* Overview of sorting (sort and sorted)
* Using sort to sort list
* Disadvantages of using sort
* Overview of sorted
* Sort iterable of integers
* Reverse Sorting
* Sort list of tuples based on **sale amount**
* Sort list of dicts based on **sale\_amount**
* Exercises and Solutions

1. Which of the following statements is true about sorting in Python?

a) Sorting is only possible with lists.

b) Sorting can be performed with both lists and tuples.

c) Sorting is a complex process and cannot be done easily in Python.

d) Sorting is only possible with numerical data.

Answer: b) Sorting can be performed with both lists and tuples.

1. What is the purpose of using the sort() method in Python?

a) To sort the elements of a list in ascending order.

b) To sort the elements of a list in random order.

c) To create a sorted copy of the list.

d) To reverse the order of the elements in the list.

Answer: a) To sort the elements of a list in ascending order.

1. What is a disadvantage of using the sort() method in Python?

a) It is slower than using sorted().

b) It only works with lists and not other iterable types.

c) It modifies the original list, which can be undesirable in some cases.

d) It cannot sort lists with mixed data types.

Answer: c) It modifies the original list, which can be undesirable in some cases.

1. What is the purpose of using the sorted() function in Python?

a) To sort the elements of a list in ascending order.

b) To sort the elements of a list in descending order.

c) To create a sorted copy of the list.

d) To reverse the order of the elements in the list.

Answer: c) To create a sorted copy of the list.

1. How would you sort an iterable of integers in Python using the sorted() function?

a) sorted(iterable)

b) sorted(iterable, key=int)

c) sorted(iterable, reverse=True)

d) sorted(iterable, key=int, reverse=True)

Answer: a) sorted(iterable)

1. How would you sort a list in reverse order in Python?

a) list.sort()

b) sorted(list)

c) list.sort(reverse=True)

d) sorted(list, reverse=True)

Answer: c) list.sort(reverse=True)

1. How would you sort a list of tuples based on the second element of each tuple in Python?

a) sorted(list, key=lambda x: x[1])

b) list.sort(key=lambda x: x[1])

c) sorted(list, key=lambda x: x[1], reverse=True)

d) list.sort(key=lambda x: x[1], reverse=True)

Answer: a) sorted(list, key=lambda x: x[1])

1. How would you sort a list of dictionaries based on the value of the 'sale\_amount' key in each dictionary in Python?

a) sorted(list, key=lambda x: x['sale\_amount'])

b) list.sort(key=lambda x: x['sale\_amount'])

c) sorted(list, key=lambda x: x['sale\_amount'], reverse=True)

d) list.sort(key=lambda x: x['sale\_amount'], reverse=True)

Answer: a) sorted(list, key=lambda x: x['sale\_amount'])

1. What is the difference between the sort() method and the sorted() function in Python?

a) The sort() method modifies the original list, while sorted() creates a new sorted list.

b) The sort() method only works with lists, while sorted() works with any iterable.

c) The sort() method is faster than sorted().

d) The sort() method can sort lists with mixed data types, while sorted() cannot.

Answer: a) The sort() method modifies the original list, while sorted() creates a new sorted list.

1. Which of the following is the correct syntax to sort a list of strings in Python using the sorted() function?

a) sorted(list)

b) sorted(list, key=str)

c) sorted(list, reverse=True)

d) sorted(list, key=str, reverse=True)

Answer: a) sorted(list)