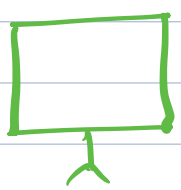
Introduction to Python-1

Instructions ✓
Language to Interact with Machine ✓

Python is a high-level (I), general purpose (II), interpreted (III) programming Language.

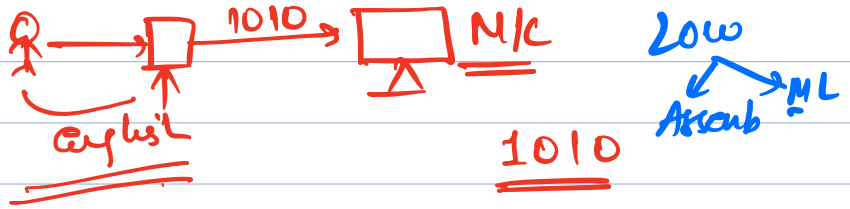
→ Easily used for wide range of Appl'n

Types of Prog. Lang. →
1. Low level
2. High level ✓
3. Compiled
4. Interpreted ✓
5. OOL ✓
6. Funct'n L.



1,0,0,1. — Binary.

python → print ("Hello!") ← Humans can read.

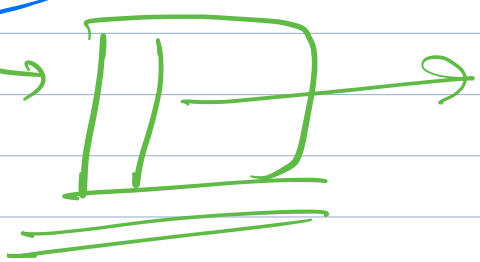


R.I → definitions x Technical

Interpreted

1. Take a bowl
2. put some all flour in the bowl
3. put S...

Compiled



read, understand do it yourself

Q. Compiled languages / Interpreted which are faster?

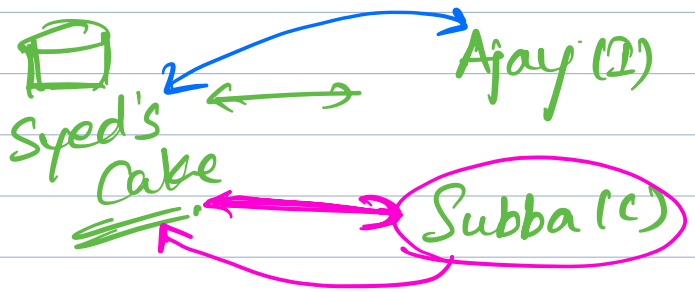
COMPILED.

Compiled	Interpreter
13	7

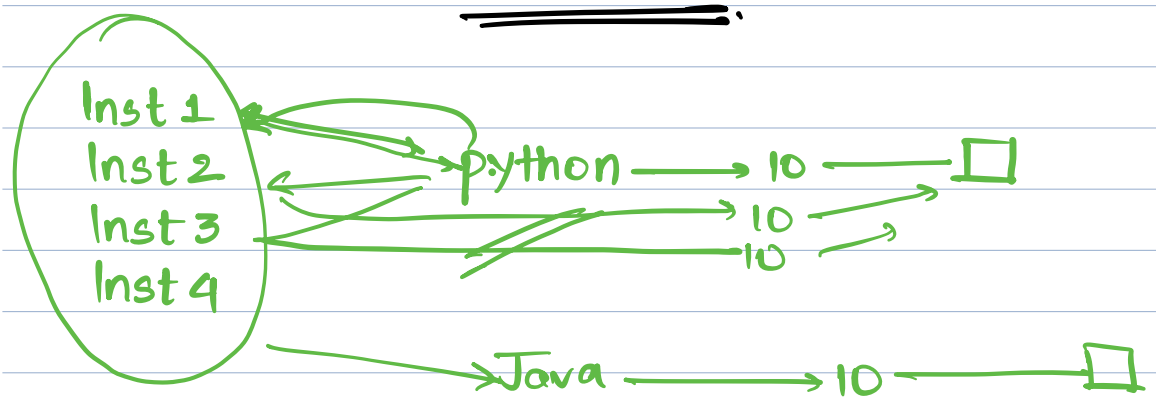
I Interpre reads 1by1
Compiled does everything @ once.

II Srinath → interpreter is slower (Sun!)

III



COMPILED



10 TB ⟷ 1.47 sec (Py)
1.46 sec
0.01

#>3.10
Pycharm
Practice
Mocks
Interview Prep.

} 5

Data Types
Data Structures
Loops & Conditionals
Coding.

Data Types

Numeric

1) Integers - 1, 2, 3
2) Float - 1.23, 1.44
3) Complex - 2+3i

Text

1) Strings
"Hi There"
"s"

Boolean

1) bool → T / F

'Santosh' ⟶ string
'Sunil' ⟶ string
34.31 ⟶ Float
6 ⟶ Integer.

DATA STRUCTURES

a way to store & organize the data so that it can be accessed and worked.

Python provide you built-in data structures that'll allow us handle collections of data

(M) # Lists
(I) # Tuples
(M) # Dictionaries
(M) # Sets

→ module in python that provide addtn D.S like
- → Queue
- → stacks
- → Dequeue
- → Named Tuple

(25th OCT)

MUTABLE & IMMUTABLE

↳ you can add/change/delete even after creation.

→ You can't do anything once created

LIST	[,]	M
TUPLE	(,)	I
DICT	{k:v}	M
SETS	{,}	M

LISTS

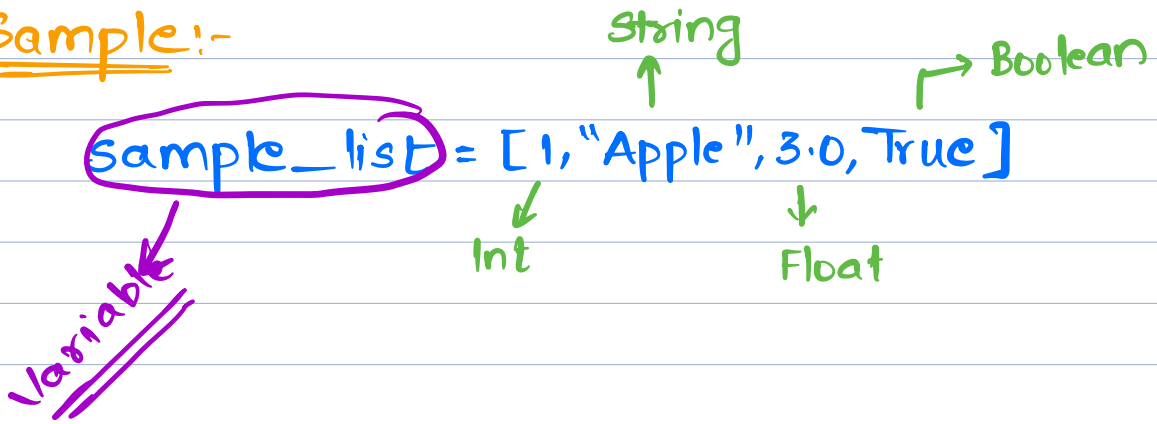
- Ordered Collection of elements
  (each element is placed @ a particular position)

- Mutable
  (You can modify them after creation- add/remove/change)

✓ Heterogeneous

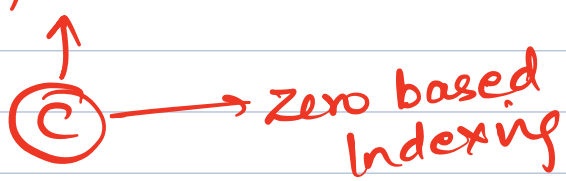
(elements can be of different data types)

Sample:-

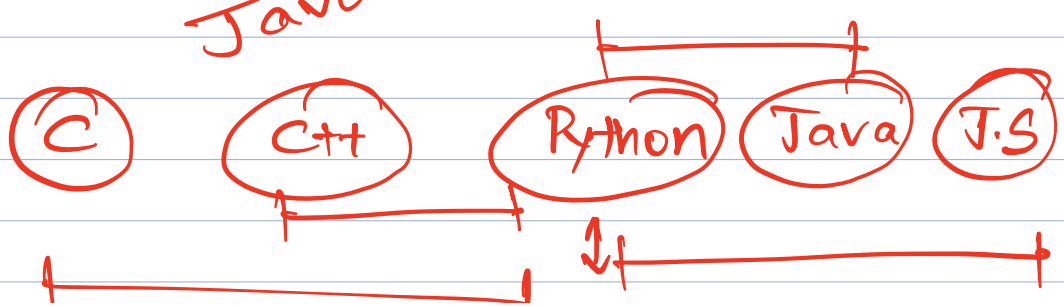


Note:

Python

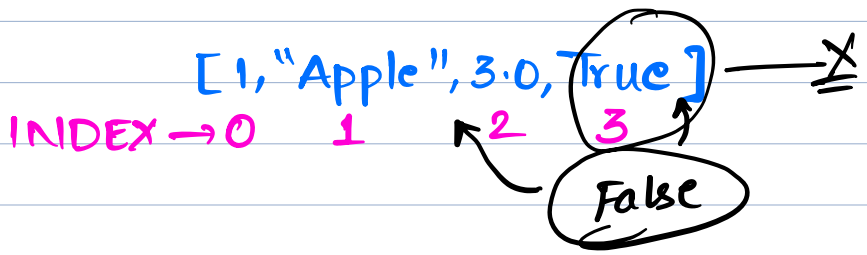


C++,
Java,
JavaScript



ZERO BASED INDEXING IS STANDARD

Memory Mgmt.
Math.
Compatibility.



1) CREATE
2) ACCESS ELEM
3) MODIFICATIONS
4) SLICING
5) LIST OPERATIONS -
6) LIST METHODS -
7) LIST COMPREHENSIONS.

25 ↑.

TUPLE

↓

DICT

↓

SET (30)

LOOPS — FUNCT

↓

OOPS

LIST CREATION

1) SQUARE BRACKETS
2) list() FUNCTION
3) LIST COMPREHENSION
4) Using range()

#1. [ ]

@ = [1,2,3,4,5]

numbers = [1,2,3,4,5]

numx

fruits = ["apple","mango", "Pear"]
Sunil_details = ["Sunil", 24, 40000]
vowels = ['a', 'e', 'i', 'o', 'u']

#2. list()

letters = list('Ajay') → o/p → ['Ajay'] ✗

→ ['A', 'j', 'a', 'y'] ✓

list ((1,2,3,4)) → o/p

~~[1234]~~ – I

[1,2,3,4] – II

[(1,2,3,4)] – III

#3. LIST COMPREHENSION

→ 25 min.

#4. range() ⟶ Function

↓

Generates a Sequence.

Syntax:- range(start, stop, step)
(0) (M) (1)

range(5) ⟶ 0, 1, 2, 3, 4

stop ↑ ↓ 0

range(1, 7) ⟶ 1, 2, 3, 4, 5, 6

stop ↓ → (stop - 1)

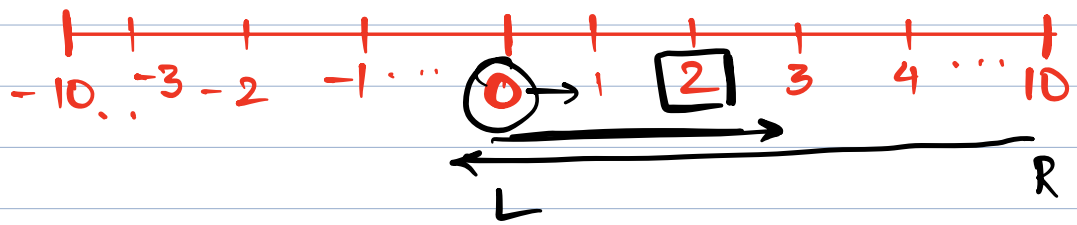
range(1, 7, 2) ⟶ 1, 3, 5 (1, 2, 3, 4, 5, 6)

range(11, 13, 4) ⟶ 11

range(10, 0, -1) ⟶ 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

Stop - 1

0



list(range(5)) ⟶ O/P ⟶ [0,1,2,3,4] ✓

empty list=[ ]
color = ["red", "orange"] " " ✓

Have a list of all numbers divisible by 5 till 125.

divisible_5 = list(range(5,126,5))

H/W :- all numbers that are divisible by 2 from 310 till 680)

ACCESSING

numbers=[1,2,3,4,5]

Index 0 1 2 3 4
-5 -4 -3 -2 -1

print(numbers[3]) ⟶ 4

print (numbers[-4]) ⟶ 25

If you want to access multiple elements

↓

SLICING

Syntax:- list_name [start (0) : stop (0) : step (0)]

numbers = [1,3,5,6,7,8,9,11,44,1,3,5,7]

numbers [0:5] ⟹ [1,3,5,6,7]

start, stop

numbers[:2] ⟹ [1,3]

stop

numbers[2::1] ⟹ [5 ——— 7]

start, stop, step

MODIFICATION

1. fruits = ["apple", "mango", "orange"]

fruits [1] ⟹ "mango"

```
fruits[1] = "syed"
print (fruits) ⇒ ["ap ", 'syed', "orange"]
```

```
fruits = ['A', 'B', 'M', 'D']
fruits[1:3] = ["Blueberry", "cherry"]
print (fruits)
        ⇒ ["Apple'["Blueberry", "cherry", "Drayx"]
```

LIST OPERATIONS
CONDITIONAL STATEMENTS
LIST COMPREHENSION
COMBINING LISTS
SORTING
REVERSING

} LIST (1HR) + TUPI30MIN) ⇒ CODING

9:30 AM ⟶ 9:00 – 10:00

10:20

10:30