AMIT ACADEMY
for Computer Education

SUBJECT – PROGRAMMING WITH PYTHON

Faculty Name – Ankit Rami

Unit-3

Built In Functions of Python

AMIT ACADEMY FOR COMPUTER EDUCATION

Nr Vardayini Mataji Temple, Rupal, Gandhinagar-382630

Email – amitacademy1117@gmail.com

Mobile No – 8460467193

YouTube Link –

https://www.youtube.com/@ankitramijoinar

Instagram Link –

https://www.instagram.com/amitacademy17/

Facebook Link –

https://www.facebook.com/aramitacademy/

Built in general functions

- ✓ Python provides a lot of built-in functions that eases the writing of code. In this article, you will find the list of all built-in functions of Python.
- ✓ **Built-in Conversion Functions**
- ✓ Below is a selection of Python built-in functions all focused on converting one data type into another Description for each and the syntax when utilizing them in Python code are also given.

Built-in Conversion Functions	Description of Function	Syntax
ascii()	ASCII is the character encoding standard for electronic communication. This function will return readable version of the data. If any point of data within a variable is non-ascii character it will be replaced with escape characters. Ergo 'ö' will turn into '\xf6'	| ascii(identifier) |
bin()	Binary. Used to convert an integer number to a binary string prefixed with | 0b |.	| bin(identifier) |
hex()	Converts an integer number to a lowercase hexadecimal string. This string will be prefixed with | 0x |.	| hex(identifier) |
oct()	Converts an integer number to an octal string which is prefixed with | 0o |.	| oct(identifier) |
chr()	Returns a character from the specified Unicode code. For example: | chr(97) | will return 'a'	| chr(identifier) |
ord()	Converts a string representing a Unicode code returning the specified character delineated by the Unicode value. This is the inverse of chr(). For example: | ord('a') | will return '97'	| ord(identifier) |

- ✓ **Built-in Type Functions**
- ✓ Below is a selection of Python built-in functions all focused on determining variable types. They can be used either to establish the type of the variable or alter it. A description for each and a code example when utilizing them inside Python to make variables are also given.

PROGRAMMING WITH PYTHON

Built-in Type Functions	Description of Function	Code Example. Creating variables with Indentifier A
str()	Creates a string variable.	| A = str('Hello') |
bool()	Boolean. Returns either True or False.	| A = bool(1) |
int()	Creates a integer number variable.	| A = int(42) |
float()	Creates a floating point variable constructed from a number or string.	| A = float(4.5) |
complex()	Returns a complex number. This number is a combination of a real and an imaginary component notated with a j (representing the square root of negative one).	| A = complex(6j) |
list()	Creates a list variable. This is a mutable sequence type.	| A = list((1, 'Hello', 3.4)) |
tuple()	Creates a tuple variable. This is an immutable sequence type.	| A = tuple((1, 'Hello', 3.4)) |
range()	Creates an immutable sequence type. Starts form 0 and increments increases by one (default).	| A = range(42) |
set()	Creates a set variable, mutable object. Sets are unordered and every element is unique.	| A = set((1, 'Hello', 3.4)) |
frozenset()	Creates a frozenset variable.	| A = frozenset((1, 'Hello', 3.4)) |
dict()	Creates a dictionary array.	| A = dict(name='JJ', age 26) |
bytes()	Creates a sequence of single bytes data and is immutable.	| A = bytes(42) |
bytearray()	Creates an array of bytes data, the same as bytes function, but is mutable.	| A = bytearray(42) |
memoryview()	Creates a memory view variable. Used for low level access of bytes and bytearray data.	| A = memoryview(bytes(42)) |

- ✓ **All Other Built-in Functions**
- ✓ And here is the doozy! This is a list of all the other functions built-in for default Python. The first six are commonly utilized functions. After the first six, the rest of the functions are given in alphabetical order. If the function is a Boolean or math type function it is explicitly mentioned with special tags to make them easier to find. The description of the Python function and the syntax of written code is also given.

PROGRAMMING WITH PYTHON

Built-in Function	Description of Python Function	Syntax of Code
type()	Returns the type of a particular variable.	| type(identifier) |
len()	Returns the length (the number of items) of a particular variable. It can be a sequence or a collection.	| len(identifier) |
print()	Prints the data of variables to the Python IDLE Shell (the standard output device).	| print(identifier) |
help()	Runs the built-in help system in the Python IDLE Shell. This gives a large amount of information about modules, libraries, functions, class, methods and keywords.	| help() |
locals()	Returns an updated dictionary of the current local symbol table.	| locals() |
globals()	Return the current global symbol table as a dictionary.	| globals() |
abs()	Math. Returns absolute value of a number. This produces the non-negative value of the variable without regards to its sign. If the number is complex it will return the magnitude.	| abs(identifier) |
all()	Boolean. True if all elements of iterable are true or If the iterable is empty	| all(iterable identifier, /) |
any()	Boolean. True if any element of iterable is true or if the iterable is empty.	| any(iterable identifier) |
breakpoint()	Method for debugging. Advanced.	| breakpoint(*args, **kwargs) |
callable()	Boolean. Returns True if the data can be callable. This means other functions can access this data	| callable(identifier) |
classmethod()	Converts a method into a class method. A class method is a method which is bound to the class and not the object of the class.	| classmethod(function) |
compile()	Returns the source into a variable ready to be executed/worked with.	| compile(source, filename, mode) |
delattr()	Delete attribute. This will delete the specific attribute from the specified data. Can remove either a property or method of a variable.	| delattr(identifier, name) |
dir()	Returns a list of the specified data properties and methods.	| dir(identifier) |
divmod()	Math. Returns the quotient and returns the remainder when dividing the first variable by the second variable.	| divmod(identifier1, identifier2) |
enumerate()	Creates a enumerate variable. Allows you to loop over a variable (such as a tuple or list) and have an automatic count attached to each piece of data. Default count starts at 0.	| enumerate(identifier, start = 0) |
eval()	Evaluates and executes an expression. Locals can be any mapping variable, globals must be in the dictionary.	| eval(identifier, globals, locals, /) |
exec()	Executes specified code or data. Supporting the dynamic execution of Python code.	| exec(identifier, globals, locals, /) |
filter()	Results in a variable created from iterable data with certain values excluded.	| filter(function, iterable identifier) |
format()	Formats a specified data value.	| format(value[, format_spec) |
getattr()	Get attribute. Return the value of the named attribute of variable. The name must be a string. Gives information on property or method to the data.	| getattr(identifier, name[,default]) |
hasattr()	Boolean. Has Attribute. Returns True if the specified variable has the particular attribute searched for.	| hasattr(identifier, name) |
hash()	Returns the hash value of the named variable. Hash values are just integers which are used to compare dictionary keys during a dictionary lookup fast.	| hash(identifier) |
id()	Returns the ID of a variable. This is an integer value which is unique and constant throughout its lifetime. In CPython this is the address of the object in memory.	| id(identifier) |
input()	Allows user input on the Python IDLE Shell when code is running from the Python Programming Window.	| input([prompt]) |
isinstance()	Boolean. Returns True if a specified data is a class instance of other specified data. Class instance is a variable which has all of the attributes described in the class definition.	| isinstance(identifier, classinfo) |
issubclass()	Boolean. Returns True if a specified class is a subclass of a specified variable. Could be a direct, indirect or virtual.	| issubclass(identifier, classinfo) |
iter()	Returns an iterator variable. An iterator is a variable that contains a countable number of values.	| iter(identifier, sentinel) |
map()	Returns an iterator variable that applied a function to every single item of the targeted iterable variable.	| map(function, iterable, ...) |
max()	Returns the largest item in a iterable variable or the largest of two or more arguments.	| max(iterable identifier) |
min()	Returns the smallest item in a iterable variable or the smallest of two or more arguments.	| min(iterable identifier) |
next()	Returns the next item in an iterable variable.	| next(iterable identifier[, default]) |
object()	Returns a new featureless variable. This variable is a base for all classes.	| object() |
open()	Open a file and return a file object. If the file cannot be opened an OSError is raised. Using mode gives different methods to open the file.	| open(file, mode = 'r', buffering =-1, encoding = None, errors = None, newline=None,closefd=True, opener=None) |
pow()	Math. Returns the value of the first variable to the power exponential.	| pow(base, exp, mod=None) |
property()	Returns a property attribute. Can get, set and delete the property using this function as well.	| property(fget=None, fset=None, fdel=None, doc=None) |
repr()	Returns a printable representation of the variable. Good for when non-ascii characters are abound.	| repr(identifier) |
reversed()	Returns a reversed iterator variable.	| reversed(iterator identifier) |
round()	Math. Method to round numbers. Can determine exactly how many digits you round to.	| round(number, number of digits = None) |
setattr()	Set Attribute. This function assigns an attribute to a named variable. The name must be a string. Gives property or method of the data. The counterpart of getattr().	| setattr(variable, name, value) |
slice()	Returns a sliced object.	| slice(start, stop, step) |
sorted()	Returns a sorted list from the items in an iterable.	| sorted(iterable identifier, /, *, key=None, reverse=False) |
@staticmethod()	Transforms a method into a static method. Static method do not require an implicit first argument. They are similar to methods found in Java or C++. Good for integration.	| @staticmethod(function) |
sum()	Math. Sums the items of data in an interator. Data cannot be a string.	| sum(iterable identifier, /, start=0) |
super()	Creates a variable that represents the parent class. Good way to look up attributes of data.	| super(type, identifier) |
vars()	Returns the dictonary property of an object (__dict__ attribute).	| vars([identifier]) |
zip()	Returns an iterator from two or more iterators.	| zip(iterable identifier1, iterable identifier 2, cont...) |

Python String and String Methods

- ✓ Python provides a lot of built-in functions that eases the writing of code. In this article, you will find the list of all built-in functions of Python.
- ✓ Python string is a sequence of Unicode characters that is enclosed in the quotations marks. In this article, we will discuss the in-built function i.e. the functions provided by the Python to operate on strings.
- ✓ Note: Every string method does not change the original string instead returns a new string with the changed attributes.

String Methods

1. **capitalize() -** Python String capitalize() method returns a copy of the original string and converts the first character of the string to a capital (uppercase) letter, while making all other characters in the string lowercase letters.

   Ex-

   ```
   name = "ankit rami"
   print(name.capitalize())
   ```

   Output- Ankit rami

2. **count() -** Python String count() function is an inbuilt function in python programming language that returns the number of occurrences of a substring in the given string.

Ex-

```
string = "My Name is Ankit,Hello Ankit Rami"
print(string.count("Ankit"))
```

Output- 2

3. **upper()** – Upper method converts all lowercase characters in a string into uppercase characters and returns it

   Ex-

   ```
   text = "Ankit Rami"
   print(text.upper())
   ```

   Output- ANKIT RAMI

4. **lower() -** Python String lower() method converts all uppercase characters in a string into lowercase characters and returns it. In this article, we will cover how to convert uppercase to lowercase in Python.

   Ex-

   ```
   text = " ANKIT RAMI "
   print(text.lower())
   ```

   Output- ankit rami

5. **startswith() -** Python String startswith() method returns True if a string starts with the specified prefix (string). If not, it returns False.

   Ex-

   ```
   string = "Ankit Rami from Rupal"
   print(string.startswith("Ankit"))
   ```

   Output - True

6. **endswith() -** Python String endswith() method returns True if a string ends with the given suffix, otherwise returns False.

   Ex-

   ```
   string = "Ankit Rami from Rupal"
   print(string.endswith("Rupal"))
   ```

   Output - True

7. **find() -** Python String find() method returns the lowest index or first occurrence of the substring if it is found in a given string. If it is not found, then it returns -1.

   Ex-

   ```
   word = 'I am Ankit Rami'
   print(word.find('Ankit'))
   ```

   Output - 5

8. **index() -** Python String index() Method allows a user to find the index of the first occurrence of an existing substring inside a given string.

   Ex-

   ```
   string = 'A and B'
   print(string.index('and'))
   ```

   Output - 2

9. **islower() -** Python String islower() method checks if all characters in the string are lowercase. This method returns True if all alphabets in a string are lowercase alphabets. If the string contains at least one uppercase alphabet, it returns False.

Ex-

```
a="ankit rami"
print(a.islower())
```

Output - True

10. **isupper() -** Python String isupper() method returns whether all characters in a string are uppercase or not.

Ex-

```
a="ANKIT"
print(a.isupper())
```

Output - True

11. **join() –** join() is an inbuilt string function in Python used to join elements of the sequence separated by a string separator. This function joins elements of a sequence and makes it a string.

Ex-

```
text = "ANKIT"
print("$".join(text))
```

Output - ANKIT

12. **partition()** - Python String partition() method splits the string at the first occurrence of the separator and returns a tuple containing the part before the separator, separator and the part after the separator. Here, the separator is a string that is given as the argument.

Ex-

```
text = "I am Ankit Rami"
print(text.partition("am"))
```

Output - ('I ', 'am', 'ANkit Rami')

13. **isalnum() -** Python String isalnum() method checks whether all the characters in a given string are either alphabet or numeric (alphanumeric) characters.

Ex-

```
string = "abc123"
print(string.isalnum())
```

Output - True

14. **isalpha() -** Python String isalpha() method is used to check whether all characters in the String is an alphabet.

Ex-

```
string = "ankit"
print(string.isalpha())
```

Output - True

15. **isdigit() -** Python String isdigit() method returns “True” if all characters in the string are digits, Otherwise, It returns “False”.

Ex-

```
string = '1234'
print(string.isdigit())
```

Output - True

Python List and List Function

- ✓ A list in Python is used to store the sequence of various types of data.
- ✓ A list can be defined as a collection of values or items of different types.
- ✓ Python lists are mutable type which implies that we may modify its element after it has been formed.
- ✓ The items in the list are separated with the comma (,) and enclosed with the square brackets [].
- ✓ Python lists are identical to dynamically scaled arrays that are specified in other languages, such as Java's Array List and C++'s vector.

Characteristics of Lists

- ✓ The lists are ordered.
- ✓ The element of the list can access by index.
- ✓ The lists are the mutable type.
- ✓ The lists are mutable types.
- ✓ A list can store the number of various elements.

Create Simple List Example -

```
thislist = ["AR", "MR", "RJ"]
print(thislist)
```

Output - ["AR", "MR", "RJ"]

✓ List Function and List operations

1. **append() -** Adds an element at the end of the list.
   Ex –
   ```
   fruits = ["apple", "banana", "cherry"]
   fruits.append("orange")
   print(fruits)
   ```
   Output - ['apple', 'banana', 'cherry', 'orange']
2. **clear() -** Removes all the elements from the list
   Ex –
   ```
   fruits = ["apple", "banana", "cherry"]
   fruits.clear()
   print(fruits)
   ```
   Output - []
3. **copy() -** Returns a copy of the list
   Ex –
   ```
   fruits = ["apple", "banana", "cherry"]
   x = fruits.copy()
   print(x)
   ```
   Output - ["apple", "banana", "cherry"]
4. **count() -** Returns the number of elements with the specified value
   Ex –
   ```
   fruits = ["apple", "banana", "cherry"]
   x = fruits.count("cherry")
   print(x)
   ```
   Output - 1

5. **extend() -** Add the elements of a list (or any iterable), to the end of the current list

   Ex –

```
fruits = ['apple', 'banana', 'cherry']
cars = ['Ford', 'BMW', 'Volvo']
fruits.extend(cars)
print(fruits)
```

   Output - ['apple', 'banana', 'cherry', 'Ford', 'BMW', 'Volvo']

6. **index() -** Returns the index of the first element with the specified value.

   Ex –

```
fruits = ['apple', 'banana', 'cherry']
x = fruits.index("cherry")
print(x)
```

   Output - 2

7. **insert() -** Adds an element at the specified position

   Ex –

```
fruits = ['apple', 'banana', 'cherry']
fruits.insert(1, "orange")
print(fruits)
```

   Output - ['apple','orange','banana','cherry']

8. **pop() -** Removes the element at the specified position

   Ex –

```
fruits = ['apple', 'banana', 'cherry']
fruits.pop(1)
print(fruits)
```

   Output - ['apple', 'cherry']

9. **remove() -** Removes the first item with the specified value

   Ex –

```
fruits = ['apple', 'banana', 'cherry']
fruits.remove("banana")
print(fruits)
```

   Output - ['apple', 'cherry']

10. **reverse() -** Reverses the order of the list

   Ex –

```
fruits = ['apple', 'banana', 'cherry']
fruits.reverse()
print(fruits)
```

   Output - ['cherry','banana','apple']

11. **sort() -** Sorts the list

   Ex –

```
cars = ['Ford', 'BMW', 'Volvo']
cars.sort()
print(cars)
```

   Output - ['BMW', 'Ford', 'Volvo']

12. **cmp() -** compares elements of two lists.

   Ex-

```
list1=[1,2,3,4,5]
list2=[1,2,3,4,5]
print(cmp(list1,list2))
```

   Output - 1

13. **len(list) -** Gives the total length of the list.

Ex –

```
list=[1,2,3,4,5]
print(len(list))
```

Output - 5

14. **max(list) -** Returns item from the list with max value.

Ex –

```
list=[1,2,3,4,5]
print(max(list))
```

Output - 5

15. **min(list) -** Returns item from the list with min value.

Ex –

```
list=[1,2,3,4,5]
print(min(list))
```

Output – 1

Python Tuple and Tuple Function

- ✓ Tuples are used to store multiple items in a single variable.
- ✓ Tuple is one of 4 built-in data types in Python used to store collections of data.
- ✓ A tuple is a collection which is ordered and unchangeable.
- ✓ Tuples are written with round () brackets.
- ✓ A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.

- ✓ **Ex –**

```
tuple1 = ("apple", "banana", "cherry")
print(tuple1)
```

Output – ('apple', 'banana', 'cherry')

- ✓ Tuple items are ordered, unchangeable, and allow duplicate values.
- ✓ Tuple items are indexed, the first item has index [0], the second item has index [1] etc.
- ✓ **Ex –**

```
tuple1 = ("apple", "banana", "cherry", "apple", "cherry")
print(tuple1)
```

Output – ("apple", "banana", "cherry", "apple", "cherry")

✓ Tuple Function

1. **Length -** To determine how many items a tuple has, use the len() function

   Ex -

```
t1 = tuple("apple", "banana", "cherry")
print(len(t1))
```

   Output – 3

2. **count() -** This function will help us to fund the number of times an element is present in the tuple.

   Ex -

```
t1 = tuple(11,22,33,44,11)
print(t1.count(11))
```

   Output – 2

3. **index() -** The tuple index() method helps us to find the index or occurrence of an element in a tuple.
   Ex -
   t1 = tuple(11,22,33,44,11)
   print(t1.index(33))
   Output – 2

4. **sorted() -** This method takes a tuple as an input and returns a sorted list as an output.
   Ex -
   t1 = tuple(11,22,44,55,33)
   print(sorted(t1))
   Output - 11,22,33,44,55

5. **min() -** gives the smallest element in the tuple as an output. Hence, the name is min().
   Ex -
   t1 = tuple(11,22,44,55,33)
   print(min(t1))
   Output – 11

6. **max() -** gives the largest element in the tuple as an output. Hence, the name is max().
   Ex -
   t1 = tuple(11,22,44,55,33)
   print(max(t1))
   Output - 55

Creating Tuple and Accessing Tuple Values

✓ **Access Tuple Items using Index No**

Ex-

```
t1 = (11,22,33,44,55,66)
print(t1[1])
```

Output – 22

✓ **Access Tuple Items using Negative Indexing No.**

✓ Negative indexing means start from the end.

Ex-

```
t1 = (11,22,33,44,55,66)
print(t1[-1])
```

Output – 66

✓ **Access Tuple Items using Range of Indexes No**

✓ You can specify a range of indexes by specifying where to start and where to end the range.

✓ When specifying a range, the return value will be a new tuple with the specified items.

Ex-

```
t1 = (11,22,33,44,55,66)
print(t1[2:5])
print(t1[-4:-1])
```

Output –

33,44,55

33,44,55

- ✓ **Check if Item Exists in Tuple or not.**
- ✓ To determine if a specified item is present in a tuple use the in keyword

Ex-

```
t1 = (11,22,33,44,55,66)
if 55 in t1:
  print("Yes")
else:
  print("No")
```

Output – Yes

- ✓ **Print Tuple using Loop**

Ex-

```
t1 = (11,22,33,44,55)
for x in t1:
  print(x)
```

Output – 11 22 33 44 55

- ✓ **Print Tuple using Loop and Index No**

Ex-

```
t1 = (11,22,33,44,55)
for i in range(len(t1)):
  print(t1[i])
```

Output – 11 22 33 44 55

- ✓ **Change Tuple Values**
- ✓ Once a tuple is created, you cannot change its values. Tuples are unchangeable or immutable as it also is called.

- ✓ But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

  Ex-

```
x = (11,22,33,44,55)
y = list(x)//convert to list
y[1] = 99 // change 1 index value 22 to 99
x = tuple(y)//convert to tuple
print(x)
```

  Output – 11 99 33 44 55

- ✓ **Join Tuple Values**
- ✓ join two or more tuples you can use the + operator.

  Ex-

```
t1 = (11,22,33,44,55)
t2 = (66,77,88,99)
t3=t1+t2
print(t3)
```

  Output – 11 22 33 44 55 66 77 88 99

- ✓ **Multiply Tuples**
- ✓ If you want to multiply the content of a tuple a given number of times, you can use the * operator

  Ex-

```
t1 = (11,22,33,44,55)
t2=t1*2
print(t2)
```

  Output – 11 22 33 44 55 11 22 33 44 55

- ✓ **Add Items in Tuple**

  Ex-

```
x = (11,22,33,44,55)
y = list(x)//convert to list
y.append(99) // add Value
x = tuple(y)//convert to tuple
print(x)
```

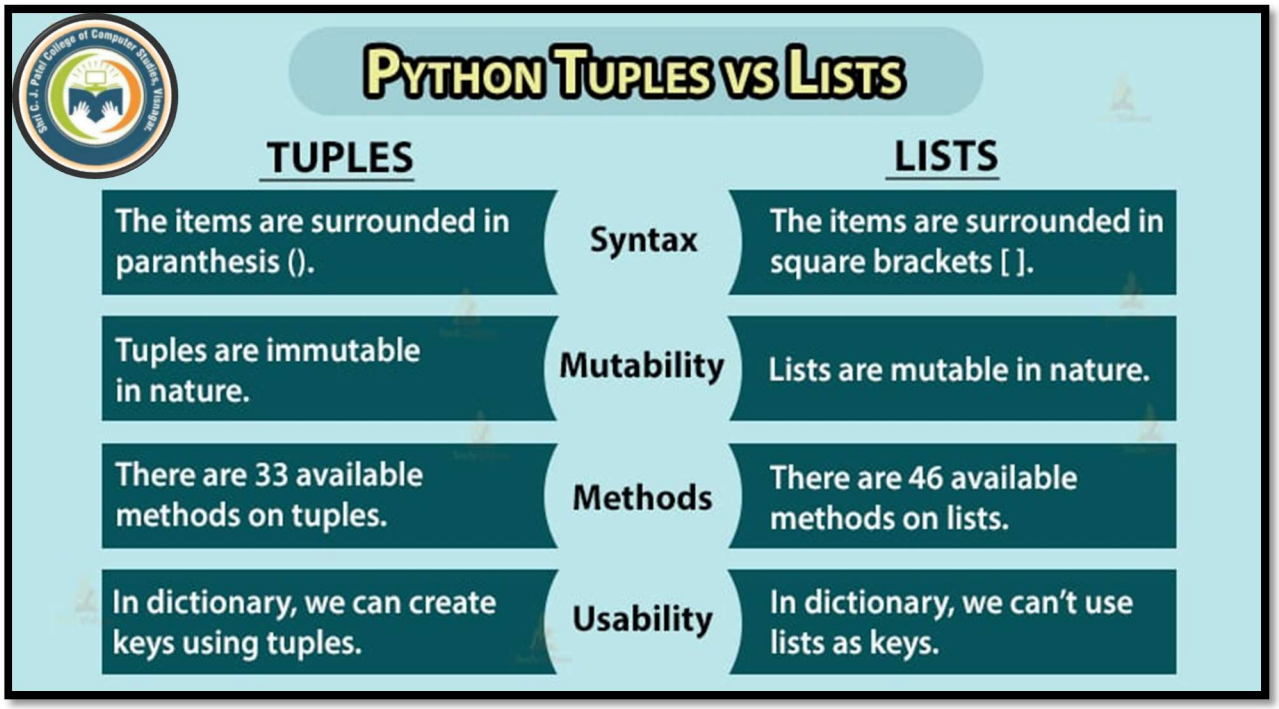
  Output – 11 22 33 44 55 99

- ✓ **Remove Items in Tuple**

  Ex-

```
x = (11,22,33,44,55)
y = list(x)//convert to list
y.remove(33) // remove Value
x = tuple(y)
print(x)
```

  Output – 11 22 44 55



Extra Topics Related to Python Like Dictionary and SET

Set in Python

- ✓ Sets are used to store multiple items in a single variable.
- ✓ A set is a collection which is unordered, unchangeable*, and unindexed.
- ✓ Used curly bracket { } bracket to create Set.
- ✓ Duplicates Not Allowed in Set.
- ✓ **Ex -**

```
s1 = {"apple", "banana", "cherry"}
print(s1)
```

Output –

```
{'apple', 'banana', 'cherry'}
```

Dictionary in Python

- ✓ Dictionaries are used to store data values in key: value pairs.
- ✓ A dictionary is a collection which is ordered*, changeable and do not allow duplicates.
- ✓ Dictionaries are written with curly brackets, and have keys and values.
- ✓ **Ex-**

```
d1 =    {
  "brand": "Samsung",
  "model": "S21FE",
  "year": 2022
}
print(d1)
```

Output –

```
{'brand': 'Samsung', 'model': 'S21FE', 'year': 2022}
```

Reference Link

1. https://www.w3schools.com/python/python_ref_functions.asp
2. https://www.geeksforgeeks.org/python-string-methods/
3. https://www.tutorialspoint.com/python/python_strings.htm
4. https://www.tutorialspoint.com/python/python_lists.htm
5. https://www.w3schools.com/python/python_ref_list.asp

Any Query Contact Us

Faculty Name- Ankit Rami

Email – ankitramiblog@gmail.com

Contact No – +91 8460467193

Website - amit.arinfoway.com

Subscribe Our YouTube Channel

https://www.youtube.com/channel/UCWbJh2iQ8w-8nrU0Xpjpw7g