Python Intro

21 May 2024 02:58 PM

Day-1

Language:

Language is used for communication.

Classification of Languages:

In terms of computer, we have 2 types of languages

Low level language --> understandable by the system

High level language --> understandable by humans

To convert HLL to LLL, we use:

- Compilers
- Interpreters

Process:

While compiling and executing, we'll perform 2 process i.e., compilation and interpretation

Compilation:

is a process of converting high level language to low level language

Interpretation:

is a process of executing converted low level language to get some output

Classification of high level languages:

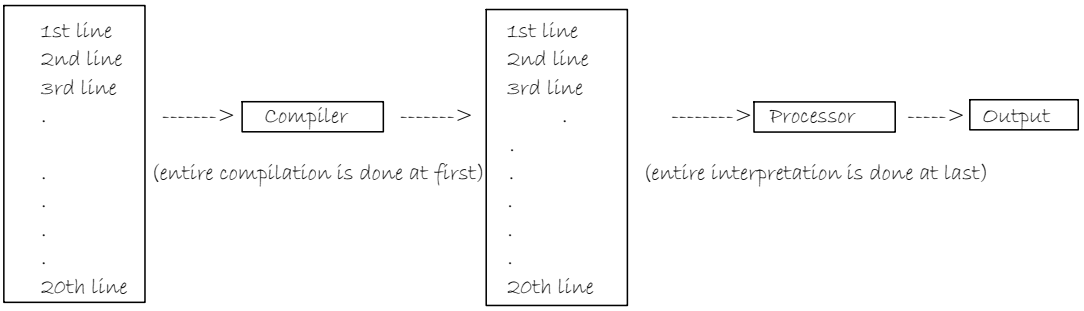
High level languages are classified into 2 types based on the **Converters**:

1) Programming High level language
2) Scripting High level language

Programming high level language:

1) compilers are used
2) entire set of instructions will be converted at a time to binary format and entire converted code will be executed at one time.

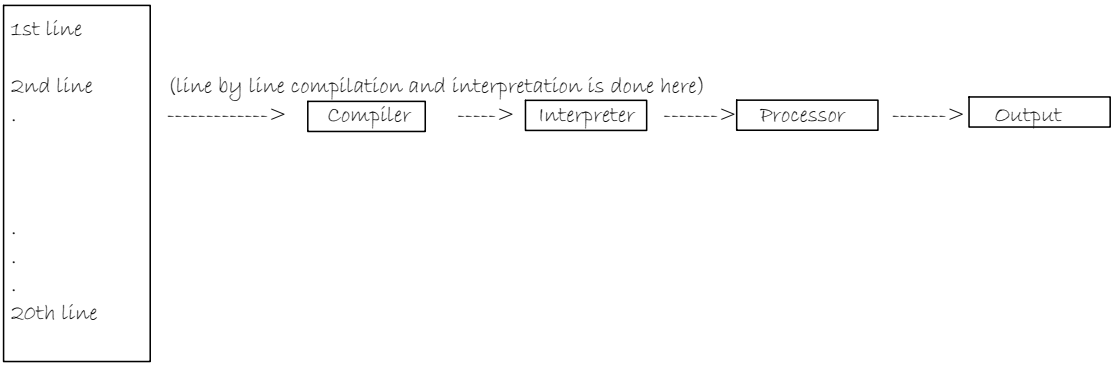
Eg: Java, C, C++



Scripting High Level Language:

1) Interpreter is used
2) Line by line compilation and interpretation is done

Eg: Python, Javascript, PHP, Ruby



Day-2

Definition/Features of Python:

1)Python is Open source, High level, Interpreted, Scripting Language

2)Python is a Dynamic typed language -
3)Python supports Functional Programming and Object Oriented approaches for solving programming problems
4)Python has more built-in libraries
5)Python is case-sensitive language
6)Python is very easy to understand and learn.

Day-3

Desktop Apps --> Java
Web Development, AI, ML, Data Science, IOT, Automation --> Python

The fields where python is used are:

- Web development
- Artificial Intelligence
- Data science
- Machine Learning
- IOT (Internet of Things)
- Automation.. etc

Editors of Python:

- PyCharm
- Visual studio code
- Atom
- Jupyter notebook
- Sublime text
- Editplus

Adding the path is to: Register the software with system environment variables

Installation of Python software:

- Search for python.org in browser
- Click on downloads --> click on download python 3.12.3 button
- One python.exe file will be downloaded
- Double click on the exe file
- Before clicking on the install in the wizard, please check on Add Path checkbox
- Click on Install Now

Day-4 : IDLE,Shell, Python Module etc.

IDLE (IDE):

IDLE stands for Integrated Development Learning Environment
IDLE is and environment which is integrated with all the default implementations of python language

We can utilise the IDLE in 2 ways:

1. **Python Shell**
   1. It is an interactive console
   2. We can execute only one statement at a time in Python shell (and can't be saved...?)
2. **Python Module**
   File with .py extension with the collection multiple python statements

Steps to create Module from Shell:

Open IDLE Shell
Click on New and click on New File(CTRL+N)
Save the file without giving extension

Command for executing python module: Press F5

Syntax for executing python module through command prompt:

python modulename.py
or
py modulename.py

Note:

It is mandatory to provide the extension as .py while creating python modules from other editors

Day 5:

Tokens:

Tokens are the **Essential Elements** for writing a Python Program

Tokens of Python:

1. Keywords
2. Identifiers

3. Variables
4. Data types

1. Keywords:

1) Keywords are reserved/Built-in/pre-defined words which are defined for doing some specific task
2) True, None, False are in Title case and the remaining words are in lower case
3) We can't change the functionality of keywords
4) In python we have 35 keywords
5) To see the list of keywords, follow the below commands:
    1. Import keyword
    2. keyword.kwlist
       The list of keywords is displayed as follows:
       ['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', ' finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Import Keyword:

1) Import keyword is used to perform importing process
2) Importing is the process of accessing the contents of another module into the current module

Day 6

2. Variables:

1) Variable is a name given to the Memory Allocation (address)
2) The value stored in a variable might get varied (changed)

Syntax for declaring single variable:

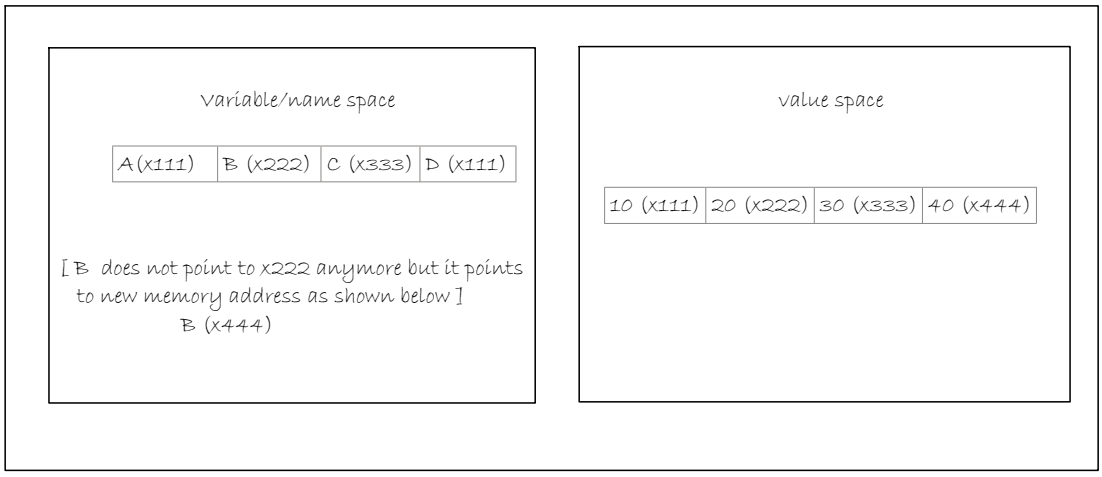
variableName = value [right to left (<----) is the direction of execution]

Note:

1. If we assign multiple variables with same integer value, then they will point to same memory address
2. If we assign a new value for an existing variable, then it (the variable) will point to the new memory address
3. Once we create a variable, 2 spaces are created in memory (Variable/Name space and Value space)

Suppose,

```
a=10
b=20
c=30
d=10
b=40
```




```
id(a) = 55224
Id(b) = 55544
Id(c) = 55864
id(d) = 55224
id(b) = 56184
```

Garbage Collection:

Garbage collection is used to delete memory of unused values

Drawbacks when garbage collection is done by developers:

1. Deleting the memory before the completion of its usage
2. Not deleting the memory even after the completion of its usage

PMM:

1. In python Garbage collection is done by PMM

2. Python Memory Management will delete the value once it has zero reference counts

ID function:

It is used for getting the address of the variable

Syntax:

id(variable_name)

Syntax for executing multiple variables at a time in python:

Variable_name1, vaiable_name2, variable_name3, ... = value1, value2, value3, ...

Day - 7

Syntax for creating new variable with existing variable value:

new_variable = existing_variable

Eg: n = 30
w = n
|
|
V
W = 30 (is internally assigned)

Syntax for swapping the values of variables:

Var1, var2 = var2, var1
Eg: a, b = 10, 20
a, b = b, a (--> swapping the vars)

Day - 8 28 - 05 - 24

3. **Identifiers:**

Identifier is a name with which we can identify variables, functions or classes.

Identifiers-> Variables
Functions
Classes

Note: All identifiers are not variables but all variables are identifiers

Rules that must be followed while declaring the Identifiers:

1. We should not use Keywords as identifiers
2. As a first character of identifier, we should not use a number.
3. Other than underscore, we cannot use any special characters in identifiers because each and every other special character has some functionality in python.
4. We can use combination of alphabets, numbers and underscore.
5. Standard length of identifiers is 4 to 15 characters, but we can have a maximum of 79 characters in identifiers.

Standards of Identifiers:

Variables ---> use only lower case
Functions ---> One word ---> lower case
Multiple words ---> first word lower case and remaining words in title case
Classes ---> Title case

4. **Data Types:**

Data types are used for defining the type of data that we are going to store in variable.

Classification of data types:

Scenario-1: Based on **Number of values** we store in variable

1. Single valued data type
   1. Number Data type
      a) integer
      b) float
      c) complex
   2. Boolean data type
      a) True
      b) False
2. Multi valued/collection/group data types
   a) string
   b) list
   c) tuple
   d) set
   e) dictionary

Scenario-2: Based on **behavior of value** stored in variable

1. Immutable datatypes
    a) All Single Valued Data Types
    b) String
    c) tuple
2. Mutable datatypes
    a) list
    b) set
    c) dictionary

1.Single valued data type:

We can store only one or single data in a variable

1. Number DT:

a) int: The numbers without decimal points
b) float: The numbers with decimal points.
c) complex: Complex numbers are the combination of real part and imaginary part.
a + bj where j=square root(-1)

2. Boolean DT:

- These DTs are used for defining the **yes** or **no** type of data
- True keyword is used for defining True Boolean type
- False keyword is used for defining False Boolean type

type function: It is used for returning the **type of data stored in a variable**
syntax : type(variable)

Note:
1.In python everything is considered as an object
2.We can store only one value or one memory address at a time in a sub memory address.

Examples:

```
a=90                    d = True
type(a)                 type(d)
<class 'int'>           <class 'bool'>

b=15.5                  e = False
type(b)                 type(e)
<class 'float'>         <class 'bool'>

c=2+8j
type(c)
<class 'complex'>
```

Day 9	29-05-24

2. **Multi valued or collection data:**

In this category, we will be storing more than one data item or value.

In order to define multiple individual elements into one collection we have to use boundaries.
1. If we store multiple values inside a variable, memory will be divided into as many sub memory addresses as the number of elements present in the variable.
2. We can identify these sub memory addresses by using Index Positions

In python, we have both Positive and Negative Index Positions

Positive Index Positions:

Direction ---> Left to right
Range ---> 0 to n-1 where n is the length of the collection.

Negative Index Positions:

Direction ---> right to left
Range ---> -1 to -n where n is the length of the collection.

Classification of CDT (Collection Data Types):

CDT are classified into 2 types:
1. Ordered CDT
2. Unordered/Random CDT

1) Ordered CDT:

In ordered CDT, in the memory data will be stored in the same order as have defined in the variable.
Eg: String
List
Tuple
Dictionary

2) Un-Ordered CDT:

In un-ordered CDT, in the memory data will be stored in the random order.

Eg: Set

1. String:

1. String is collection of individual elements which are enclosed in a pair of quotation marks.
2. String is immutable data type
3. String is Ordered CDT
4. In string, Indexing and Slicing is possible.
5. String is a fixed length data type as it is immutable.

For representing single line strings, we use single, double or triple quotes
For representing multi-line strings, we use triple quotes.

Syntax for declaring the single line string:

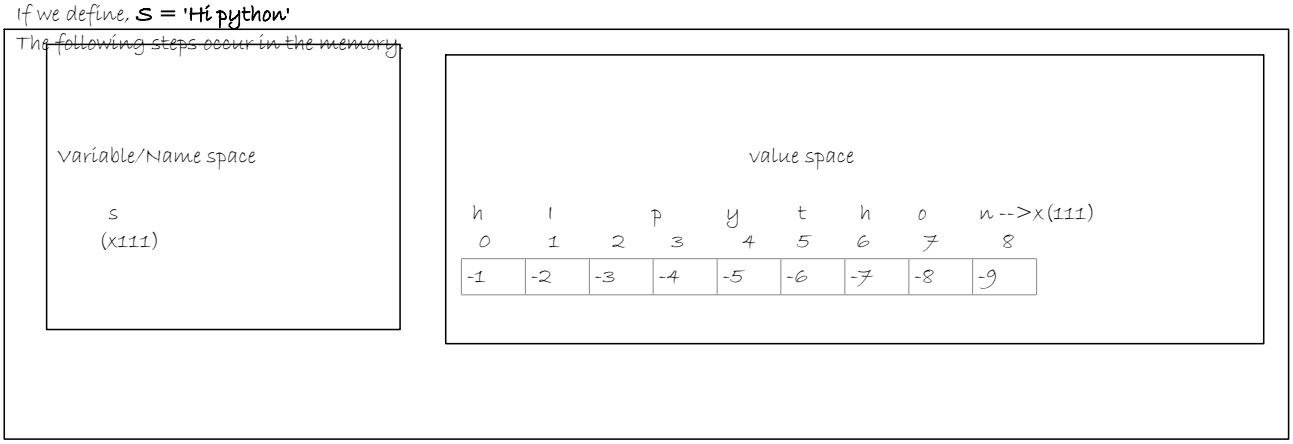
Variable-name ='element-1,element-2,.....element-n'
Variable-name ="element-1,element-2,.....element-n"

Syntax for declaring multi-line strings:

Variable-name = '''element-1element-2element-3.....
.....element-n'''

If we define, **S = 'Hi python'**

The ~~following steps occur in the memory.~~



Indexing:

Indexing is a process of extracting single element from a given collection

Syntax for indexing:

Variable-name[index-position]

We can give both positive and negative index position
But internally, Negative indexes are converted into its respective positive index positions only

Positive Indexing:

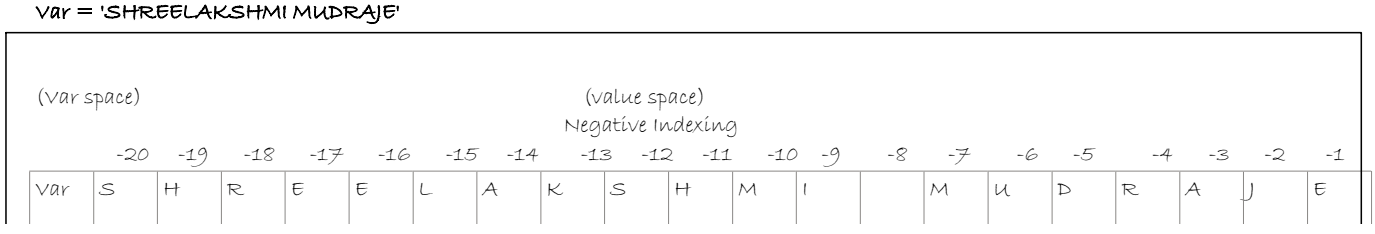
The process of extracting single element from a given collection by using Positive Index Positions

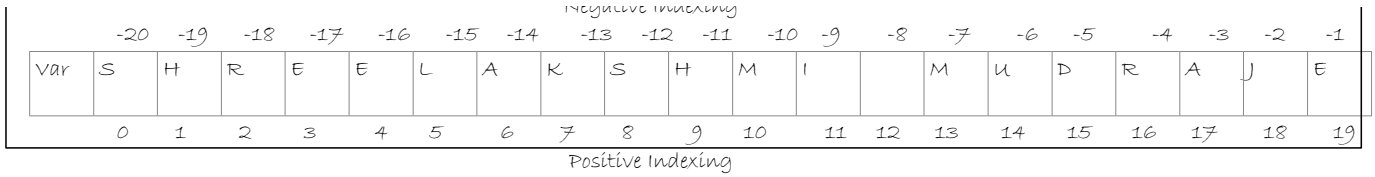
Negative Indexing:

The process of extracting single element from a given collection by using Negative Index Positions

S = 'hi python'
s[4] = 'p' s[-4] = 't'
s[8] = 'o' s[-3] = 'h'
s[1] = 'a' s[-10] = 'h'
s[9] = 'n' s[-5] = 'y'
s[3] = ' '

Var = 'SHREELAKSHMI MUDRAJE'





var[1] = H
var[-5] = D
var[-18] = R
var[8] = S
var[U] = 14

Day 10 | 30-05-24

Slicing:

It is a process of extracting multiple elements from a given collection.
We can perform slicing in two ways.

1. Positive slicing'
2. Negative slicing

1. **Positive Slicing:**

It is the process of extracting multiple elements in the left to right direction.

Syntax for performing positive slicing:

Variable-name[start-index : end-index : updation]

While performing positive slicing, the interpreter checks for condition mentioned below. i.e., SIP < EIP (start_index_pos < end_ind_pos)
In Positive slicing, default values for

Starting index = 0
Ending index = len(collection)
Updation = 1

Eg: S = 'Hai python'

S[0:2:1] = 'ha'	S[0:9:4] = 'hpo'
S[0:3:1] = 'hai'	S[-8:3:1] =
S[4:10:1] = 'python'	S[-8:9:1] = 'l pytho'
S[4:9:2] = 'pto'	S[-10:5:2] = 'hip'
S[0:9:2] = 'hipto'	S[-6:-1:2] = 'pto'
s[-10:7:3] = 'h t'	S[0:-1:4] = 'hp'
S[-3:0:2] = '	S[-3:0:0] = Val.Err-> Slice step can't be zero
S[2:-2:2] =	S[4:10:1] =
S[4:10] =	S[4:10:] =
S[4::] =	S[::] =
S[::3] = 'h tn'	

VAR = 'SHREELAKSHMI MUDRAJE'

	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
	S	H	R	E	E	L	A	K	S	H	M	I		M	U	D	R	A	J	E
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

VAR[3:8:1] = 'EELAK'
VAR[0:-5:3] = 'SEAH '
VAR[-9::2] = 'IMDAE'
VAR[:8:] = 'SREA'
VAR[::4] = 'SES R'
VAR[15::] = 'DRAJ'

Day 11 | 31-05-24

Note:

If the specified index position is more than the length, then

In indexing: Error ---> IndexError
In slicing: It will extract till the end of the string

Negative Slicing:

I t is the process of extracting multiple elements in the right to left direction

Syntax for performing negative slicing:

Variable-name[start-index : end-index - 1 : updation]

In negative slicing, default values for Start-index = -1

End-index = -(len(collection) + 1)

For updation, there is no default value; we need to provide the value for updation in negative values.

Note:

It is mandatory to pass updation value in case of negative slicing (in negative values only).

In case of negative slicing, interpreter will check for below mentioned condition:

Consider S[-1:-5:-1]

SIP < EIP

-1<-5 ---> True -- > extracts 't'

-2<-5 ---> True -- > extracts 'ht'

-3<-5 ---> True -- > extracts 'oht'

-4<-5 ---> True -- > extracts 'noht'

-5<-5 ---> Not True --> final 'noht'

Eg: Consider S = 'hai python'

s[-:-5:-1] = 'noht'	s[7:2:-1] = 'htyp'
s[-3:-7:-1] = 'htyp'	s[7:-9:-1] = 'htyp i'
s[-2:-7:-2] = 'otp'	s[-1:-6:] = ''
s[-6:-10:-3] = 'pa'	s[-1:-10:-1] = 'nohtyp ia'
s[2:8:-1] = ''	s[-1:-11:-1] = 'nohtyp iah'
s[0:7:21] = 'h'	s[::-1] = 'nohtyp iah'

Note: S[::-1] ---> SYNTAX FOR REVERSING A STRING

Day 12	01-06-24

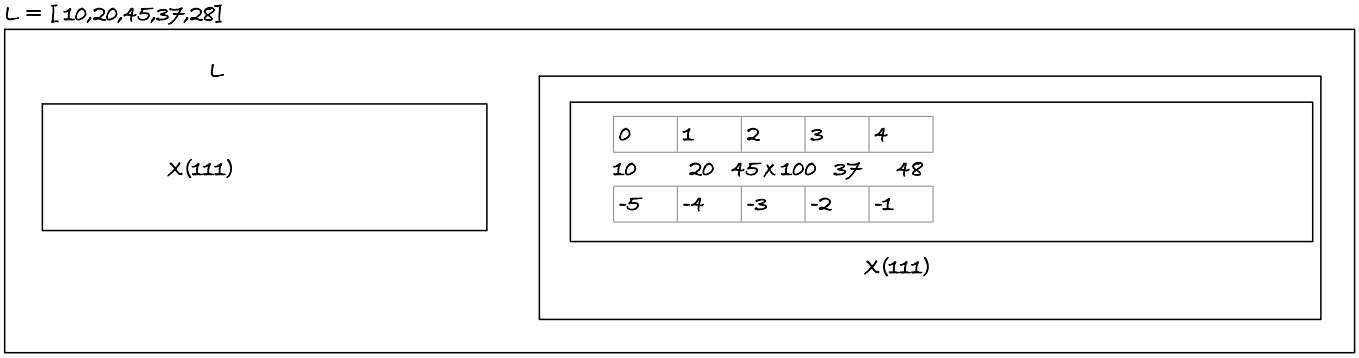
1. **Lists:**

1. List is a Collection of both homogeneous and heterogeneous data, in which each and every element is separated by a comma (,) operator and enclosed in the pair of [] (square brackets).
2. List is a mutable data type
3. List is an Ordered CDT
4. In list Indexing and slicing are possible.
5. List is a Variable length data type as it is mutable.

Syntax for declaring List data type:

Variable-name = [element-1, elemet-2, element-3, ..., element-n]

L = [10,20,45,37,28]



type(L) = < class 'list' >

L[3] = 37	L[1] = 20
L[-5] = 10	L[-1] = 28
L[1:4:] = [20,45,37]	L[1:4:2] = 20,37
L[::4] = [10,28]	L[::2] = [28,45,10]
L[-2:-5:-2] = [37,20]	

Behaviour of DTs:

Immutable DTS:

In case of Immutable DTs, we cannot modify its value space.

Mutable DTs:

In case of mutable DTs, we can modify its value space.

Syntax for modifying the value space by indexing:

Variable-name[index-position] = new-value

Syntax for deleting the value space by indexing:

del variable-name[index-position]

L[2] = 100

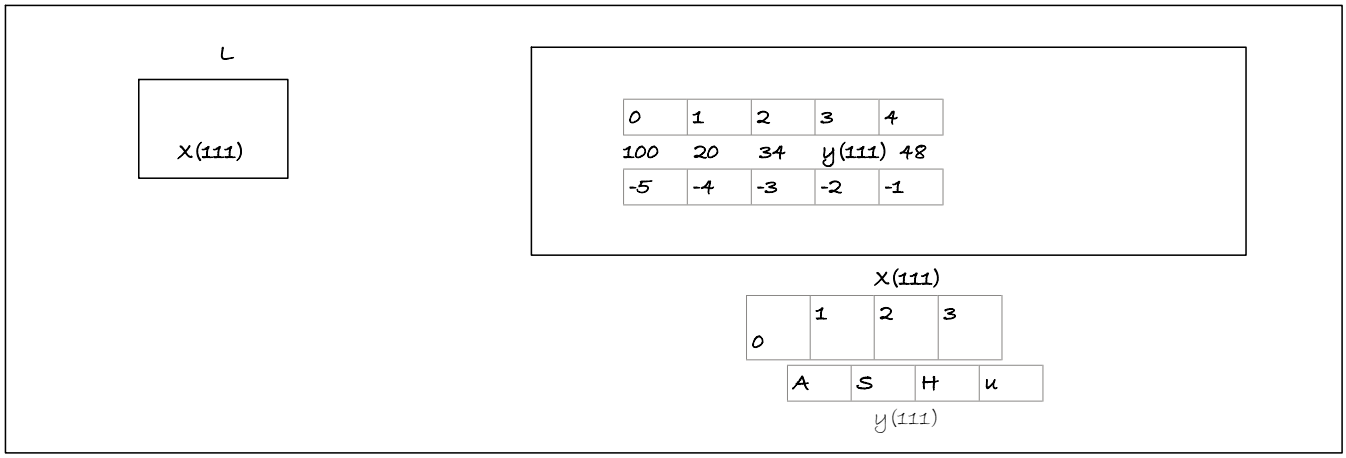
L --> [10,20,100,37,28]

Del L[3] =

L --> [10,20,100,28]

L = [100,20,34,ashu', 89]

L[2] = 34	L[3] = 'ashu'
L[3][0] =	L[3][3] = 'sh'
L[3][1:3:] =	L[3][::-3] = 'ua



Day 13	03-06-24

L = ['ashu', 'shree', 'anu', sara']

Day 14	04-06-24

Difference between List and Tuple:

- Collection Data Type with homogeneous/heterogeneous dt enclosed in [] or (), separated by ','
- [] defines a list and ',' defines a tuple
- To define a single element, ',' is mandatory for tuple and not () and for list, [] is mandatory not ', '
- List is mutable/tuple is immutable
- List is variable length dt and tuple is fixed length data type (dt)
- Both are ODTs

Data Type	Indexing	Slicing
String	String	String
List	We cannot predict the dt	List
Tuple	We cannot predict the dt	Tuple

Tuple data type

1. Tuple is a collection of both homogeneous and heterogeneous data in which each and every element is separated by a ',' (comma) operator and enclosed in a pair of ().
2. Tuple is immutable CDT
3. Tuple is an Ordered CDT
4. Comma operator defines the tuple not the parenthesis
5. Both indexing and slicing can be performed in tuple
6. Tuple is a fixed length CDT

Syntax for defining multiple values in tuple:

Variable-name = (element-1, element-2,,,,,,element-n) or
Variable-name = element-1, element-2, element-3, ,,,,,element-n

Syntax for defining single value tuple:

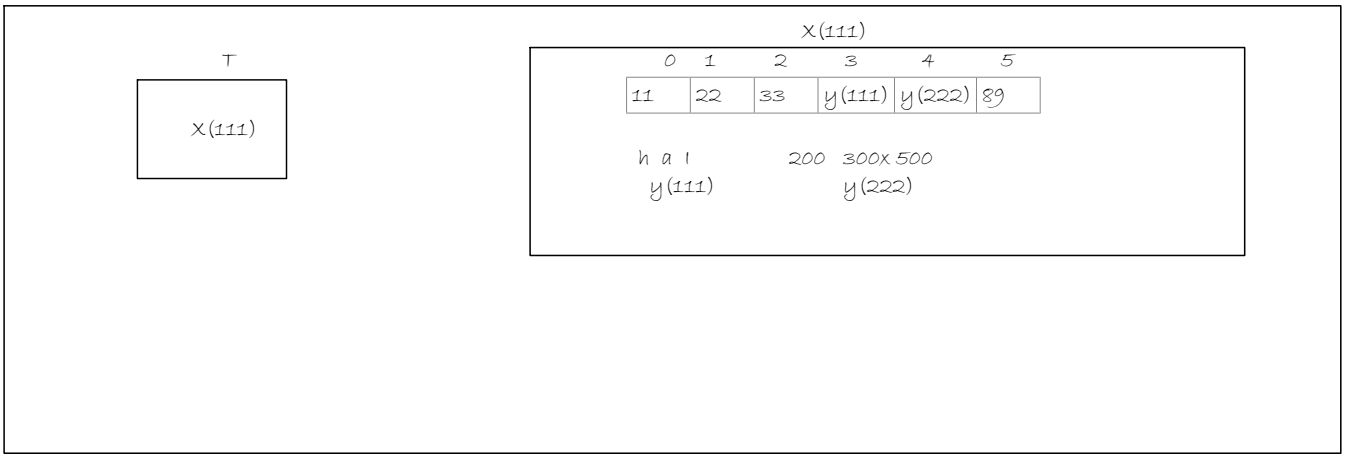
Variable-name = element-1,
or
Variable-name = (element-1,)

T = 11,22,33,44,55,66

T[2] =33 T[-2:-5:-2] = (55,33)

T[-1] =66 T[1:4:1] = (22,33,44)

T= 11,22,33,'hai',[200,300], 90,89
T[3] = 'hai' T[4] = [200,300]
T[3][1] = 'a' T[4][0] = 200
T[4] = [200,300] T[4][1] = 500
T[4][0] = 200 T[4] = 'hello'
T[4][1] = 500 | ---- > ERROR



Questions - HW

- Difference b/w list and tuples
- Create a tuple with nested list and nested tuples and perform minimum of 5 positive and negative indexing and slicing each; Also modify the nested list

```
t = (15,
   'asdfgh',
   ( ('apple', 'mango', 'cherry'), (32, 43.47,'lmnop', 16), [101,202], 89,100),
   [['list', 'tuple', 'dict'], 75, 25, (['a', 'e'], 'i', 'o', 'u')],
   80,
   400,
   ['rashmi', 'raksha' ])
```

Day 15	05-06-24

Set :

1. Set is a collection of both homogeneous and heterogeneous data in which each and every element is separated by , operator and enclosed in the pair of {}
2. Set does not allow duplicates (we can only store unique values)
3. Set is an Ordered Data type
4. As set is random ordered, we cannot perform indexing and slicing
5. Set is mutable data type
6. In set we can store only immutable data types
7. Set is a variable length data type

S = {13,34,13,90,89,24} s = {2,1,3,4,9,8,7,5,6,10}
s--> {34,24,89,90,12,13} s = {1,2,3,4,5,6,7,8,9,10}

Syntax for declaring a set:

Variable-name = {elememt-1, element-2,...., element-n}

Day 16	06-06-24

H.W.
List and set
Tuple and set

Examples for defining a set:

S = {33, 'hai', '22,78,11,12,44} s = {11,(33),33,89,56}

S = {11,(33,),33,89,56 |.....> s = {56,33,11,89}

Dictionary:

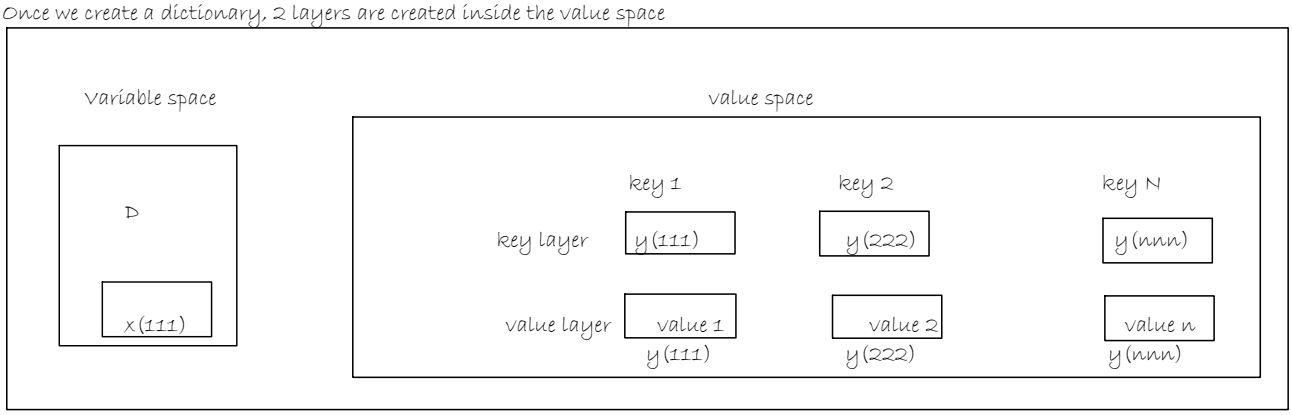
1. Dictionary is a collection of key and value pairs
2. In dictionary, key and values are enclosed in a pair of {}
3. Each and every key, value pairs are separated by using ',' operator
4. Each and every keys and values are separated by using ':' operator
5. Dictionary is mutable data type

6. As the data is in the form of pairs, we cannot perform indexing and slicing

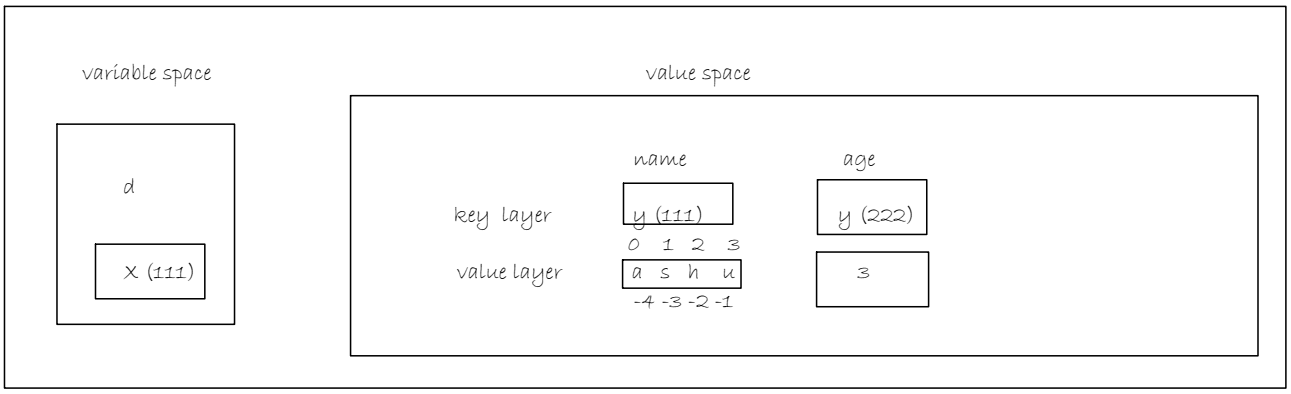
syntax:

Variable-name = {'key-name':value1, 'key-name2':value2.....,'keyname-n':value-n}

Once we create a dictionary, 2 layers are created inside the value space



D = {'name': Ashu, 'age':3}



Properties of keys in dictionary:

1. We can use **immutable data types as keys** but its advised to use only strings as keys because keys are used for defining the values
2. Keys are case sensitive
3. If we declare duplicate keys, then the recent value will be assigned to that key

Properties of values in the dictionary:

1. We can use duplicate values as values in a dictionary
2. we can use any type of data in values

Syntax for accessing value from dictionary:

Variable-name[key-name]

Syntax for modifying the values from a dictionary:

Variable-name[key-name] = new-value

Note:

If a specified key is present in the dictionary, then it will update the value
Else it will create a new value pair in given dictionary

Syntax for deleting the element from a given dictionary:

del variable-name['key-name']

```
D = {'name': Ashu, 'age':3}
d['name'] = 'Ashu'                d['age'] = 3
d['Ashu'] --> Key Error           d['age'] = 4 --->existing key updated successfully
d['mobile'] = 653654          ----> d = {'name': Ashu, 'age':3,'mobile' : 653654}  ---> dictionary after updation of a new key

del d['mobile']                  ----> d = {'name': Ashu, 'age':3}  ---> dictionary after deleting a key
d['name'] = 'Ashu'        d['name'][0][::2] = 'us'
d['name'][0] = 'A'

details = {'name': 'Shreelakshmi',
           'age': 24,
           'mobile': {8762280584, 7676895376},
```

'achievements' :{'singing': 'Senior Grade', 'Dancing': 'Hip Hop'}

List	Set
Enclosed in square brackets	Enclosed in flower brackets
Ordered CDT	Random Ordered CDT
Indexing and slicing	Indexing and slicing not possible
Can store duplicate values	Allows only unique values
Both homogeneous and heterogeneous DTs	Only immutable data types can be stored
Both are mutable and hence variable length DTs	

Tuple	Set
CDT enclosed in parenthesis	CDT enclosed in curly braces
Ordered CDT	Random Ordered CDT
Indexing and slicing	Indexing and slicing not possible
Can store duplicate values	Allows only unique values
Both homogeneous and heterogeneous DTs	Only immutable data types can be stored
Both are mutable and hence variable length DTs	

Creation of empty CDT:

Data types	Syntax
String	'',"", ''' ''', str()
List	[], list()
Tuple	(), tuple()
Set	set()
Dictionary	{}, dict()

Type Casting:

It is the process of converting one type of data into another type of data.

Into which data type	From which data type	Syntax
Integer	Float, Boolean values and string containing integers only	int(value/variable)
Float	Integer, Boolean and string containing integer and float values	float(value/variable)
Complex	Integer, float, Boolean and string containing integer, float and complex	complex(value/variable)
Boolean	All data types (bool gives False for 0(zero) and empty CDTs; in the remaining cases it gives True.	bool(value/variable)
String	All the data types	str(value/variable)
List	String, tuple, set and dictionary(considers only keys)	list(value/variable)
Tuple	String, tuple, set and dictionary(considers only keys)	tuple(value/variable)
Set	String, List, tuple (should not have mutable entities as elements) Dictionary (considers only keys)	set(value/variable)
Dictionary	List, tuple, set (data should be given in pairs) (Also while type casting set into dictionary, use only tuple to represent pairs)	dict(value/variable)

****

Day	13-06-24

S = 'I love Biriyani'
' '.join(s.split()[::-1])
o/p: 'Biriyani chicken love I'

Format method:

Format()	'content {} content{}'.format(value1, value2) f' content{val1} content {val2}'	1. It is used for creating dynamic strings 2. We need to create placeholders 3. Placeholders are created by using {}

Eg:

'This is {} and his/her age is {}'.format('Ashu', 3)
o/p: 'This is Ashu and his/her age is 3'
'This is {1} and his/her age is {0}'.format(3,'Ashu')
o/p: 'This is Ashu and his/her age is 3'

'This is {n} and his/her age is {a}'.format(a=3, n='Ashu')
o/p: 'This is Ashu and his/her age is 3'
n = 'Ashu', a = 3
f' This is {n} and his/her age is {a}'
o/p: 'This is Ashu and age is 3'

List Built-in Methods:

Method Name	Syntax of method	Functionality
Count()	Count(value)	1. It is used to count the number of times the given element is repeated 2. Count returns integer
Index()	Index(value, [start_index], [end_index]	1. It is used for returning the index position of a given element 2. If the element is not present it returns Value Error
Clear()	Clear()	It is used for deleting all the elements of given list but not the memory
Copy()		

L = [11,22,33,44,11,55,66]
L.index(11) ---> 0
L.index(11,1) ---> 4
L.inedx(11,5) ---> Value Error: 11 not in list
L.count(11) ---> 2
L.count(110) ---> 0

L = [11,22,33]
L.clear() ---> L = []
Note: del L ---> Printing L gives Name error: name L is not defined

Insertion methods of list data type:

Append()	Extend()	Insert()
Syntax: Append(data) Data can be SVDT or CDT	Syntax: Extend(data) Data can be only CDT	Syntax: Insert(index-position, data) Data cs be SVDT or CDT
It is used for adding element at the last position	It is used for adding element at the last	It is used for adding elements at specified index positions
If CDT is passed as an argument, then entire CDT is considered as a single element	If a CDT is passed as an argument, then it extracts individual elements and adds them at the end of the list.	If specified index position gretaer than the length, then that element will be added at the end of the list.

```
>>> L=[11,22,33]
>>> L.extend(88)
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    L.extend(88)
TypeError: 'int' object is not iterable
>>> L.extend('hai')
>>> L
[11, 22, 33, 'h', 'a', 'i']
>>> L = [11,22,33]
>>> L.append(90)
>>> L
[11, 22, 33, 90]
>>> L.append(67)
>>> L
[11, 22, 33, 90, 67]
>>> L.append('hai')
>>> L
[11, 22, 33, 90, 67, 'hai']
>>> L.insert(1,100)
>>> L
[11, 100, 22, 33]
>>> L.insert(1,'hai')
>>> L
[11, 'hai', 100, 22, 33]
>>> L.insert(19999999, [77,66])
>>> L
[11, 'hai', 100, 22, 33, [77, 66]]
```

Deletion methods of List Data type:

Pop()	Remove()
Syntax: Pop([index_position]) Default value for index_position = -1	Syntax: Remove(value)

Based on given index position, pop will delete the element	Based on given value, remove will delete the element Even if the specified element is present for multiple times, it deletes the element only once
If the specified index is more than the length, it will throw an error	If specified value is not present, it will throw an error

```
>>> L
[11, 'hai', 100, 22, 33, [77, 66]]
>>> L.pop(1)
'hai'
>>> L.pop(2)
22
>>> L
[11, 100, 33, [77, 66]]
>>> L.pop()
[77, 66]
>>> L
[11, 100, 33]
>>> L.remove(11)
>>> L
[100, 33]
>>> L.remove(11)
Traceback (most recent call last):
  File "<pyshell#33>", line 1, in <module>
    L.remove(11)
ValueError: list.remove(x): x not in list
>>> L.pop
<built-in method pop of list object at 0x0000026
>>> L.pop(99)
Traceback (most recent call last):
  File "<pyshell#35>", line 1, in <module>
    L.pop(99)
IndexError: pop index out of range
>>> L = [90,78,89,90]
>>> L.remove(90)
>>> L
[78, 89, 90]
```

Built-in Methods of Dictionary:

Keys()	Values()	Items()
Syntax: Keys()	Syntax: Values()	Synatx: Items()
It is used for getting the keys of specified dictionary	It is used for getting the values of specified dictionary	It is used for getting both thee keys and value s of specified dictionary.

Built-in methods of dictionary:

1. Copy() ----------> it is used for performing shallow copy
2. Clear() ----------> It will delete only the elements from the given dictionary

Get()	Setdefault()
Syntax: Get(key-name, [default-value])	Syntax: Setdefault(key-name, [default-value])
If specified key is present, then get will display the value of specified key, else it will return the default Value and it will not update the dictionary	If specified key is present, then setdefault will display the value of specified keys, else it will return the default value and update the dictionary with specified key and default value.

Update method:

It is used for updating the actual dictionary with multiple key and value pairs at a time.

Note:

It is used for merging two dictionaries.
We can provide pairs of data as value for update method.

Syntax:

Base-dictionary.update({'key1':'value1', 'key2':'value2', ...})

If specified key is present, it will update the value.
If key is not present, then it will create a new key value pair.

Deletion methods:

Pop()	Popitem()
Syntax: Pop(key-name)	Syntax: Popitem()
1. It is used for deleting key value pairs based on specified key 2. If key is present, it will delete the key, else it will throw key error	1. It is used for deleting last key and value pair by defalut 2. Output format of popitem is a tuple 3. If the dictionary is empty, it throws a key error

fromkeys:

Syntax:

fromkeys(CDT, [default-value])

t

It is used for extracting each and every element of CDT and represent them as keys and default value as value for all the keys.