Variable data types

**Comments**: there are two types

1. **Single line**: to denote a single line we use the hashtag(#).
2. **Multiline**: to denote a multi-line comment we use (‘’’).

**Syntax**:- variable name = value

ex:- a=1,4,5

print(a)

output = (1,4,5)

**variable cases** :

1. camelCase:

ex:- ram Nagar

1. snake\_case:

ex:- ram nagar

1. PascalCase:

ex:- Ram Nagar

**Data Types**

**NUMERIC TYPES**

1. **Float**: the float is numerical and contains decimal in it. Ex:- 3.4,2.5 etc.
2. **Integer**: the integer contains all the whole numbers. Ex:- 1,2,4 etc.
3. **Boolean**: the Boolean contains statements like true or false.
4. **Complex**: the complex contains all the complex numbers. Ex:- 1+2i , j-3 etc.

**Type function**

**Syntax:-** ex**:-** a=3

b=5.5

Print(type(a),type(b))

Output=<class “int”> <class “float”>

**Type conversion**

* Int()
* Float()
* Bool()
* Complex()

Ex:

1. float🡪int

d= 2.22

a = int(d)

print(a)

output: 2

1. int🡪float

d=22

a= float(d)

print(a)

output: 22.0

1. float🡪complex

d=22.4

a= complex(d)

print(a)

output: 22.4+0j

**Athematic operators**

* addition: “+”
* subtraction: “-“
* multiplication: “\*”
* division: “/”
* modules: “%”
* exponents: “\*\*”
* floor division:”//”

**syntax:-**

x+y = result

here x+y are the operands.

**Logical operators**

|  |  |  |
| --- | --- | --- |
| **x** | **y** | **x & y** |
| false | false | false |
| false | true | false |
| true | false | false |
| true | true | true |

**Assignment operators**

* assignment: =
* add & assignment: +=
* subtract & assignment: -=
* multiplication & assignment:\*=
* division & assignment:/=
* modules & assignment:%+

**Relational operators**

* less than: <
* greater than: >
* less than or equal: <=
* greater or equal:>=
* equal to: ==
* not equal to:!=

**Loops**

1. **if:**

Syntax:- if (condition):

ex: if a>b :

print(“this is if condition” )

2.**else**:

Syntax:- else (condition):

print(“this is else”)

3.**elif:**

Syntax:- elif (condition):

print(“this is elif”)

4.**if within if**:

Syntax:- if (condition) :

print(“this is if condition”)

if (condition) :

print(“this is inner if”)

else:

print(“this is inner else”)

else (condition):

print(“this is else”)

5.**short hand if**:

Syntax:- if (condition):

Print(“if”)

6.**short hand if and else**:

Syntax:- print(“if) if (condition) else print (“else”).

7.**while loop**:

Syntax:- while(condition):

print(“this is my while loop”)

If (condition):

print(“this is if “)

Else(condition):

print(“this is else”)

8. **for loop :**

Syntax:-for i in “Yamini”:

Here:

i = the temp variable.

in = membership operator

Yamini= sequence (string, list, dictionary, set, range)

9**. nested for loop:**

Syntax**:-** for i in range():

for j in range():

print(the required condition)

10. **nested while loop**:

Syntax:- while (condition):

While (condition):

print(the required condition)

**Functions**

1. **break function :**

Syntax:- while (condition):

if (condition) :

break

print(“the condition”)

print(“the loop is done”)

2.**continue function** :

Syntax:- for j in “Yamini”:

If (condition):

Continue

print(“the variable”)

**SEQUENCE TYPES**

**1.List**

The variable values are taken in []

**Type function:**

Syntax**:-** ex:- dharani=[1,6,8.3,2,”harshini”]

print(type(dharani))

output=<class “list”>

**index:**

Structure of a list:

Names=[“lasya”, ”Priya”,”Sonu”,”ram”]

Index:- lasya=[0]

Priya=[1]

Sonu=[2]

Ram=[3]

Syntax:-

bhanu=[0,4,3.9,”Dhanush”,true,4+3i]

print(bhanu[3])

Fatima=[0,3,10,”sree”,(0,4,6)]

Print(Fatima[4],[2])

**Slicing**

Syntax:-

L[start:stop:step]

Start position, Stop position, Step position.

Ex: thanuja=[1,5,7,9,10,6,4,3,8]

print( thanuja[0:8:3])

**Specifying**

Syntax:-

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f | g | h | i | j |

L.H.S to R.H.S is positive.

R.H.S to L.H.S is negative.

Ex: rakesh=[3,2.4,5,6,’Kumar,7]

Print(rakesh[5:0:-1])

Output= [7,‘Kumar,6,5,2.4]

* if two colons(::)are used the output obtained would be the total list.

Ex: print(rakesh[::])

Output:- [3,2.4,5,6,’Kumar,7]

* if one colon (:)is used the output obtained would be the total list.

Ex: print(rakesh[:])

Output:- [3,2.4,5,6,’Kumar,7]

* if we use [:-2], the output obtained would be that the last element is skipped.
* If we use [::-2], the output obtained would be that the first element is skipped.

**List methods**

1. **append:** In this method, the append is added at the end of the list.

**Syntax:** variable.append(string)

1. **extend:** In this method, the extend is added at the end of the list.

**Syntax:** variable.extend(string)

1. **copy:** In this method, the total list is copied as it is in the original list.

**Syntax:** variable.copy()

1. **clear:** In this method, the whole list is shown as cleared.

**Syntax**: variable.clear()

1. **index:** In this method, when the index of the particular element is given the output obtained is an element.

**Syntax:** print(variable.index())

1. **insert:** In this method, we can replace the element with another by defining the index of the old one.

**Syntax:**  variable.insert(index,string)

1. **pop:** In this method, the element can be removed by defining its index.

**Syntax:** variable.pop(index)

1. **remove**: In this method, the element can be removed by defining it.

**Syntax:**  variable.remove(element)

1. **reverse**: In this method, we get the elements in the list in reverse order.

**Syntax:** print(variable.reverse())

1. **sort:** In this method, the list is either arranged in ascending or descending order.

**Syntax**: variable. sort(reverse = true or false)

**List comprehension**

**Syntax:** list [expression if condition for item in sequence ]

1. **String**

The variable value is taken in “”.

**Type function:**

**Syntax**: gurkul=’tarun’

Ramana**=’**rama**’**

Print(type(gurkul),type(ramana))

Output= <class ‘string’> <class ‘string’>

**String methods**

1. **upper:** all words in the string are changed into capital letters.

**Syntax:** x= variable.upper()

print(x)

1. **Lower**: the all-capital letters in the string are changed into small letters.

Syntax: x= variable.lower()

print(x)

1. **count**: this method is used to find how many times a word is repeated.

Syntax: print(variable.count(‘string’))

1. **starts with**: from this we can find whether the sentence is starting with a particular word or a letter is true or false.

Syntax: print(variable.startswith(‘string’))

1. **ends with:** from this, we can find whether the sentence is ending with that particular word or a letter is true or false.

Syntax: print(variable.endswith(‘string’))

1. **find**: from this, we can find the index number of a string in the variable and if we provide a string that is not present in the variable then we get -1 as the index.

Syntax: print(variable.find(‘string’)

e) **index**: we can find the index of the word.

Syntax: print(variable.index(‘string’)

1. **is al num:** we check whether the string contains both alphabets or numbers.

Syntax: print(variable.isalnum()

1. **digit**: we check whether the string contains digits.

Syntax: print(variable.isdigit()

1. **strip**: In this, we adjust the extra spaces in the sentences.

Syntax: print(len(variable))

x= variable.strip()

print(len(x))

print(x)

1. **lstrip**: In this method, the left side spaces of the sentence are adjusted.

Syntax: x= variable.lstrip()

print(len(x))

print(x)

1. **rstrip:** In this method, the right side spaces of the sentence are adjusted.

Syntax: x= variable.rstrip()

print(len(x))

print(x)

1. **split:** the sentence is divided into strings

Syntax: x= variable.split()

print(x)

1. **title:** the first letters in the words are to be capital in the sentence.

Syntax: x= variable.title()

Print(x)

1. **replace:** the words can be replaced in a sentence with a new one.

Syntax: x= variable.replace(old string, new string)

1. **remove suffix**: we can remove the last word in the sentence.

Syntax: x= variable.removesuffix(‘string’)

1. **remove prefix:** we can remove the starting word in the sentence.

Syntax: x= variable.removeprefix(‘string’)

1. **format**: we can add any name in the sentence for a given message.

Syntax: d= “hi {r} how are u doing”.format(r=’string’)

Print(d)

1. **join**: we can join the sentence which is separated into strings.’

Syntax: print(original variable)

x= variable.split()

print(x)

a= variable.join(x)

print(a)

**3.TUPLE**

This variable value is denoted by ().

**Type function:**

Syntax**:** dinesh = ()

print(type(dinesh))

**Built-in functions**

* **Min:** the minimum number in a tuple.

Syntax: print(min(variable))

* **Max:** the maximum number in a tuple.

Syntax: print(max(variable))

* **Sum**: the sum of all the numbers in both sets.

Syntax: print(sum(variable))

* **Length:** the length of the set is defined.

Syntax: print(len(variable))

**Tuple methods**

1. **Repetition:**

The tuple can be repeated many times.

Syntax: print(variable\*no. of times)

1. **Concatenation**:

The values in one tuple can be added with the corresponding numbers in the other tuple.

Syntax: print(tuple1 + tuple2)

1. **Membership**:

Here there are two conditions whether the value is in the tuple or not in the tuple.

Syntax: print(int not in variable)

print(int in variable)

1. **Iteration:**

Here the numbers are printed line by line.

Syntax: for i in variable:

print(i)

**python dictionary**

the variable value is denoted by {}

**Type function:**

Syntax: madhu = {}

Print(type(madhu))

**Structure**:

Ganesh={‘dharani’:’girl’}

Here dharani = keyword

Girl= value

**Python functions**

1. **Clear**: In the dictionary, all the elements are cleared.

Syntax: variable.clear()

1. **Copy:** the whole dictionary is printed as it is easy.

Syntax: variable.copy()

1. **Get:** here we use the key to call any value in the dictionary.

Syntax: print(variable.get(“key”)

1. **Keys:** by using this we can only print the keys.

Syntax: print(variable.keys())

1. **Values:** by using this we can only print the values in the dictionary.

Syntax: print(variable.values())

1. **Items:** by using this function we can print both the keys and the values.

Syntax: print(variable.items())

1. **Update**: we can add a key and the value in the dictionary by using update.

Syntax: variable.upadte({‘key’ : ‘value’})

1. **Pop**: we can remove or skip any value by using the keyword.

Syntax: variable.pop(key)

**For loop**

**Syntax:** for i in variable:

Print(variable)

# here only keys are printed**.**

for i,j in variable.items():

Print(i,j)

# here both are key and value is printed.

**Set types**

1. **SET**

The variable value is defined in {}.

**Type function:**

S= {4,6,8,10,3,2}

print(type (s))

**set methods**

* **Add:** to the set we add some string.

Syntax:variable.add(‘string’)

* **Clear:** the set is cleared by using the clear method.

Syntax: variable.clear()

* **Copy:** the same original set is obtained in the output also.

Syntax: variable.copy()

* **Pop:** here we can remove only the first element in the set.

Syntax: variable.pop()

* **Remove:** In this method, we can remove any element in the set.

Syntax: variable.remove

* **Update:** we can add a set of numbers in the given set by using this function.

Syntax: variable.update({})

**Set operations**

* **Union:** the combination of two different sets if at all it has common numbers then duplicate is eliminated and the original number is taken.

Syntax: print(set1.uinon(set2))

* **Intersection:** the common numbers in the two sets are taken.

Syntax: print(set1.intersection(set2))

* **Difference:** we subtract the numbers from the two sets concerning the first set elements only.

Syntax: print(set1.difference(set2))

* **Symmetric difference:** leaving the common numbers in the two sets we only take the remaining numbers.

**Syntax:** print(set1.symmentric\_diference(set2))

* **Is subset:** if the set2 contains all elements in the set1.

Syntax: print(set1.issubet(set2))

* **Is superset:** if the set2 contains another set in it i.e set1.

Syntax: print(set1.issuperset(set2))

* **Is disjiont:** if the two sets are comnbined together.

Syntax: print(set1.disjiont(set2))

**2.FROZEN SET**

The variable value is denoted by []

**Syntax:** eswar = [1,3,4,5]

x= frozen set(eswar)

print(x)

* Note: frozen set can’t be changed using any functions.