Built in data types in python

**1. List**

Lists are created using square brackets.

\*Ordered

\*Changeable

\*Allow duplicates

**List length**

Use len() function

Eg:

L = ["apple", "banana", "cherry"]

Print(len(L))

**Data types**

X= ["apple", "banana", "cherry"]

Print(type(x))

**List() constructor**

X=list("apple", "banana", "cherry")

Print(x)

**Access list items**

X=["apple", "banana", "cherry"]

Print(X[1])

**Check if item exists**

X=["apple", "banana", "cherry"]

if “apple” in x:

print(“Yes, apple exist in the list”)

**change item values**

X=["apple", "banana", "cherry"]

X[1]=”kiwi”

Print(X)

**Insert item**

The insert() method inserts an item at the specified index.

Eg:

X = ["apple", "banana", "cherry"]

X.insert(1,”orange”)

Print(X)

**Append items**

X = ["apple", "banana", "cherry"]

X.append(”orange”)

Print(X)

**Extend list**

X = ["apple", "banana", "cherry"]

Y = [“red”,”green”,”blue”]

X.extend(Y)

Print(X)

**Add any iterable**

X = ["apple", "banana", "cherry"]

Y = (“red”,”green”,”blue”)

X.extend(Y)

Print(X)

**Remove specified item**

X = ["apple", "banana", "cherry"]

X.remove(“cherry”)

Print(X)

**Remove specified index**

X = [“apple”,”orange”,”cherry”]

x.pop(1)

print(x)

If you do not specify the index, the pop() method removes the last item.

X = [“apple”,”orange”,”cherry”]

x.pop()

print(x)

The del keyword also removes the specified index.

X = [“apple”,”orange”,”cherry”]

del x[0]

print(x)

The del keyword can also delete the list completely.

X = [“apple”,”orange”,”cherry”]

del x

**Clear the list**

The clear() method empties the list.

The list still remains, but it has no content.

X = [“apple”,”orange”,”cherry”]

x.clear()

print(x)

**Loop through a list**

You can loop through the list items by using a for loop.

x = [“apple”, “orange”, “cherry”]

for i in x:

print(x)

**Loop through the index number**

x = [“apple”, “orange”, “cherry”]

for i in range(len(x)):

print(x[i])

**Using a while loop**

x = [“apple”, “orange”, “cherry”]

i = 0

while i < len(x):

print(x[i])

i = i + 1

**Sort list alphanumerically**

List objects have a sort() method that will sort the list alphanumerically, ascending, by default.

x = [“apple”, “orange”, “cherry”]

x.sort()

print(x)

**Sort descending**

To sort descending, use the keyword argument reverse = True.

x = [“apple”, “orange”, “cherry”]

x.sort(reverse = True)

print(x)

**Copy a list**

There are ways to make a copy, one way is to use the built-in List method copy().

x = [“apple”, “orange”, “cherry”]

newlist = x.copy()

print(newlist)

Another way to make a copy is to use the built-in method list().

x = [“apple”, “orange”, “cherry”]

newlist = list(x)

print(newlist)

**Join two lists**

One of the easiest ways are by using the + operator.

list1 = ["a", "b", "c"]  
 list2 = [1, 2, 3]

list3 = list1 + list2  
 print(list3)

Another way to join two lists is by appending all the items from list2 into list1.

list1 = ["a", "b", "c"]  
 list2 = [1, 2, 3]  
 for x in list2:  
  list1.append(x)  
 print(list1)

Or you can use the extend() method.

list1 = ["a", "b", "c"]  
 list2 = [1, 2, 3]  
 list1.extend(list2)  
 print(list1)

**2. Tuple**

Tuples are written with round brackets.

\*Ordered

\*Unchangeable

\*Allow duplicates

**Tuple length**

X = (“apple”,”orange”,”kiwi”,”cherry”)

Print(len(x))

**Tuple datatype**

X = (“apple”,”orange”,”kiwi”,”cherry”)

Print(type(x))

**Tuple() constructor**

X =tuple( (“apple”,”orange”,”kiwi”,”cherry”))

Print(x)

**Access tuple items**

X = (“apple”,”orange”,”kiwi”,”cherry”)

Print(x[2])

**Change tuple values**

X = (“apple”,”orange”,”kiwi”,”cherry”)

Y =list(X)

Y[1]=”grape”

X=tuple(y)

Print(x)

**Check if item exists**

thistuple = ("apple", "banana", "cherry")

if "apple" in thistuple:  
   print("Yes, 'apple' is in the fruits tuple")

**Add items**

Tuples are immutable, they do not have a build-in append() method, but there are other ways to add items to a tuple.

1. **Convert into a list:**

You can convert it into a list, add your item(s), and convert it back into a tuple.

thistuple = ("apple", "banana", "cherry")  
y = list(thistuple)  
y.append("orange")  
thistuple = tuple(y)

1. **Add tuple to a tuple:**

thistuple = ("apple", "banana", "cherry")  
y = ("orange",)  
thistuple = thistuple + y  
print(thistuple)

**Remove items**

Tuples are **unchangeable**, so you cannot remove items from it.

Convert the tuple into a list, remove “apple”, and convert it back into a tuple.

thistuple = ("apple", "banana", "cherry")  
 y = list(thistuple)  
 y.remove("apple")  
 thistuple = tuple(y)

Or you can delete the tuple completely.

t = ("apple", "banana", "cherry")  
 del t  
 print(t)

**Tuple unpacking**

When we create a tuple, we normally assign values to it. This is called "packing" a tuple.

But, in Python, we are also allowed to extract the values back into variables. This is called "unpacking".

eg:

fruits = ("apple", "orange", "banana")

(red,green,blue) = fruits

print(red)

print(green)

print(blue)

**Loop through a tuple**

x = ("apple", "banana", "cherry")  
 for i in x:  
   print(x)

**Loop through the index numbers**

x = ("apple", "banana", "cherry")  
 for i in range(len(x)):

print(thistuple[i])

**Join tuples**

T1 = (“a”, “b”, “c”, “d”)

T2 = (1,2,3,4)

T3 = T1 + T2

Print(T3)

**Multiply tuples**

x = ("apple", "banana", "cherry")  
 for i in range(len(x)):

print(x[i])

**3. Set**

Sets are written with curly brackets.

\*Unordered

\*Unchangeable

\*Not allow duplicates

Length of a set

s = {"apple", "banana", "cherry"}

print(len(s))

**Data type**

set1 = {"apple", "banana", "cherry"}  
 print(type(set1))

**set() constructor**

s = set(("apple", "banana", "cherry"))  
 print(s)

**Access items**

You cannot access items in a set by referring to an index or a key. But you can loop through the set items using a for loop.

s = {"apple", "banana", "cherry"}

for x in s:

print(x)

**Add items**

Once a set is created, you cannot change its items, but you can add new items.

To add one item to a set use the add() method.

s = {"apple", "banana", "cherry"}  
 s.add("orange")  
 print(s)

**Add sets**

To add items from another set into the current set, use the update() method.

thisset = {"apple", "banana", "cherry"}  
 tropical = {"pineapple", "mango", "papaya"}  
 thisset.update(tropical)  
 print(thisset)

**Remove item**

To remove an item in a set, use the remove(), or the discard() method.

s = {"apple", "banana", "cherry"}  
 s.remove("banana")  
 print(s)

Remove “banana” by using the discard() method.

s = {"apple", "banana", "cherry"}  
 s.discard("banana")  
 print(s)

You can also use pop() method to remove an item.

s = {"apple", "banana", "cherry"}  
 x = s.pop()  
 print(x)  
 print(s)

Sets are unordered, so when using the pop() method, you do not know which item that gets removed.

The clear() method empties the set.

s = {"apple", "banana", "cherry"}  
 s.clear()  
 print(s)

The del keyword will delete the set completely.

s = {"apple", "banana", "cherry"}  
 del s  
 print(s)

**Loop items**

s = {"apple", "banana", "cherry"}  
 for x in s:  
  print(x)

**Join two sets**

There are several ways to join two or more sets in Python.

1. **Using union() method**

s1 = {"a", "b", "c"}  
 s2 = {1, 2, 3}  
 s3 = s1.union(s2)  
 print(s3)

1. **Using update() method**

s1 = {"a", "b", "c"}  
 s2 = {1, 2, 3}  
 s1.update(s2)  
 print(s1)

**Keep only the duplicates**

The intersection\_update() method will keep only the items that are present in both sets.

x = {"apple", "banana", "cherry"}  
 y = {"google", "microsoft", "apple"}  
 x.intersection\_update(y)

print(x)

The intersection() method will return a new set, that only contains the items that are present in both sets.

x = {"apple", "banana", "cherry"}  
 y = {"google", "microsoft", "apple"}  
 z = x.intersection(y)  
 print(z)

**Keep all, but not the duplicates**

The symmetric\_difference\_update() method will keep only the elements that are NOT present in both sets.

x = {"apple", "banana", "cherry"}  
 y = {"google", "microsoft", "apple"}  
  
 x.symmetric\_difference\_update(y)  
 print(x)

The symmetric\_difference() method will return a new set, that contains only the elements that are NOT present in both sets.

x = {"apple", "banana", "cherry"}  
 y = {"google", "microsoft", "apple"}  
 z = x.symmetric\_difference(y)  
 print(z)

**4. Dictionaries**

Dictionaries are written with curly brackets, and have keys and values.

\*Ordered

\*Changeable

\*Not allow duplicates

Eg:

thisdict = {“name”:”anu”, ”place”:”Malappuram”, “year”:2022}  
 print(thisdict)

Dictionary items are presented in key:value pairs.

**Dictionary length**

thisdict = {“name”:”anu”, ”place”:”Malappuram”, “year”:2022}

print(len(thisdict))

**Data type**

thisdict = {“name”:”anu”, ”place”:”Malappuram”, “year”:2022}

print(type(thisdict))

**Accessing items**

You can access the items of a dictionary by referring to its key name, inside square brackets.

thisdict = {"brand": "Ford", "model": "Mustang","year": 1964}  
 x = thisdict["model"]

There is also a method called get() that will give you the same result.

thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964}

x = thisdict.get("model")

print(x)

**Get keys**

The keys() method will return a list of all the keys in the dictionary.

thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964}

x = thisdict.keys()

print(x)

**Get values**

The values() method will return a list of all the values in the dictionary.

thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964}

x = thisdict.values()

print(x)

**Get items**

The items() method will return each item in a dictionary, as tuples in a list.

thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964}

x = thisdict.items()

print(x)

**Check if key exists**

thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964}

if “model” in thisdict:

print(“Yes, ‘model’ is one of the keys in the thisdict dictionary”)

**Change values**

thisdict = {  
   "brand": "Ford",  
   "model": "Mustang",  
   "year": 1964

}  
 thisdict["year"] = 2018

Print(thisdict)

**Update dictionary**

The update() method will update the dictionary with the items from the given argument.

thisdict = {  
   "brand": "Ford",  
   "model": "Mustang",  
   "year": 1964  
 }  
 thisdict.update({"year": 2020})

**Adding items**

thisdict = {  
   "brand": "Ford",  
   "model": "Mustang",  
   "year": 1964  
 }  
 thisdict["color"] = "red"  
 print(thisdict)

**Update dictionary**

The update() method will update the dictionary with the items from a given argument. If the item does not exist, the item will be added.

thisdict = {  
   "brand": "Ford",  
   "model": "Mustang",  
   "year": 1964  
 }  
 thisdict.update({"color": "red"})

print(thisdict)

**Remove items**

1. **Pop() method**

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.pop("model")  
print(thisdict)

1. **Popitem() method**

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
 }  
 thisdict.popitem()  
 print(thisdict)

1. **Del**
2. thisdict = {  
     "brand": "Ford",  
     "model": "Mustang",  
     "year": 1964  
    }  
    del thisdict["model"]  
    print(thisdict)
3. thisdict = {  
     "brand": "Ford",  
     "model": "Mustang",  
     "year": 1964  
   }  
   del thisdict  
   print(thisdict)
4. **Clear() method**

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
 }  
 thisdict.clear()  
 print(thisdict)

**Loop through a dictionary**

a) thisdict = {"brand": "Ford","model": "Mustang","year": 1964}

for x in thisdict:

print(x)

b) thisdict = {"brand": "Ford","model": "Mustang","year": 1964}

for x in thisdict:

print(thisdict[x])

c) thisdict = {"brand": "Ford","model": "Mustang","year": 1964}

for x in thisdict.values():

print(x)

d)thisdict = {"brand": "Ford","model": "Mustang","year": 1964}

for x in thisdict.keys():  
   print(x)

e) thisdict = {"brand": "Ford","model": "Mustang","year": 1964}

for x, y in thisdict.items():  
   print(x, y)

**Copy a dictionary**

1. **Using the built-in Dictionary method copy().**

thisdict = {  
   "brand": "Ford",  
   "model": "Mustang",  
   "year": 1964  
 }  
 mydict = thisdict.copy()  
 print(mydict)

1. **Using the built-in function dict().**

thisdict = {  
   "brand": "Ford",  
   "model": "Mustang",  
   "year": 1964  
 }

mydict = dict(thisdict)  
 print(mydict)

**Nested dictionaries**

A dictionary can contain dictionaries, this is called nested dictionaries.

child1 = {

"name" : "Emil",

"year" : 2004

}

child2 = {

"name" : "Tobias",

"year" : 2007

}

child3 = {

"name" : "Linus",

"year" : 2011

}

myfamily = {

"child1" : child1,

"child2" : child2,

"child3" : child3

}

print(myfamily)