Dictionaries:

```
collection of key-value pair———{key:value}
```

```
dict1 = {"name": "kris", "age":20}
```

Properties

Ordered

It is an ordered collection of elements

No Duplicate Keys

doesn't contain duplicate elements.

Mutable and Heterogeneous

There elements can be changed and contain elements of different data types

Accessing Dictionary Elements

scllerts.int soler()";

(doesn't support indexing)

variablename["keys"]

```
dict1 = {"name": "kris", "age":20}
print(dict1["age"])
```

variablename.keys()

```
dict1 = {"name": "kris", "age":20}
print(dict1.keys())
```

variablename.values()

```
dict1 = {"name": "kris", "age":20}
print(dict1.values())
```

Modifying Dictionaries

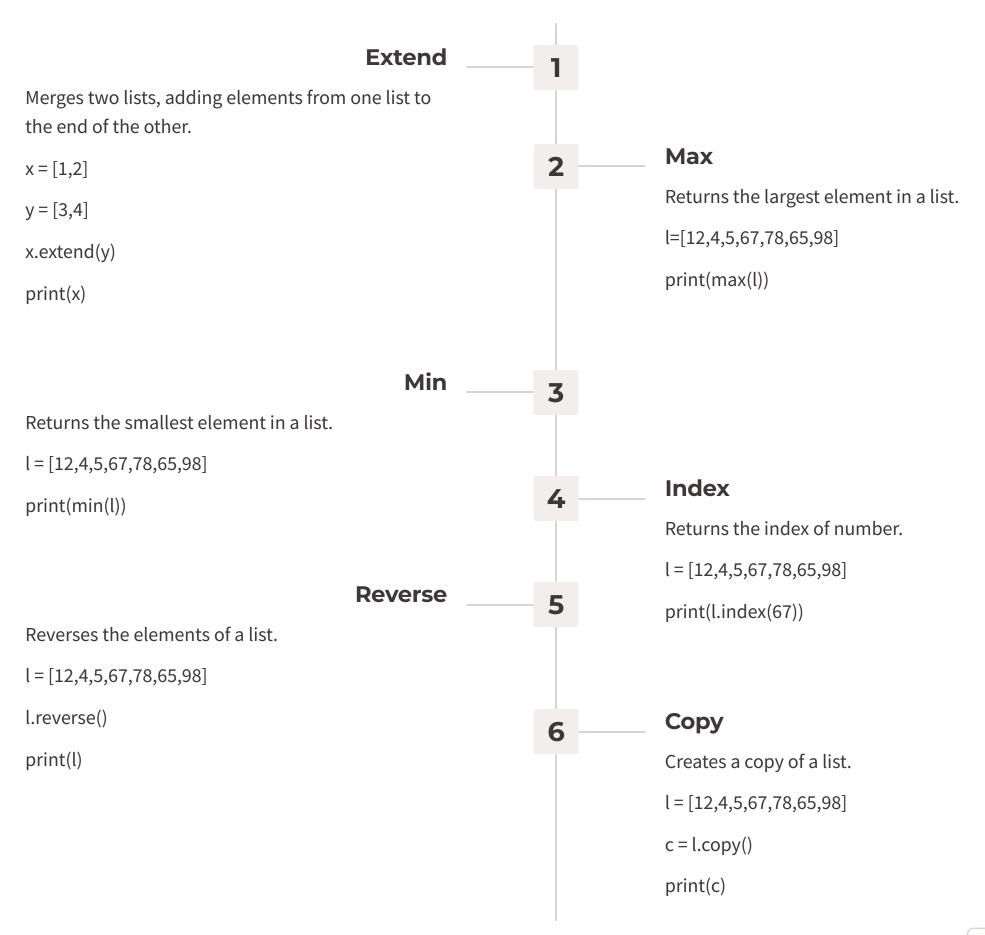
Add New Key-Value Pair

```
dict1 = {"name": "kris", "age":20}
dict1["address"] = "mohali"
dict1["id"] = 12345
print(x)
```

Remove Key-Value Pair

```
dict1 = {"name": "kris", "age":20, "address": "mohali"}
dict1.pop("age")
```

Predefined List Functions



Slicing and Sorting

Slicing

Extract a portion by specifying a start and end index.

lst = [10, 20, 30, 40, 50]

sliced = lst[1:4] #Extract elements from index 1 to 3 print(sliced)

my_string = "Hello, World!"

sliced = my_string[7:12] # Extract substring from index 7 to 11

print(sliced)

Sorting

Rearrange the elements of a list in a specific order, either ascending or descending.

lt = [40, 10, 30, 20, 50]

lt.sort() (by default in ascending)

print(lt)

2

1

for descending

my_list = [40, 10, 30, 20, 50]

my_list.sort(reverse=True)

print(my_list)