PYTHON WORKBOOK – SECTION 4

Data Structures in Python

Programmer's Hub – by CodeWithVivek
https://www.youtube.com/@code-with-vivek

4.1 Data Structures in Python

Python is powerful largely because of its **built-in data structures**.
These structures allow you to store, organize, and work with data efficiently.

In this section, we'll learn:

- Lists
- Tuples
- Dictionaries
- Sets
- List Comprehension

4.2 Lists

Quick Explanation

A **list** is a collection of ordered, changeable items.

Examples:

fruits = ["apple", "banana", "mango"]

numbers = [10, 20, 30, 40]

mixed = ["hello", 10, True]

Lists allow:

- Adding items
- Removing items
- Updating items
- Looping through items

🔧 **Useful List Methods**

append()

insert()

remove()

pop()

sort()

reverse()

Try This:

Create a list named movies having **5 favourite movies**.

Add a new movie using append().

Write code:

Debug This:

Why does this give an error?

fruits = ["apple", "banana", "cherry", "date", "fig", "grape"]

fruits[6] = "guava"

Your answer: _____

List Indexing & Slicing

Explanation

Indexing:

fruits[0] # first item

fruits[-1] # last item

Slicing:

Python list slicing provides a flexible way to access a range of elements using the syntax my_list[start:stop:step]. The start index is inclusive, and the stop index is exclusive.

fruits = ["apple", "banana", "cherry", "date", "fig", "grape", "guava"]

Indices: 0 1 2 3 4 5 6

#Neg Indices -7 -6 -5 -4 -3 -2 -1

Basic Slicing

Syntax	Code	Description	Result
[start:stop]	fruits[2:4]	Items from start (inclusive) to stop (exclusive)	["cherry", "date"]
[:]	fruits[:]	All items	["apple", ... , "guava]
[start:]	fruits[:3]	Items from start to the end of the list	["date", "fig", "grape", "guava"]
[:stop]	fruits[:3]	Items from the beginning up to stop (exclusive)	["apple", "banana", "cherry"]

Slicing with Negative Indices

[-N:]	fruits[-2:]	The last N items of the list	["grape", "guava"]
[:-N]	fruits[:-2]	All items except the last N items	["apple", "banana", "cherry", "date", "fig"]
[start:stop]	fruits[-4,-2]	Slice using negative indices, stop value exclusive	["date", "fig"]

Slicing with a Step

[start:stop:step]	fruits[1:5:2]	Items from start to stop with the given step	["banana", "date"]
[::step]	fruits[::2]	Every step-th item in the entire list	["apple", "cherry", "fig", guava"]
[::-1]	fruits[::-1]	Reverse the list	["guava", "grape", "fig", "date", "cherry", "banana", "apple"]

Modifying Lists with Slicing

1. **Modify Elements**
   fruits[1:3] = ["blueberry", "melon"]
   # Output: ["apple", "blueberry", "melon", "date", "fig", "grape", “guava”]

2. **Remove Elements**
   del fruits[1:4]
   # Output: ["apple", "fig", "grape", “guava”] (removes blueberry, melon, date)

3. **Insert Elements (by assigning to an empty slice)**
   fruits[1:1] = ["apricot", "avocado"]
   # Output: ["apple", "apricot", "avocado", "fig", "grape", “guava”]

Your Turn:

Given:

nums = [5, 10, 15, 20, 25]

Write the output for:

1. nums[0] = _____
2. nums[-1] = _____
3. nums[1:4] = _____

4.3 Tuples

Quick Explanation

A **tuple** is like a list, but *unchangeable* (immutable).

point = (10, 20)

days = ("Mon", "Tue", "Wed")

Tuples are used for fixed data: coordinates, settings, constants.

Try This:

Create a tuple of 3 colours:

colors = (_____)

Try modifying colors[0] — What happens?

_____

4.4 Dictionaries

Quick Explanation

Dictionaries store **key–value** pairs.

```
student = {
   "name": "Vivek",
   "age": 25,
   "marks": 88
}
```

Access values:

```
student["name"]
student.get("age")
```

Add values:

```
student["city"] = "Delhi"
```

Try This:

Create a dictionary for **your profile** with:

- name
- age
- city
- favourite language

```
me = {
   "name": _____,
   "age": _____,
   "city": _____,
   "language": _____
}
```

Debug This:

What's wrong?

```
student = {"name": "Amit"}

print(student[name])
```

Answer: _____

Dictionary Methods

Useful methods:

keys()

values()

items()

update()

pop()

Example:

```
student.update({"grade": "A"})
```

Your Turn:

Using your own dictionary from earlier, add a new key "hobby".

```
# Write code here:

```

_____

4.6 Sets

Quick Explanation

A **set** is an unordered collection of *unique* items.

numbers = {1, 2, 3, 3, 4}

result: {1,2,3,4}

Useful when you want to remove duplicates.

Useful Set Operations

union()

intersection()

difference()

add()

remove()

Try This:

Create two sets:

a = {1, 2, 3}

b = {3, 4, 5}

Write results for:

1. a.union(b) = _____
2. a.intersection(b) = _____
3. a.difference(b) = _____

Practice Problems

Problem 1

Write a program to find the **largest number** in a list.

Write code:

Problem 2

Given a sentence, count the **frequency of each word** using a dictionary.

Write code:

Problem 3

Remove duplicates from a list using a set.

Write code:

Section Summary

✓ Lists → ordered, changeable
✓ Tuples → ordered, unchangeable
✓ Dictionaries → key-value data
✓ Sets → unique item collection
✓ Each structure has its ideal purpose
✓ Methods help manipulate data quickly

Mini Assignment

Create a program that:

1. Creates a dictionary for **5 employees**
2. Each employee should have: name, age, salary
3. Print the **highest salary**
4. Print the **average age**
5. Convert all employee names to **uppercase**

Write your draft here: