PYTHON PROGRAMMING
GATE DA & DSA

Agenda:

- Variables
- Data Types
- User Input

} Operators

Python Program → print variables

Reference Text Books:
1. Head First Python
2. Let us Python - Yashwant Kanetkar

Naming and Using Variables

When you're using variables in Python, you need to adhere to a few rules and guidelines. Breaking some of these rules will cause errors; other guidelines just help you write code that's easier to read and understand. Be sure to keep the following rules in mind when working with variables:

- Variable names can contain only letters, numbers, and underscores. They can start with a letter or an underscore, but not with a number. For instance, you can call a variable message_1 but not 1_message.
- Spaces are not allowed in variable names but underscores can be used to separate words in variable names. For example, greeting_message works but greeting message will cause errors.
- Avoid using Python keywords and function names as variable names. For example, do not use the word print as a variable name; Python has reserved it for a particular programmatic purpose.
- Variable names should be short but descriptive. For example, name is better than n , student_name is better than s_n , and name_length is better than length_of_persons_name.

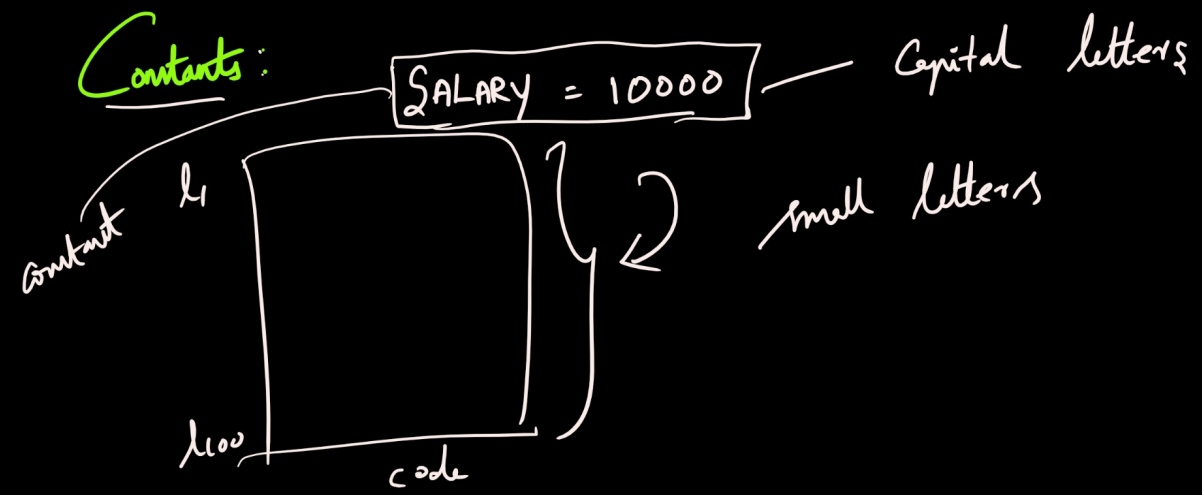
Usage of Semicolon (;)

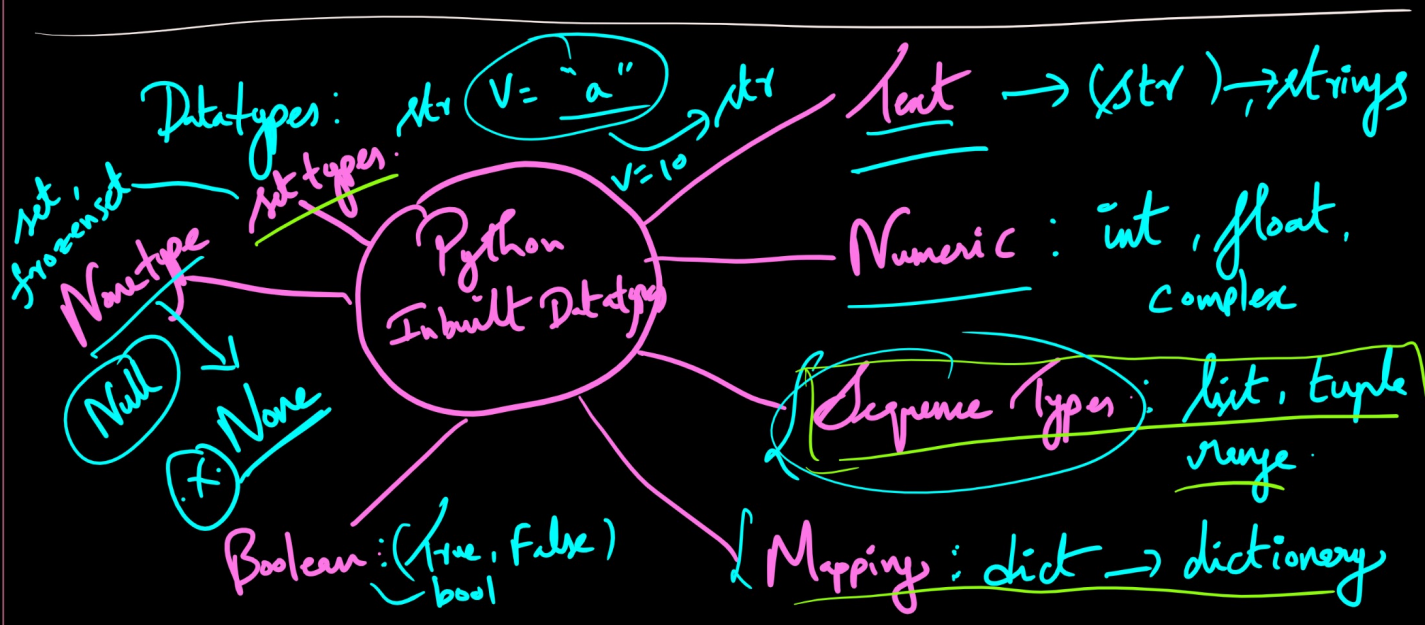
C++ ⇒ cout << "print something" ;

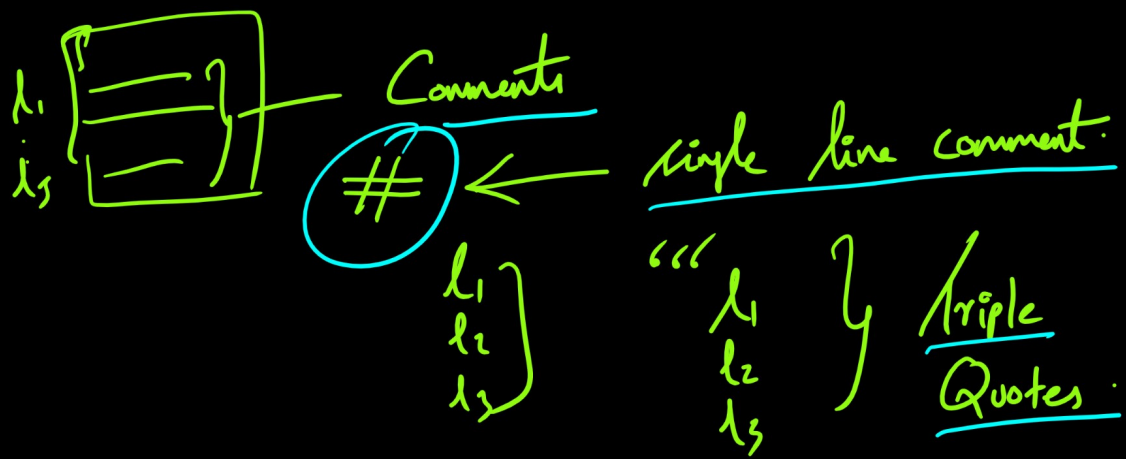
Python → Semicolon is not required

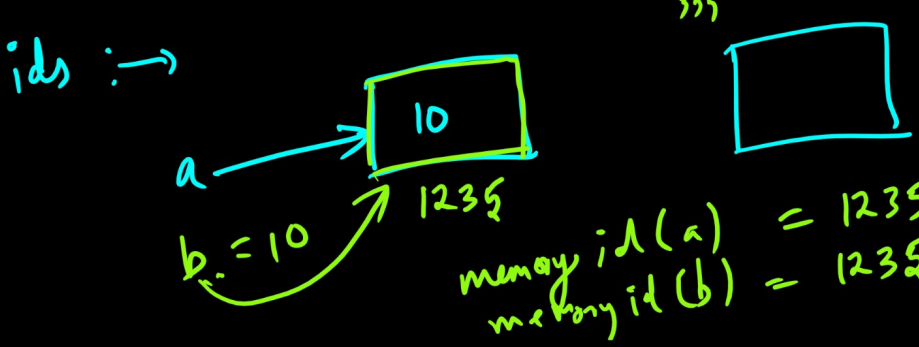
L1 → print ("abc") ;
L2 → print ("Venky")

Indentation : { }] all code inside this bracket belong to particular section


Constants:
SALARY = 10000
Capital letters
constant
l1
l100
code
small letters


Datatypes:
str
V = "a"
str
V = 10
Text → (str) → strings
set, frozenset
set types:
Nonetype
Python Inbuilt Datatypes
Numeric: int, float, complex
Null
None
Sequence Types: list, tuple, range
Boolean: (True, False)
bool
Mapping: dict → dictionary


l1
l3
Comments
#
single line comment:
l1
l2
l3
"""
l1
l2
l3
Triple Quotes:


ids :→
10
a
1235
b = 10
"""
memory id(a) = 1235
memory id(b) = 1235

id: returns same id only on certain cases:

Small integers → -5 to 256
Small strings

250

a = 250
b = 250
id(a) = 1356
id(b) = 1356

a → 250 (1356)
b = 250

258

a = 1356

Datatypes
- Strings (text) — intro
- Numeric
  - int
  - float
  - complex

Plenty of operations

a = 12
b = 5

a/b = 12/5 = 2.4

a//b = 12/5 = ⌊2.4⌋ = 2

2.1 floor(2.1) = 2
2.2 floor(2.2) = 2
2.3 floor(2.8) = 2

2.99

2 3

a//b = 14/5 = ⌊2.8⌋ = 2

a b c b > a

a/b

a//b

: import math 12/5

math.floor(a/b) → 2

math.ceil(a//b) → 3

a = 12.2

b = 12

a - b → Python → 12.2 − 12 → 0.2

128

→ 8 × 10^0

→ 2 × 10^1

→ 1 × 10^2

Base 10 = 8 + 20 + 100 = 128

12 → 2 × 10^0 + 1 × 10^1

Binary form

12 → 1100

1100

→ 0 × 2^0 = 0

→ 0 × 2^1 = 0

→ 1 × 2^2 = 4

→ 1 × 2^3 = 8

12

1/8 = 0.125 → 1/10

12.5

$\rightarrow 5 \times 10^{-1} = 0.5$

$\rightarrow 2 \times 10^{0} = 2$

$\rightarrow 1 \times 10^{1} = 10$

12.5

0.125

$1 \times 10^{-1} = 0.1$

$2 \times 10^{-2} = 0.02$

$5 \times 10^{-3} = 0.005$

$1/8 \Rightarrow 0.125$

0.125

$1/3 \Rightarrow 0.3333333\ldots\ldots$

Value 0.1 → represent in binary form.

0.1 (binary) → 0.5 → base 10

0.1

0×2^{0}

$1 \times 2^{-1} = 1/2 = 0.5$

0.01x

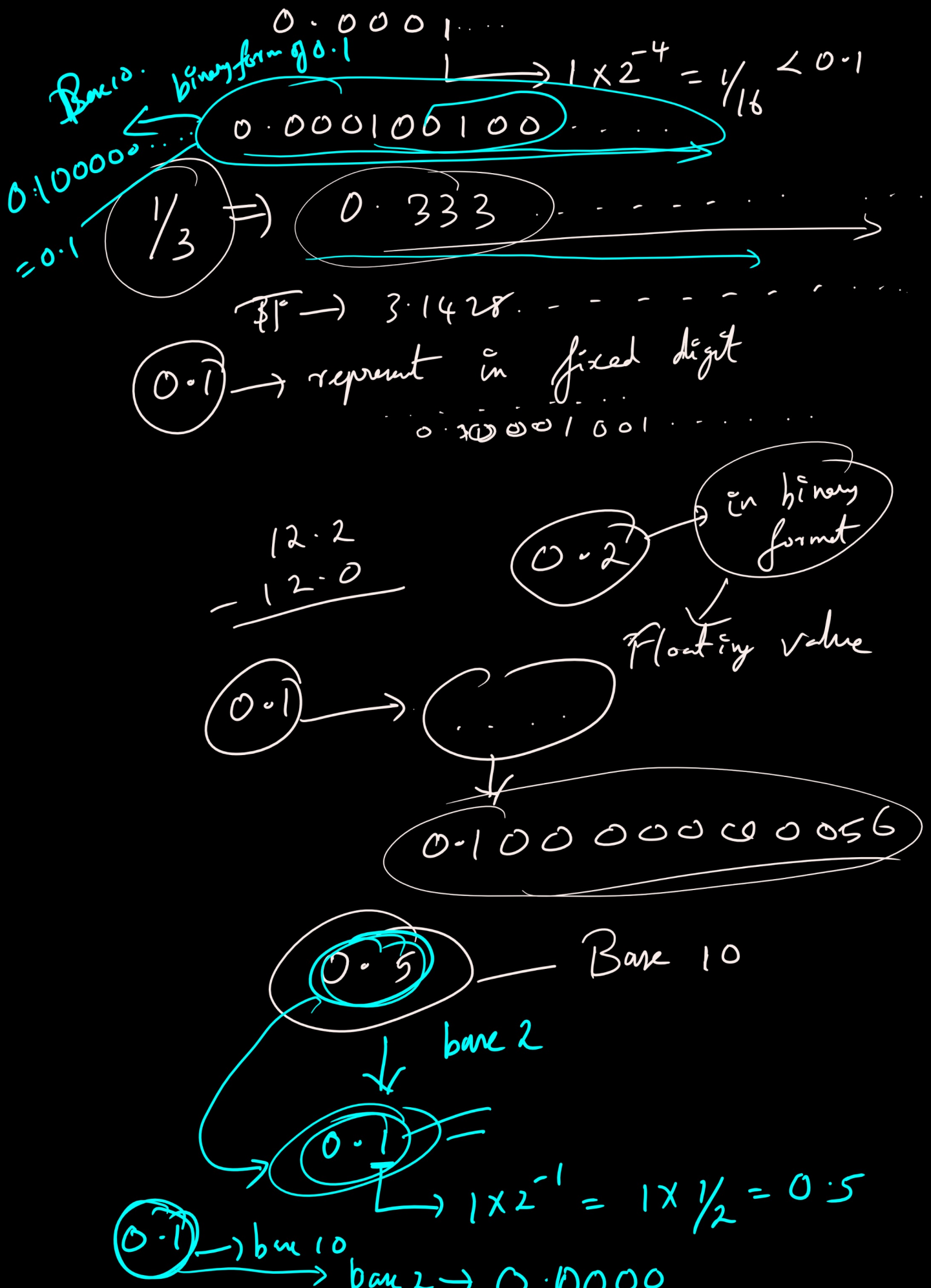
$0 \times 2^{-1} = 0$

$1 \times 2^{-2} = 1/4 = 0.25$

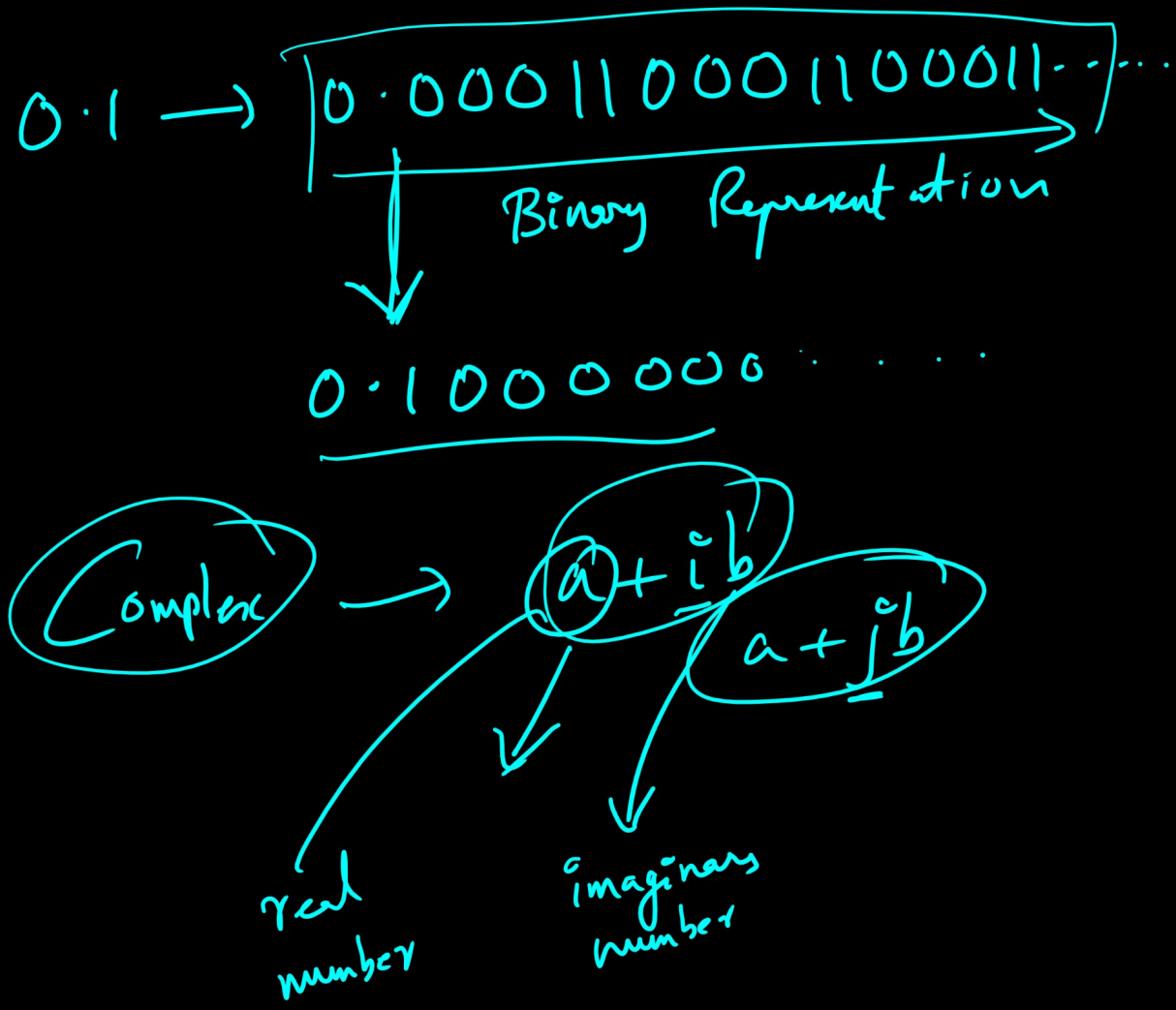
0.001

$1 \times 2^{-3} = 1/8 = 0.125$

> 0.1


0.0001....
Basic binary form of 0.1
$1 \times 2^{-4} = 1/16 < 0.1$
0.0001100100....
0.100000....
=0.1
1/3
0.333.....
$\pi \rightarrow 3.1428.....$
0.1 → represent in fixed digit
0.xxxx1001.....
12.2
-12.0
0.2
in binary format
Floating value
0.1
0.1000000000056
0.5
Base 10
base 2
0.1
$1 \times 2^{-1} = 1 \times 1/2 = 0.5$
0.1 → base 10
→ base 2 → 0.0000

Floating Point - Python Representation:
https://docs.python.org/3/tutorial/floatingpoint.html



[1, 2, 3, 4]

[2.5, 2.7, 2.9]

lists [2, 5, 9, -2, 2.5, 2.7, 2.9, "string"]

"venky", "a", [2, 9]]

Mutable → lists are mutable

[2 (5) (9)] ← add a new element

n = [2, 5, 9] [2, 8, 9] → [2, 5, 9, 10]

new_list = [2, 5, "Venky", 12.5, -12]
0 1 2 3 4

new_list_2 = [2, 2, 2, 2, 5, 5, 10, 10, "Venky", "Venky"]

Duplicate is allocated.

new_list[0] = 2
[1] = 5
[2] = "Venky"
[3] = 12.51

list1 = ["a", "b", "c", "d", "e"] — 5 element
0 1 2 3 4
-5 -4 -3 -2 -1
0 → 4
n
0 → n-1

list1[0] = "a"
list1[-1] = "e" ; list1[1] = "b"

[2,4,6,10,12.4, 12.8, "Venky", "Gate", "Python", 128, "IPL", "VIRAT"]
0 1 2 3 4 5 6 7 8 9 10 11
-12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1

12 element
0 → 11
1 → -12

list(p) = 4 ⟵
list[-1] = "Virat" ⟵

n elements

+ve → 0 to n-1

-ve → -1 to -n

new_list → [1, 2, 3, "Virat", "Vicky" / "Venky"] → updating the list but memory id doesn't change

memory id - 1235

Mutability

new_list[-1] = "Vicky"

name = "Venky"

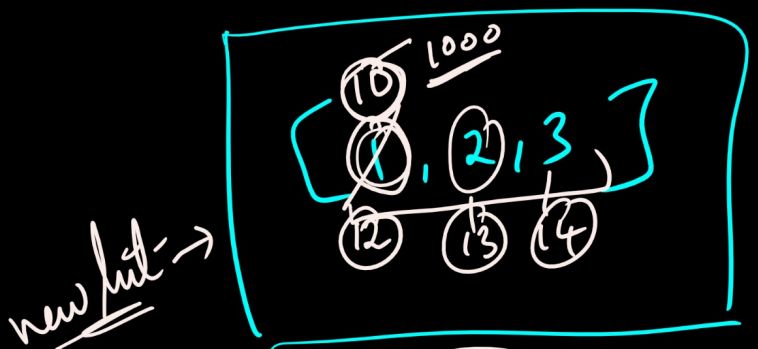
0 1 2 3 4

-5 -4 -3 -2 -1

Strings
↳ immutable

Lists
↳ Mutable.

[ ] → square brackets.



Memory 1235

1235

Tuple: → ( )

(1, 2, 3, 4, "Virat", 12.5, 15.0, "Venky")

immutable (X)

→ you can't update the existing tuple

("Vicky", 12.5, "Venky")] Tuple.

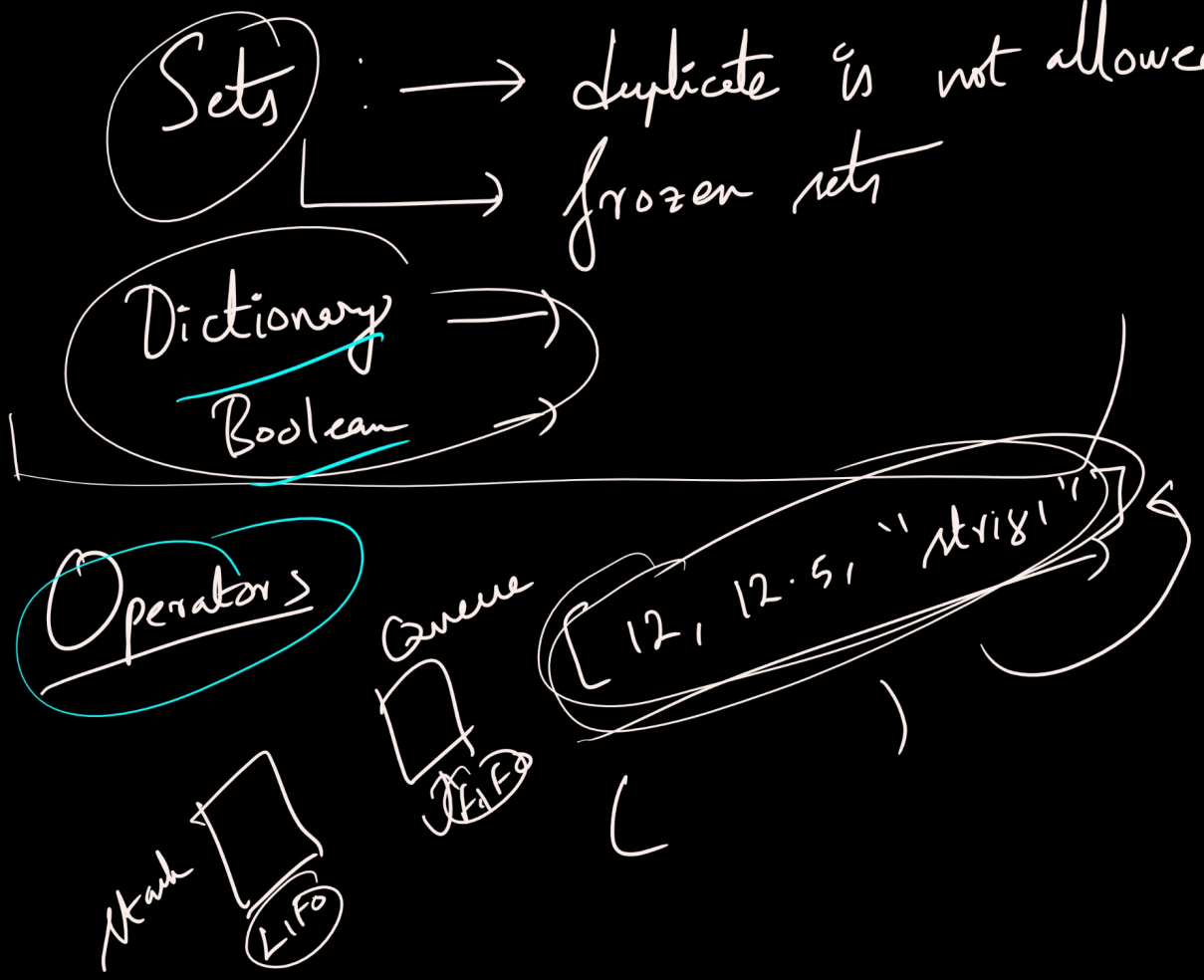
0	1	2	3	4
-5	-4	-3	-2	-1

Tuple →	← Lists
duplicates allowed	
you can have tuple inside tuple	tuple inside lists
you can have list inside tuple	lists inside list

Lists → Mutable

Tuple → Immutable

String →


Sets
duplicate is not allowed
frozen sets
Dictionary
Boolean
Operators
Queue
FIFO
12, 12.5, "string"
Stack
LIFO