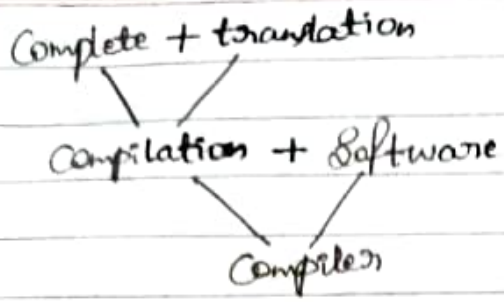
21/09/20

PYTHON

chandra's



* Compilation :- The process of Receiving all code and convert it into a Binary instruction.
Software that will do a compilation is called Compiler.

Interpretation = Interprete + translation
Interpreter = Interpretation + S/w

* Interpretation :- The process of converting code into line by line instruction is called interpretation.
and the S/w i.e. used for interpretation is called Interpreter.

* Python is an Interpreted and Scripting language.
* Scripting languages are used the Interpretetion.
* Scripting languages are more efficient than programming languages because they are converting line by line instructions. So it decreases the no. of errors.

* The process of converting code into computer language is called translation.

* python is an open source language (No company can owns it & anybody can get the source code)

* open source :- It is a type of S/w where the source code is exposed to the normal users.

* As it is exposed to the external environment many people's contribution will be there in it. that makes the language to grow faster.

* Usually open source are not the proprietory one.
Ex :- C programming, Linux and UNIX OS, python, Java.

* Closed Source :- It is the type of Software where the code is exposed only to the team of developers who had worked on the development of the language.

* If the source code is not exposed. the ~~possible~~ people's contribution in the development of a language will not be there that avoids / reflects on the growth of the language.

* closed source SWs are proprietary. (owned by company / an organization).

Ex :- Window OS, C# language, VLC media player, NFS most wanted.

* Python is a high level language.

[Because it is well advanced to get adopted to the current environment and it is one of the recently developed language]

* Static Typing :- It is a phenomenon of keeping the type & size of the memory location are fixed.

The languages that follows Static typing are called Static typed programming languages. The languages are C, C++, Java etc.

In C, C++, Java. if we say int a; then till the end of program a must store only integers.

The size of particular memory is only of 4B.

* Dynamic Typing :- It is the phenomenon of making the type and the size of the ML of a not fixed variable.

The language that follows Dynamic typing are called as Dynamic typed programming languages.

The language that follows Dynamic typing is Python, PHP, perl etc.

* Python is called a Dynamic typed programming language.

* Python is completely object oriented programming language.

* Python is a high level object oriented & dynamic typed programming language.

2) proff the Start button type env in Search and edit S/m environment click

3) click on environment variable → click on path → click new → paste both the path individually click on ok.
Again in the S/m variable Section double click on path → click on new & paste both the paths individually and click on OK options till you find the last OK button.

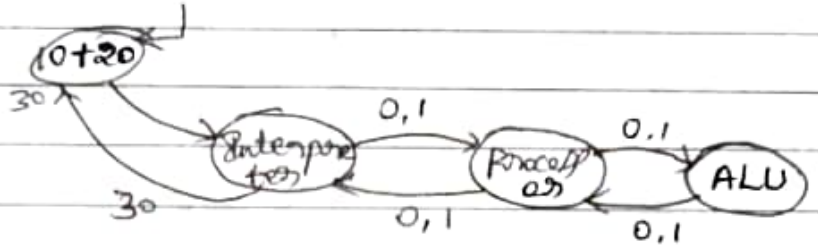
4) To check wheather the path given is Successfull or not goto command chrome and type python and proff enter.

5) if you find python Shell like Structure then the installation happened Successfully.

IDE (Integrated development environment)

* It is a place where we write and execute our code.
* Every programming languages will have it's own IDE with the Same time even there will be Some generic IDE's.
  Ex:- Eclipse, netbeans, Microsoft Studio, etc.
* for the first time in python the developers hap come up with a new concept of introducing the dedicated IDE for python and they called it as IDLE Stands for integrated development & learning environment. bcz of the reason of having a help desk within itself that helps to know the use of functions, Syntaxes & arguments used in it.

* To get the idle prompt the Start button & type IDELE So that gives you to the Shell where you write your code.
* In Shell just type a line of instruction & press enter the enter specified the application to give the instruction to interpreter.



* The memory in our Computers is like the blocks or the Computer memory is designed to be in the form of blocks.
  Each block size will be fixed.
  One block = The bit size of the Slm
  Usually we say 32 bit Slm, 64-bit Slm etc
* If our PC / Computer is 32-bit PC then one block is equals to 32-bits which is further equals to 4-Bites
* Variables :- A variable is a named memory location where the value is Stored.
* We name the memory location so that accessing the value present in a easier. The Syntax to Store variable

  Syntax 1 Variable name = Value
* If we wanted to Store multiple variables at one

Var 1, Var 2, Var 3 - Var n = Var1, Var2, Var3 — Varn

Var = Value

C = 30

e.g. = 1, 2

To Install

steps-2 click on downloaded file → Select path

→ install wt system / custom install wt to user requirements.

16/10/2020 Python

Library functions

Library functions are the inbuilt functions which has been created or designed to perform the assigned task or specific task

→ Keywords (Type of lib function)

Keywords are the words which has been created to perform the assigned specific task. we have 36 keywords (from version Python 3.9)

the keywords are only for the usage purpose we cannot modify the meaning of any keyword (only for usage)

```
import keyword
print (keyword.kwlist)
```

```
['False', 'None', 'True', '__peg_parser__', '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']
```

3.6 python had 33 keywords

3.8 " has 35 keywords

3.9 " has 36 keywords

Variables

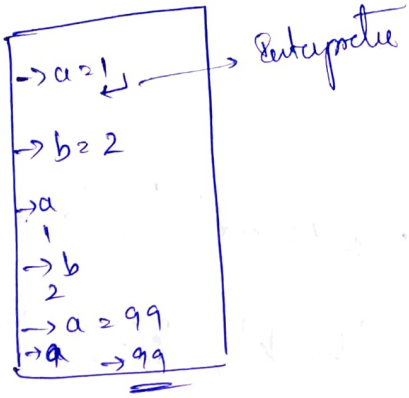
Variable is a process to assign a reference name for the data which is being stored in memory. (Or) Giving a name for a value is called Variable Syntax

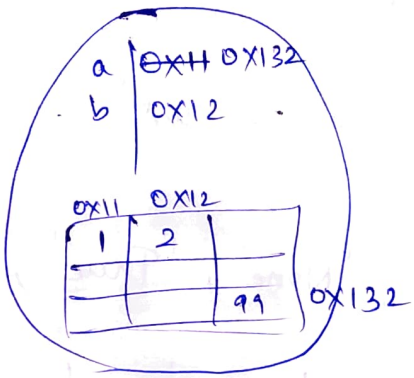
```
Variable_name = value
```

Eg a=1, b=2 ; a=99 etc

id(a) → gives address of variable a

hex(id(a)) → gives hex value of address of variables





Identifiers

Are the names that are given to identify the memory locations

All the Variables are Identifiers

Rules to Define

(1) Identifiers Should Not be keywords

(2) No Special Symbols are allowed (except underscore)

(3) Reason → Every special that are present on keyboard are associated with a Task If we use that to name then that leads to lot of Confusion & ambiguity

(3) The number of Characters in the Identifier Should not exceed 32 Characters. (Reason → human brain will not have the capability to remember such lengthier words)

(4) No Space Reason

Spaces .

(5) Identifiers

(6) Identifiers

Identify

1) A2A = 12

2) a;b = 12

3) a b

4) _a = 12

5) _a_ = 13

6)

Note pyth

[upper C

Data Types

the

Language

need fo

d

④ No space is allowed defining a Identifier.

Reason → ambiguity / confusion in placing the spaces. If we want to access the variables.

⑤ Identifier must start with alphabet (or) an underscore

⑥ Identifier should not be repeated

Identify whether the variables are valid (or) not

1) A2A = 12 ; Valid
2) a;b = 12 ; Invalid
3) a b = 12,13 ; invalid
4) _a = 12 ; valid
5) _a_ = 13 ; valid
6) [redacted] = 4 ; valid

[redacted] = 5 ; valid

_ _ = 5 ; invalid

true = 0 ; valid

True = 1 ; invalid

Note Python is a case sensitive language

[Upper Case & Lower Case are different]

Data Types

These are going specify the type of the values that are present, in the programming language. These will also define how the values need to be stored in the memory.

Based on the no of data items the data types are segregated into 2 types

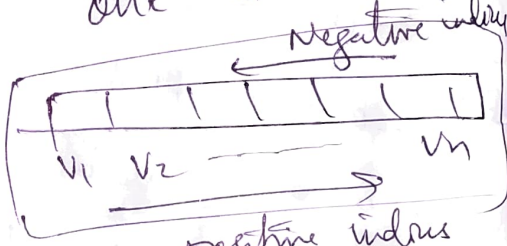
① Single value data types

② Collection data type (Multivalue data type)

Data type

- Single Value
  - Numeric
    - Int (Integer)
    - Complex (a+bj form)
    - float
  - Boolean
- Multi value

It is a category that deals with more than one data item



N → total no of data items

#Note

① Block of memory can able to store only 1 data item [value (or) address]

② To know the type value that we have or that is stored inside the variable we use type() function.

Syntax : type (value) or type (variable_name)

```
>>> a = 10
>>> a
10
```

type(a)
<class 'int'>

b = 3
type(b)
<class 'int'>

c = 1.2
type(c)
<class 'float'>

type(-1.4)
<class 'float'>

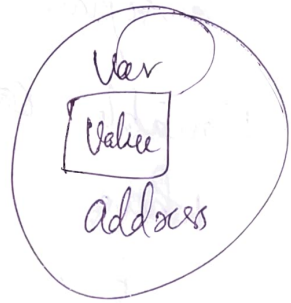
type(2+3j)
<class 'complex'>

c = 3+0j
type(c)
<class 'complex'>

d = 3j
type(d)
<class 'complex'>

Boolean

If the variable has
True (or) False
value then it is a Bool type



True & false are keywords

a = True
type(a)
<class 'bool'>

type(True)
<class 'bool'>

b = False
type(b)
<class 'bool'>

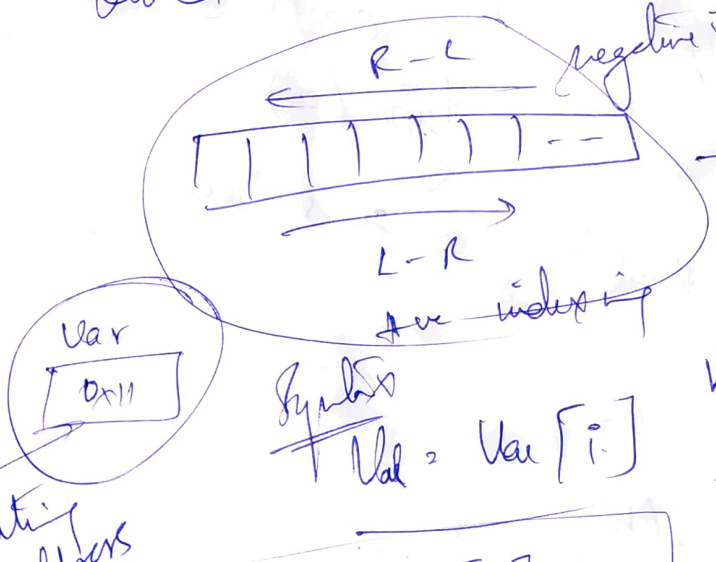
Collections

→ It is a type or Category that deals with more than one data item

→ as there are more than one data item i cannot able to store it inside a single block because we know that one block can able to store only one data item. in that case we store the values separately & the variable separately.

→ Variable contains the address of the memory where the values are stored So it is called as Variable space

The space where the values are stored is called as Value space



Syntax
Val = Var [i]

n → Total no of items

Val = Var [0] (n)
= Var [(n-1)]

As it is very difficult to identify each & every value present in the value space using the starting address of the memory we came up with a concept of assigning the sub-addresses to each and every block of memory and we called it as indexing

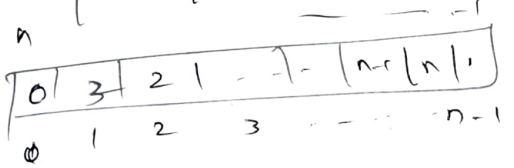
In python we have a concept of +ve indexing as well as -ve indexing that helps us to do the bi-directional traversing.

Whenever we consider from left to Right then we consider the +ve indexing in the next

whenever we consider from R → L we consider -ve addressing

+ve addressing ranges from 0 - (n-1) (and)

-ve addressing ranges from -1 → -n



n is nothing but the total no of data items that are present in the collection.

To find the total no of data items (or) length of the collection. we are going to use a built in function called len()

The Syntax is

Var = len (Collection Variable or Collection Value)

→ To identify Each and Every block of memory in the Value Space we use the Syntax

Vi ⇒ Var[index] This syntax will help us to fetch the values present at the Specified index

→ The index Can be both +ve as well as -ve

Example

a = 'hello'	b = [1, 2, 3, 4]
len(a) = 5	len(b) = 4
a[1] → e	b[2] = 3
a[-1] → o	b[-2] = 3
a[5] → ⊘	

To Modify the value present at the Specified index in value space we use a Syntax

Var[index] = newValue

If Ever we wanted to Modify the Variable space we use a Syntax Variable is

Var = newValue

There are 5 diff
present in pyt

① String

→ String

string is a
enclosed between

1. one character
2 in python

③ string

• only

④ for string

⑤ Var = 'Var'

⑥ we use

⑦ a str

Eg a = Good

There are 5 different types of collections (built in) present in python

① String ② List ③ Tuple ④ set

⑤ dictionary.

→ String

string is a collection of characters that are enclosed between the pair of '' or "" or """ """

1. One character is Equivalent to 1 key in keyboard
2. in python single quotes and double quotes doesn't make any difference
3. string is an ** immutable ** (cannot be modified)
   - only value space cannot be modified
4. for string type we use str as the name of data types
5. Var = 'Val Val Val ---- Valn' or Var = "Val ---- Valn"
   or """Val Val ---- ---- Valn"""
6. we use """ """ only when we have the string containing 1 line of character
7. a string that has multiple lines is called doc string.

Eg as Good morning

-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
G	O	O	D	M	O	R	N	I	N	G
0	1	2	3	4	5	6	7	8	9	10	11

a = "Good Morning" ⇒ o/p 'Good Morning'

b = 'Good Morning' ⇒ 'Good Morning'

c = "Raja 's" ⇒ "Raja 's"

d = """ Hello, how
are you
buddy """

o/p ⇒ 'Hello, how \n are you \n buddy'

a[3] ⇒ 'd' a[6] = 'o'

a[-1] ⇒ 'd' a[-6] = 'o'

a[12] ⇒ Error ⇒ string index out of range

a[3] = 'a' ⇒ Error ⇒ does not support assignment

Str is a name of data type of object / of type string.

List

A list is a collection of homogenous and heterogenous data items, where the data items are separated by comma, operator and the collection is enclosed b/w the pair of [ ]

① Syntax: var = [val1, val2, val3 ... valn]

② list is mutable type (modifiable)

③ list is name of the data type used in representation

④ list allows data of any type in it.

① a = [1, 2, 3, 4, 5]

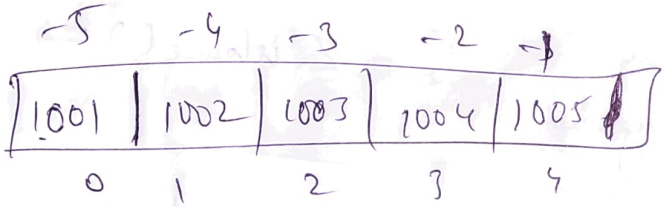
a[1]

⇒ 2

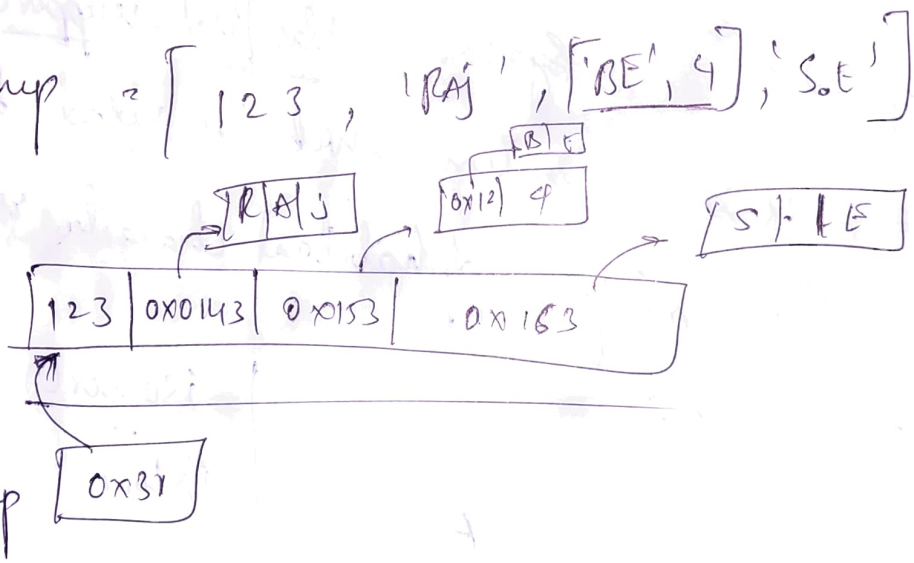
② a[5]

⇒ Error

Regno = [1001, 1002, 1003, 1004, 1005]



Emp = [123, 'RAJ', ['BE', 4], 'Sat']



Inbuilt functions on strings

① islower()

Syntax: Var/Val.islower()

→ It is a function which is used to check the strings alphabetical character is in lower case or not.

Example Z = 'a'
Z.islower() ⇒ True

Z = 'I'
Z.islower() ⇒ False

Z = '123 hai'
Z.islower() ⇒ True
will be neglected

Z = '123Hai'
Z.islower() ⇒ False

② isupper()

Syntax :- Var/Val.isupper()

→ It is a function which is used to check the strings alphabetical character is in uppercase or not

Example Z = 'A'
Z.isupper()
⇒ True

isupper() } neglect nos
islower() } Consider only characters

Z = 'A123BC'
Z.isupper() ⇒ True

③ isnumeric()

Syntax : Var/Val.isnumeric()

Z = '1234'
Z.isnumeric() ⇒ True

Z = '1234.999'
Z.isnumeric() ⇒ False

Strictly if only nos are there it will be True else False

④ isalpha()

Var/Val.isalpha()

Z = 'abc123'
Z.isalpha() ⇒ false

Z = 'abcdABF'
Z.isalpha() ⇒ True

Strictly only alphabets/characters in alpha are allowed

⑤ isalnum()

Var/Val.isalnum

[no special symbols & spaces not allowed] [Consider only alphabets & numbers valid]

Z = 'Zoya'
Z.isalnum() ⇒ True

Z = 'Zoya143'
Z.isalnum() ⇒ True

Z = 'Zoya143' ⇒ Z.isalnum() ⇒ False

Z = 'Zoya - waqs - faisal'

Z.isalnum() ⇒ False

(6) upper()

Var/val.upper()

Z = 'hai123'

Z.upper() ⇒ 'HAI123'

(7) lower()

Z = 'HaiHow123'

Z.lower()

⇒ 'haihow123'

(8) Capitalise()

Z = 'roopan'

Z.capitalize()

⇒ Roopan

Z = 'ROPA'

Z.capitalize()

⇒ 'Roopa'

Z = 'hai roopa how are you'

Z.capitalize()

'Hai roopa how are you

It is a function which is used to change the 1st character (alphabet) to upper & rest of the upper case to lowercase

Z = '123hai'

Z.capitalize()

⇒ '123hai'

Z = '123Ahai'

Z.capitalize()

⇒ '123ahai'

(9) title Var/val.title()

Z = 'Hai Roopa How you'

Z.title()

⇒ 'Hai Roopa How You'

Eg② Z = '123hai how'

Z.title()

⇒ '123Hai How'

It is a function which is used to convert the first alphabetical character of each word to uppercase and remaining alphabetical character to small.

(10) istitle()

Z = 'How Are You'

Z.istitle() ⇒ True

Z = 'How ARe YOU' ⇒ Z.istitle() ⇒ False

(11)

count

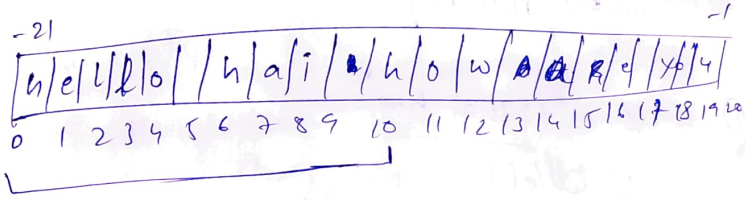
① Syntax: Var.count('substr')

It is a function which is used to count the number of specified substring occurence of in a given string.

② Syntax: Var.count(substr, startval, endval+1)

start & end value in range of
↳ is index nos

Eg z = 'hello hai how are you'



z.count('h', 0, 9)
z.count('h', 0, 10)
z.count('h', 10, 21)

we can also give 2 parameter
z.count('h', 10)
⇒ 1 : it start counting from 10

we can pass the limits There 3 are syntax of count

count → Var.count(substr)
Var.count(substr, startval)
Var.count(substr, startval, endval+1)

(12) index

z.index(substr)
z.index(substr, startval, endval+1)

It is will index of 1st occurial substring

z.index('a') → 7
z.index('h') → 0
z.index('hx') → 0
z.index('ho') → 10

If the substr not found it will through/throw an error → substring not found

(13) find()

It is similar index but when the substr is not found it will return '-1' instead of error

Everything else is same.

Syntax: Var.find(substr)
Var.find(substr, startval)
Var.find(substr, startval, endval+1)

(14) startswith

Var.startswith(') → substr

Z.startswith('h')
⇒ True

Z.startswith('a')
⇒ False

Z.startswith('hello')
⇒ True

Z.startswith('ha')
⇒ False

Var.startswith(substr, startval)

Var.startswith(substr, startval, endval+1)

Z.startswith('a', 7) ⇒ true

Z.startswith('ai', 7) ⇒ true

Z.startswith('how', 9, 20)
⇒ False

Z.startswith('how', 10, 20)
⇒ True

Z.startswith('hello', 0, 2)
⇒ False

---

(15) endswith()

Var.endswith(') → substr

Var.endswith(substr, startval, endval+1)

Var.endswith(substr, startval)

Z.endswith('a')
⇒ True

Tuple

is a type of collection which is used to store the data of homogenous type & heterogeneous type the data will be enclosed upon the pair of parenthesis and the data items are separated by using comma operator

→ Tuple is an immutable type where the value space cannot be modified.

→ Tuple is completely similar to list, and the only difference is its immutable so it is faster

→ Syntax

Var = (Val1, Val2, ... Valn)

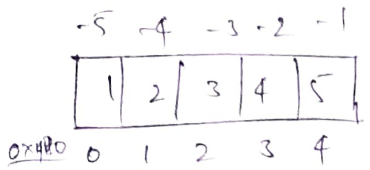
(a) Var = Val1, Val2, Val3, Val4 ... n

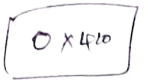
To store a single Element

Var = (Val1,)

Var = Val,

a = (1, 2, 3, 4, 5)

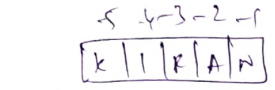


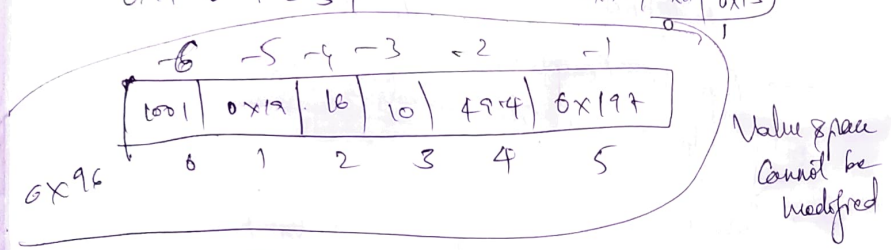


a[3] => 3 <= a[-3]

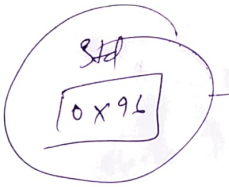
a[1] => 2 <= a[-4]

Std = [1001, 'Kiran', 16, 10, 49.4, 'PCMC', 'k']





Value space Cannot be modified



→ Variable space can be modified

Std[5][0] = 'c' ✓

Std[1] => 'Kiran' <= Std[-5][-3]

Std[5][0] => 'PCMC'

Std[5][0][2] => 'M'

Cont.. of String inbuilt functions

x = 'happy dasara where is the sweet'

(16) Split()

Syntax

```
x.split
x.split(substr)
x.split(substr, no_of_splits)
```

function to split the given string whenever is specified substring is found. for specified no of times

Examples

① x.split()

→ ['happy', 'dasara', 'where', 'is', 'the', 'sweet']

② zz = x.split('s')

= ['happy dasara ', 'where is the sweet']

(17) join()

~~join()~~ → guests.join(listofstrings)

Eg ' '.join(zz)

'-'.join(zz)

It is a function which is used to join the splitted strings present in a list

(18) Replace()

It is a function which is used to replace the specified substring present in a given string for the specified no of times.

Replace

Var. replace (oldstr, newstr)

Var. replace (oldstr, newstr, no. of replacements)

List inbuilt functions

(1) append()

Syntax: Var. append (Val/data)

It is a function which is used to add an element at the end of the list collection

Element → Can be any data

without inbuilt

a = [1,2,3,4]

b = 88

a + [b]

[1,2,3,4] + [88]

⇒ [1,2,3,4,88]

(2) insert()

Var. insert (position, data)

Eg a = [1,2,3,4]

a. insert (1,5)

→ a = [1,5,2,3,4]

used to insert an element in the user defined position once after the insertion is done the other index value will be right shifted one position

a = [1,2,3,4]

a. insert (1000, 99)

inserted in last position if index not provided

a ⇒ [1,2,3,4,99]

(3) Remove

(1) → pop()

↳ It is a function which is used to remove the last element from the list and display

↳ Var. pop()

(2) Remove()

Var. remove (element)

Removes specified element from the collection.

(5) clear()

existing

It will clear all the elements in list and gives a empty list

a.clear()

⟹ a ⇒ []

(6) count()

It is a function which is used to count the number of occurence of an element in a list

⟶ Var.count(Element)

Eg a = [1, 3, 5, 1, 6, 1, 7, 9, 1]

a.count(1) ⇒ 4

(7) index()

Var.index(Value)

Var.index(Value, start index value)

Var.index(Value, start index value, end index value+1)

(8) reverse()

Var.reverse()

The existing var will be modified to reversed list.

a.reverse()

(or) a[::-1] without output

(9) sort()

a.sort()

⟹ a ⇒ [1, 2, 3, 4, 6, 7, 7, 89]

a = [34, 'hai', 45.67, [44, 55], 3+5j, False, (4, 2j)]

a.sort()

↳ error → str & int must separated with '<'.

a = ['hai', 'bye', 'how', 'hello', 'poja', 'roopa']

a.sort()

a ⇒ ↳ ['bye', 'hai', 'how', 'hello', 'poja', 'roopa']

To do in descending

Var.sort(reverse = True)

(5) clear()

existing

It will clear all the elements in list and give a empty list

a.clear()

⟹ a ⇒ []

(6) count()

It is a function which is used to count the number of occurence of an element in a list

⟶ Var.count(Element)

Eg a = [1, 3, 5, 1, 6, 1, 7, 9, 1]

a.count(1) ⟹ 4

(7) index()

Var.index(Value)

Var.index(Value, start index value)

Var.index(Value, startindex, endindex)
Value Value+1

(8) reverse()

Var.reverse()

The existing var will be modified to reversed list.

a.reverse()

(or) a[::-1] without inbuilt

(9) sort()

a.sort()

⟹ a ⇒ [1, 2, 3, 4, 6, 7, 7, 8]

a = [84, 'hai', 45.67, [44, 55], 345, False, (45, 23)]

a.sort()

↳ error → str & int not supported with '<'.

a = ['hai', 'bye', 'how', 'hello', 'raja', 'deepa']

a.sort()

a ⟹ ↳ ['bye', 'hai', 'how', 'hello', 'raja', 'deepa']

To do in descending

Var.sort(reverse = True)