***# list []– ordered-changable-allow duplicate  
# touple ()- ordered-unchangable-allow duplicate  
# set {}-un ordered,-unchangable-unindexed-no duplicate  
# dictionary {}-ordered-changable-no duplicate***

***Store multiple items in single variable***

**list**

***example  
# list[0,1,2,3]***fruits=[]  
print(type(fruits))  
*# append allows to add only one argument*fruits.append(**"apple"**)  
print(fruits)  
fruits.append(**"orange"**)  
print(fruits)  
fruits.append(**"mango"**)  
print(fruits)  
fruits.remove(**"apple"**)  
print(fruits)  
fruits.insert(**1**,**"rambootan"**)  
print(fruits)  
vegitables=[**'onion'**,**'mango'**,**'cabbage'**,**'tomato'**,**'potato',’mango**]  
print(vegitables)  
vegitables.pop(**0**)*# item in index 0 removed*print(vegitables)  
vegitables.pop()*# index is not specified then last item removed*print(vegitables)  
**del** vegitables[**1**] *# item in index 1 is deleted*print(vegitables)  
shop=[**'onion'**,**'mango'**,**'cabbage'**,**'tomato'**,**'potato'**]  
print(shop)  
shop.clear() *# clear items in shop*print(shop)  
*# del shop# delete list shop  
# print(shop)*print(len(fruits)) *# to get the lengtn of list*fruits.sort()*# sort in alphabatically*print(fruits)  
fruits.sort(reverse=**True**) *#sort in decending order*print(fruits)  
print(shop)  
shop=fruits.copy()*# copy the items in list fruits to list shop*print(shop)  
market=fruits+vegitables *# combine list fruits and list vegetables into new list market*print(market)

Output

<class 'list'>

['apple']

['apple', 'orange']

['apple', 'orange', 'mango']

['orange', 'mango']

['orange', 'rambootan', 'mango']

['onion', 'mango', 'cabbage', 'tomato', 'potato',’mango]

['mango', 'cabbage', 'tomato', 'potato']

['mango', 'cabbage', 'tomato']

['mango', 'tomato']

['onion', 'mango', 'cabbage', 'tomato', 'potato']

[]

3

['mango', 'orange', 'rambootan']

['rambootan', 'orange', 'mango']

[]

['rambootan', 'orange', 'mango']

['rambootan', 'orange', 'mango', 'mango', 'tomato']

Touple

***# touple ()- ordered-unchangable-allow duplicate*** allow diff data types

Index based

Once tuple is created we cant change or edit items

Touple is used for unchangable items in program

car=(**'swift'**,**'inova'**,**'benz'**,**'bmw'**,**'aveo'**,**'kia'**,**122**) *# tuple car*print(type(car))  
print(car)  
print(car[**2**])  
vehicle=list(car) *# tuple car is changed to list vehicle*print(type(vehicle))  
x=tuple(vehicle)*# list converted to tuple*print(type(x))  
*# tuple is not changable once its created--  
  
#----------- tuple unpaking*t=(**10**,**34**,**25**,**100**)  
a,b,c,d=t *# we can assign values from tuple to variables like this this is called tuple unpaking*print(a)  
print(d)  
print()

output

<class 'tuple'>

('swift', 'inova', 'benz', 'bmw', 'aveo', 'kia', 122)

benz

<class 'list'>

<class 'tuple'>

10

100

Dictionary

***# dictionary\_name{‘key’:’value}***

***-ordered-changable- allows no duplicate***

*Key :Value pair*

*not index based-*

*It based on key,*

*key must be unique*

*value can be any datatype*

*keys are immutable*

The dict() function in Python is used to create a dictionary, which is a collection of key-value pairs. Here are a few ways to create a dictionary using dict():

1. **Creating an empty dictionary:**

my\_dict = dict()

# or

my\_dict = {}

1. **Creating a dictionary with key-value pairs:**

my\_dict = dict(name="Alice", age=25, city="New York")

1. **Creating a dictionary from a list of tuples:**

my\_list = [("name", "Alice"), ("age", 25), ("city", "New York")]

my\_dict = dict(my\_list)

1. **Creating a dictionary using zip:**

keys = ["name", "age", "city"]

values = ["Alice", 25, "New York"]

my\_dict = dict(zip(keys, values))

1. **Creating a dictionary from another dictionary:**

original\_dict = {"name": "Alice", "age": 25}

my\_dict = dict(original\_dict)

The **dict() constructor** is versatile and can be used in various ways to create dictionaries from different types of data.

EXMAPLE

dict\_name={key:value}

dict\_name={Key1:value, key2:value, key3:value}

dict\_name={

key1:value,

key2:value,

key3:value}

**Nested Dictonary**

*students* = {

**'student1'**: {

'name': 'Alice',

'age': 20,

'courses': ['Math', 'Physics'],

},

**'student2'**: {

'name': 'Bob',

'age': 22,

'courses': ['Chemistry', 'Biology'],

},

**'student3'**: {

'name': 'Charlie',

'age': 19,

'courses': ['History', 'Literature'],

},

}

# Accessing data from the nested dictionary

print(*students*[**'student1'**]['name']) # Output: Alice

print(*students*[**'student2'**]['courses']) # Output: ['Chemistry', 'Biology']

# Adding a new student

*students*[**'student4'**] = {

'name': 'David',

'age': 21,

'courses': ['Computer Science', 'Art'],

}

**# Updating a student's data**

students['student1']['age'] = 21

# Iterating through the nested dictionary

for student\_id, student\_info in students.items():

print(f"\n{student\_id}:")

for key, value in student\_info.items():

print(f"{key}: {value}")}

expln

 **Outer loop**: Iterates through each student in the students dictionary.

 **Inner loop**: Iterates through each attribute of the student (like name, age, and courses).

**Outer Loop: for student\_id, student\_info in students.items():**

* **Purpose**: This loop iterates over the outer dictionary (students).
* **students.items()**: This method returns a view object that displays a list of the dictionary's key-value tuple pairs. In this case, each item is a tuple where:
  + student\_id is the key (e.g., 'student1', 'student2').
  + student\_info is the value, which is another dictionary containing that student's details.

**2. Printing the student\_id: print(f"\n{student\_id}:")**

* **Purpose**: This line prints the student\_id, which is the key in the outer dictionary (e.g., 'student1', 'student2').
* **\n**: This adds a new line before printing the student\_id to ensure each student's information starts on a new line.

**3. Inner Loop: for key, value in student\_info.items():**

* **Purpose**: This loop iterates over the inner dictionary (student\_info), which contains details like name, age, and courses.
* **student\_info.items()**: Similar to the outer loop, this method returns the key-value pairs for each student's information.
  + key is the attribute name (e.g., 'name', 'age', 'courses').
  + value is the corresponding data (e.g., 'Alice', 20, ['Math', 'Physics']).

**4. Printing the key and value: print(f"{key}: {value}")**

* **Purpose**: This line prints each attribute of the student and its value.
* **Example**: For 'student1', the output will be:
  + name: Alice
  + age: 20
  + courses: ['Math', 'Physics']