (floating point real values)
- complex (complex numbers)



Basic operator

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
 $(+=, -=, *=, //=, \%=)$
- Logical Operators
(and, or, not)
- Bitwise Operators

Arithmetic Operator

- + Addition
- Subtraction
- * Multiplication
- / Division
- % Modulus
- ** Exponent
- // Floor division

$9//2 \rightarrow 4$

Comparison Operator

`==`
`!=`
`<>`
`>`
`<`
`>=`
`<=`

Bitwise Operator

- & Binary AND
- | Binary OR
- \wedge Binary XOR
- \sim Binary Ones Complement
- $<<$ Binary Left Shift
- $>>$ Binary Right Shift



Decision making

```
x = int( input('enter marks'))  
if (x>50) : print('pass')  
else : print('fail')
```

Or

```
x = int( input('enter marks'))  
if (x>50) :  
    print('pass')  
else :  
    print('fail')
```

```
if expression1:  
    statement(s)  
    if expression2:  
        statement(s)  
    elif expression3:  
        statement(s)  
    elif expression4:  
        statement(s)  
    else:  
        statement(s)  
    else:  
        statement(s)
```



Exercise

1. Write command to check whether input number is even or odd.
2. Write a command/program to accept marks from user and print the division.
3. Write command/s to return sum of digits of given number.



loops

while expression :
statements()

```
i=0
while (i<5) :
    print (i, 'Jai Ho')
    i=i+1
```

```
0 Jai Ho
1 Jai Ho
2 Jai Ho
3 Jai Ho
4 Jai Ho
```

while expression :
statements()
else :
statements()

```
i=0
while (i<5) :
    print (i, 'Jai Ho')
    i=i+1
else :
    print (i, ' Its over now')
```

```
0 Jai Ho
1 Jai Ho
2 Jai Ho
3 Jai Ho
4 Jai Ho
5 Its over now
```



for loop

for iterating Variable in sequence
statement/s

```
In [3]: states=['J&K', 'HimachalPradesh','Punjab','Delhi']
for st in states:
    print (st)
```

J&K
HimachalPradesh
Punjab
Delhi

```
In [4]: for st in range(len(states)):
    print (states[st])
```

J&K
HimachalPradesh
Punjab
Delhi

```
In [5]: for alpha in 'India':
    print(alpha)
```

I
n
d
i
a

for iterating Variable in sequence
statement/s
else:
statement/s

```
In [7]: for st in range(len(states)):
    print (states[st])
else :
    print('-----Its over -----')
```

J&K
HimachalPradesh
Punjab
Delhi
-----Its over -----



Exercise

1. Write program to check whether given number is prime or not
2. Write a program to find those numbers which are divisible by 7 and multiple of 5, between 1500 and 2700 (both included).
3. Write a Python program to get the Fibonacci series between 0 to 50.
4. Write a program to construct the pattern, using a nested for loop.

*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * * *
1
22
333
4444
55555
666666
7777777
88888888
999999999



Loop Control Statements

Break : Terminates loop statement

```
for alpha in 'Greatness':  
    if alpha == 'n':  
        break  
    print ('letter ', alpha)
```

letter G
letter r
letter e
letter a
letter t

continue : returns the control to the beginning of the while/for loop

```
for alpha in 'Greatness':  
    if alpha == 'n':  
        continue  
    print ('letter ', alpha)
```

letter G
letter r
letter e
letter a
letter t
letter e
letter s
letter s

pass : is used when a statement is required syntactically but you do not want any command or code to execute

```
for alpha in 'Greatness':  
    if alpha == 'n':  
        pass  
    print ('Pass block')  
    print ('letter ', alpha)
```

letter G
letter r
letter e
letter a
letter t
Pass block
letter n
letter e
letter s
letter s



Jupyter Notebook Link

- https://colab.research.google.com/drive/1jUAhD0Vvlz23RA8HrpDzK_srpTcsm?usp=sharing



Github Repository Link

- <https://github.com/sarwansingh/Python/tree/master/ORD>

