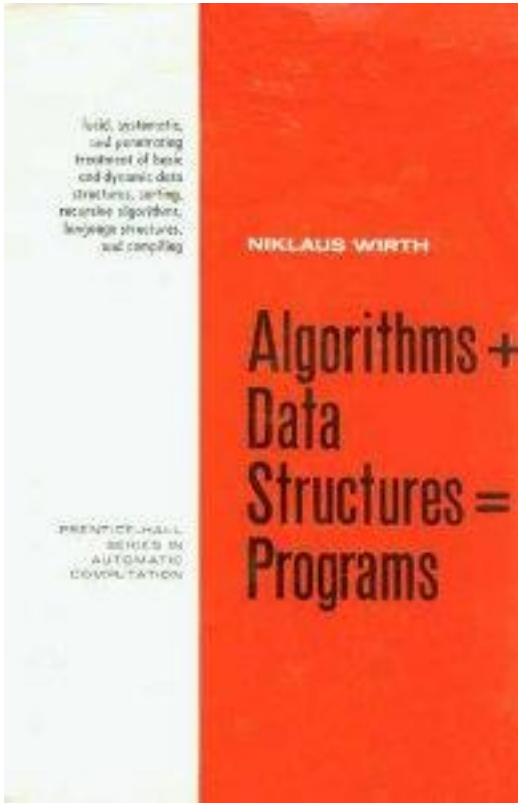
CDS2003: Data Structures and Object-Oriented Programming

Lecture: Tuple, String, Set, and Map/Dictionary

Review

- Array
  - Types of array
  - Basic operations of array
- Python List
  - Basic operations of Python List



Algorithms + *Data Structures* = *Programs*

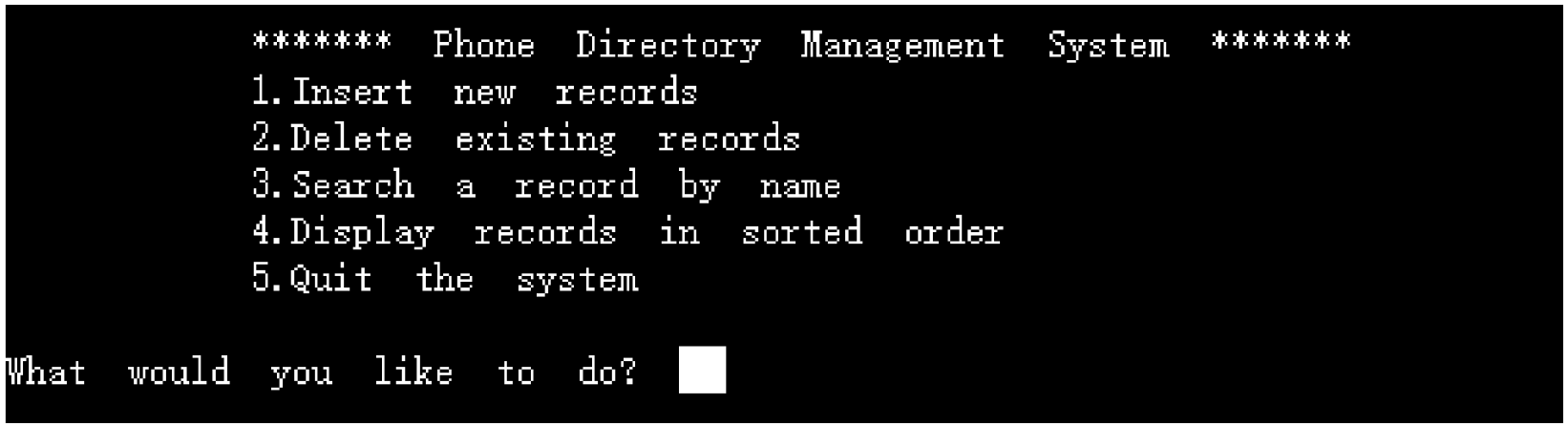
-- Niklaus Wirth

Group project

- Group project:
  - Each team can have up to 3 students, and individual projects are also welcome.
  - Hands-on experience
  - Problem-solving abilities
  - Language proficiency
  - Practical application of concepts
  - Self-confidence and motivation
- Suggested group project – Phone Directory Management System
  - If you choose the suggested project, each team has up to 2 students.

Description of the suggested project

- A phone directory is a listing of contacts, where each record mainly includes the name, phone number ,and address of a contact person. Records in the phone directory are well arranged so that each contact can be easily identified and found.
- In this project, a team of up to 2 students will be required to develop a Phone Directory Management System and analyze system implementation by using the knowledge you have learned in this course.
- A suggested menu of the system



Description of the suggested project

- Required modules of the Phone Directory Management System
  - Module 1: Inserting new records:

  This module can assist users to insert new contacts, which include name, telephone number, address and so on, in the phone directory
  - Module 2: Deleting existing records

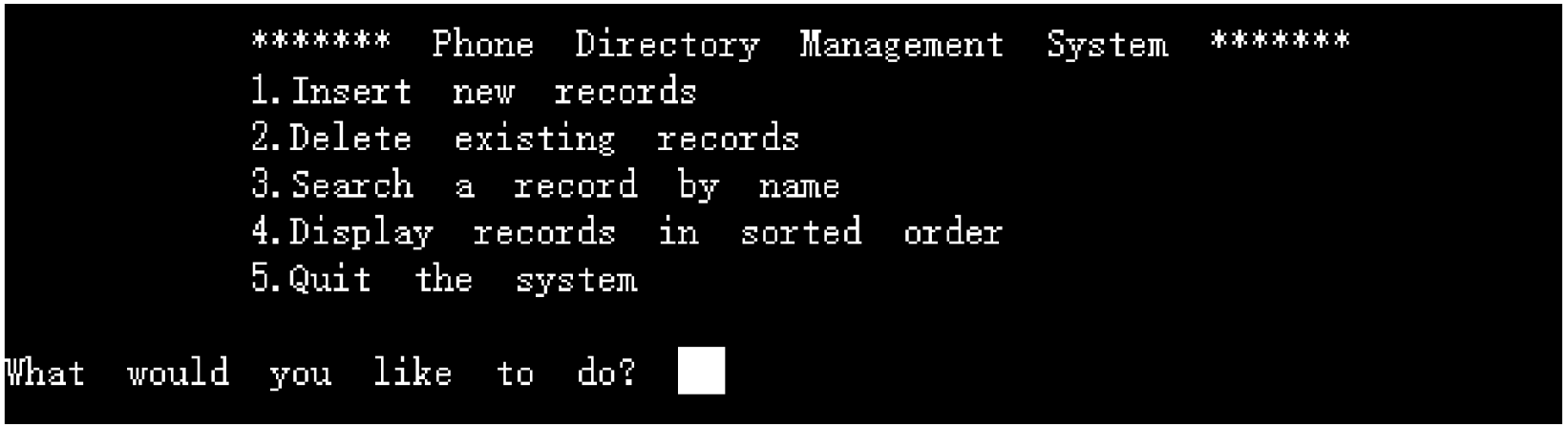
  This model can assist users to delete existing contacts in the phone directory
  - Module 3: Searching for a specific record by its name

  This module can assist users to delete existing contacts in the phone directory.
  - Module 4: Displaying the list of records in a sorted order

  This module can assist users to display all contact records in a sorted order in the phone directory.
  - Module 5: Quitting the system

Description of the suggested project

- When developing the application, you should consider making good use of the data structures and algorithms you have learned in this course.
- At least one data structure and one algorithm should be employed.
- Reasons for using the corresponding data structures and algorithms should be discussed in the written report.



Submission of the group project

- The submission will include two parts:
  1. A written report and source codes;
  2. A presentation slide and a presentation video
- Please submit the two parts in different submission entry in the MOODLE.
  - There will be one entry for the written report and source codes, and
  - The other entry for a presentation slide and a presentation video.
- Submission of a written report and source codes should be in a single compressed file like .zip or. Rar.
  - Please put all your source codes in a single folder and make sure that your source codes are executable.
- Submission of a presentation slide and a presentation video should be a compressed file including a PPT file and a presentation video file or a link to video (i.e., Youtube, Bilibili, or Google Driver) if the video is oversize.
- Each group should choose one student as the representative to submit the written materials onto Moodle.
- Each group should prepare a video presentation. The video should be about 10 minutes long. It should be no more than 15 minutes.
- Use ZOOM for the video recording.

Marking criteria (tentative)

- Problem solving and application of knowledge (35%)
  - Whether the problem is well addressed in the project
  - Whether the basic requirements are fulfilled
  - Whether knowledge is applied correctly
  - Whether the solution is suitable for the problem
- Analysis and critical thinking (40%)
  - Whether the solution is effective
  - Whether knowledge is applied wisely
  - Whether the problem is well-analyzed
  - Whether the time and complexity analysis are included in the report
  - Whether the knowledge applied is critically discussed in the report

Marking criteria (tentative)

- Object-oriented programming (OOP)
  - Whether the project is implemented correctly using the idea of OOP
- Presentation and illustration (15%)
  - Whether the problem is clearly defined in the written report
  - Whether the knowledge applied is well-illustrated
  - Whether the written report is organized elaborately
  - Whether the critical points in the project is well-revealed
  - Whether the report is written in a concise manner
- Implementation and organization (10%)
  - Whether the program code is well-organized
  - Whether the application is implemented elegantly and professionally

Guidelines for the video presentation

- Appropriate time allocation and pace:
  - Start the presentation punctually
  - Use appropriate pace
  - Manage time and content with smooth progression
- Clear, logically organized and relevant content
  - Include relevant information
  - Clearly state and develop the key points
  - Avoid ambiguities
- Make effective use of presentation tools
- Use good language, eye contact, and appropriate voice tone
- Gain/hold attention
- Clarity of speech

Tuple

- Tuple is a collection of objects separated by commas.
- Similar to a Python list but tuple is immutable/static (No add or delete)

```
var = (1, 2, 3)

print("Value in Var[-1] = ", var[-1])
print("Value in Var[-2] = ", var[-2])
print("Value in Var[-3] = ", var[-3])
```

Output:

```
Value in Var[-1] =  3
Value in Var[-2] =  2
Value in Var[-3] =  1
```

```
# tuple with different datatypes
tuple_obj = ("immutable",True,23)
print(tuple_obj)
```

Output :

```
('immutable', True, 23)
```

- Create a tuple with one element!

```
mytuple = ("Geeks",)
print(type(mytuple))

#NOT a tuple
mytuple = ("Geeks")
print(type(mytuple))
```

Output:

```
<class 'tuple'>
<class 'str'>
```

```
# Code to create a tuple with repetition
tuple3 = ('python',)*3
print(tuple3)
```

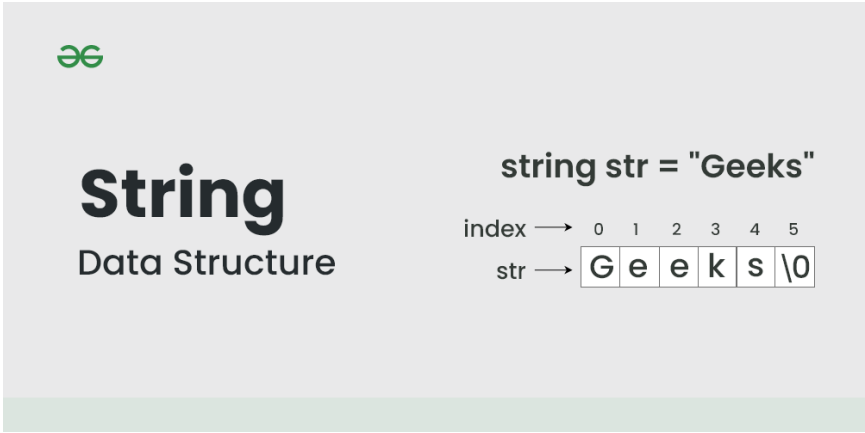
```
('python', 'python', 'python')
```

- Create a tuple of multiple same elements from a single element

[1] https://www.geeksforgeeks.org/tuples-in-python/?ref=header_outind

String

- Strings are commonly used for storing and manipulating textual data.
- A string is defined as an array of characters.
- The difference between a character array and a string is the string is terminated with a special character ‘\0’.
- In most programming languages, strings are treated as a data type.
- Programming does not have a character data type, a single character is simply a string with a length of 1.

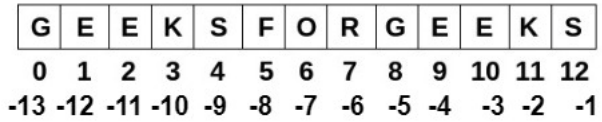


[1] https://www.geeksforgeeks.org/string-data-structure/

String operations

- Length: Determining the number of characters in a string.
- Access: Accessing individual characters in a string by index.

```
# Example string
my_string = "Hello, World!"

# Get the length of the string
length = len(my_string)
```




```
String1 = "Hello Kitty"
print("Initial String: ", String1)

# Printing a character with a given index
print("First character of String is: ", String1[3])
print("Last character of String is: ", String1[-3])
```

- Substring/Slicing: Extracting a portion of a string.

```
#Program to reverse a string
gfg = "Hello, world"
print(gfg[::-1])
```

- Concatenation: Combining two strings to create a new string.
- Comparison: Comparing two strings to check for equality or order.
- Search: Finding the position of a specific substring within a string.
- Modification: Changing or replacing characters within a string.
- …

[1] https://www.geeksforgeeks.org/python-string/

String concatenation

- Using + operation

```
# Defining strings
var1 = "Hello "
var2 = "Kitty"

# + Operator is used to combine strings
var3 = var1 + var2
print(var3)
```

- Using join() method

```
var1 = "Hello"
var2 = "Kitty"

# join() method is used to combine the strings
print("".join([var1, var2]))

# join() method is used here to combine
# the string with a separator Space(" ")
var3 = " ".join([var1, var2])

print(var3)
```

- Using % operator

```
var1 = "Hello"
var2 = "Kitty"

# % Operator is used here to combine the string
print("% s % s" % (var1, var2))
```

- Using format() function

```
var1 = "Hello"
var2 = "Kitty"

# format function is used here to
# combine the string
print("{} {}".format(var1, var2))

# store the result in another variable
var3 = "{} {}".format(var1, var2)

print(var3)
```

- Using comma “,”

```
var1 = "Hello"
var2 = "  "
var3 = "Kitty"

# using comma to combine data types
# with a single whitespace.
print(var1, var2, var3)
```

- Using f-string

```
name = "HelloKitty"
age = 50

# String concatenation using f-string
greeting = f"Hello, my name is {name} and I am {age} years old."

print(greeting)
```

[1] https://www.geeksforgeeks.org/python-string-concatenation/

Find position of a character in a given string

- Using rfind() method

```
string = 'Work hard! Play Hard!'
letter = 'k'
print(string.rfind(letter))
```

- Using index()

```
# Initializing string
ini_string = 'abcdef'

# Character to find
c = "b"
# printing initial string and character
print("initial_string : ", ini_string, "\ncharacter_to_find : ", c)

# Using Naive Method
res = None
for i in range(0, len(ini_string)):
    if ini_string[i] == c:
        res = i + 1
        break

if res == None:
    print("No such character available in string")
else:
    print("Character {} is present at {}".format(c, str(res)))
```

- Using re.search()

```
import re
string = 'Study for yourself'
pattern = 'for'
match=(re.search(pattern, string))

#getting the starting index using match.start()
print ("starting index", match.start())

#Getting the start and end index in tuple format using match.span()
print ("start and end index", match.span())
```

- Using the loop

```
# Initializing string
ini_string1 = 'xyze'

# Character to find
c = "b"
# printing initial string and character
print ("initial_strings : ", ini_string1,
            "\ncharacter_to_find : ", c)

# Using index Method
try:
    res = ini_string1.index(c)
    print ("Character {} in string {} is present at {}".format(
                                c, ini_string1, str(res + 1)))
except ValueError as e:
    print ("No such character available in string {}".format(ini_string1))
```

[1] https://www.geeksforgeeks.org/python-find-position-of-a-character-in-given-string/

String find() method

- Returning the lowest index or first occurrence of the substring if it is found in a given string

- With no start and end argument

```
word = 'geeks for geeks'

# returns first occurrence of Substring
result = word.find('geeks')
print("Substring 'geeks' found at index:", result)

result = word.find('for')
print("Substring 'for ' found at index:", result)

# How to use find()
if word.find('pawan') != -1:
    print("Contains given substring ")
else:
    print("Doesn't contains given substring")
```

Output

```
Substring 'geeks' found at index: 0
Substring 'for ' found at index: 6
Doesn't contains given substring
```

- With start and end argument

```
word = 'geeks for geeks'

# Substring is searched in 'eks for geeks'
print(word.find('ge', 2))

# Substring is searched in 'eks for geeks'
print(word.find('geeks ', 2))

# Substring is searched in 's for g'
print(word.find('g', 4, 10))

# Substring is searched in 's for g'
print(word.find('for ', 4, 11))
```

Output

```
10
-1
-1
6
```

[1] https://www.geeksforgeeks.org/python-string-find/?ref=header_outind

Updating or deleting a character

- Updating a character

```
# Python Program to Update
# character of a String

String1 = "Hello, I'm a Geek"
print("Initial String: ")
print(String1)

# Updating a character of the String
## As python strings are immutable, they don't support item updation directly
### there are following two ways
#1
list1 = list(String1)
list1[2] = 'p'
String2 = ''.join(list1)
print("\nUpdating character at 2nd Index: ")
print(String2)

#2
String3 = String1[0:2] + 'p' + String1[3:]
print(String3)
```

- Deleting a character

```
# Python Program to delete
# character of a String

String1 = "Hello, I'm a HelloKitty"
print("Initial String: ")
print(String1)

print("Deleting character at 2nd Index: ")
del String1[2]
print(String1)
```

[1] https://www.geeksforgeeks.org/python-string/

Set

- A type of data structure which stores a collection of distinct elements.
- No duplicate elements
- A set is unordered, so we cannot know access item using indexes as we do in an array or a list.

```
var = {"Geeks", "for", "Geeks"}
type(var)
```

```
# typecasting list to set
myset = set(["a", "b", "c"])
print(myset)

# Adding element to the set
myset.add("d")
print(myset)
```

```
# Python example demonstrate that a set
# can store heterogeneous elements
myset = {"Geeks", "for", 10, 52.7, True}
print(myset)
```

```
# Python program to demonstrate that
# a set cannot have duplicate values
# and we cannot change its items

# a set cannot have duplicate values
myset = {"Geeks", "for", "Geeks"}
print(myset)

# values of a set cannot be changed
myset[1] = "Hello"
print(myset)
```

[1] https://www.geeksforgeeks.org/introduction-to-set-data-structure/
[2] https://www.geeksforgeeks.org/sets-in-python/

Frozen set

- Frozen sets are immutable objects that only support methods and operators that produce a result without affecting the frozen set or sets to which they are applied.
- Function frozenset()
- From dynamic to static

```
# Python program to demonstrate differences
# between normal and frozen set

# Same as {"a", "b","c"}
normal_set = set(["a", "b","c"])

print("Normal Set")
print(normal_set)

# A frozen set
frozen_set = frozenset(["e", "f", "g"])

print("\nFrozen Set")
print(frozen_set)

# Uncommenting below line would cause error as
# we are trying to add element to a frozen set
# frozen_set.add("h")
```

[1] https://www.geeksforgeeks.org/introduction-to-set-data-structure/
[2] https://www.geeksforgeeks.org/sets-in-python/

Union and intersection operations

- Two sets can be merged using union() or | operator.
- Intersection operation can be done through intersection() or & operator

```
# Python Program to
# demonstrate union of
# two sets

people = {"Jay", "Idrish", "Archil"}
vampires = {"Karan", "Arjun"}
dracula = {"Deepanshu", "Raju"}

# Union using union()
# function
population = people.union(vampires)

print("Union using union() function")
print(population)

# Union using "|"
# operator
population = people|dracula

print("\nUnion using '|' operator")
print(population)
```

```
# Python program to
# demonstrate intersection
# of two sets

set1 = set()
set2 = set()

for i in range(5):
    set1.add(i)

for i in range(3,9):
    set2.add(i)

# Intersection using
# intersection() function
set3 = set1.intersection(set2)

print("Intersection using intersection() function")
print(set3)

# Intersection using
# "&" operator
set3 = set1 & set2

print("\nIntersection using '&' operator")
print(set3)
```

Output:

```
Union using union() function
{'Karan', 'Idrish', 'Jay', 'Arjun', 'Archil'}

Union using '|' operator
{'Deepanshu', 'Idrish', 'Jay', 'Raju', 'Archil'}
```

Output:

```
Intersection using intersection() function
{3, 4}

Intersection using '&' operator
{3, 4}
```

[1] https://www.geeksforgeeks.org/introduction-to-set-data-structure/
[2] https://www.geeksforgeeks.org/sets-in-python/

Finding differences of sets

- Find differences between sets using difference() or - operator

```python
# Python program to
# demonstrate difference
# of two sets

set1 = set()
set2 = set()

for i in range(5):
    set1.add(i)

for i in range(3,9):
    set2.add(i)

# Difference of two sets
# using difference() function
set3 = set1.difference(set2)

print(" Difference of two sets using difference() function")
print(set3)

# Difference of two sets
# using '-' operator
set3 = set1 - set2

print("\nDifference of two sets using '-' operator")
print(set3)
```

Output:

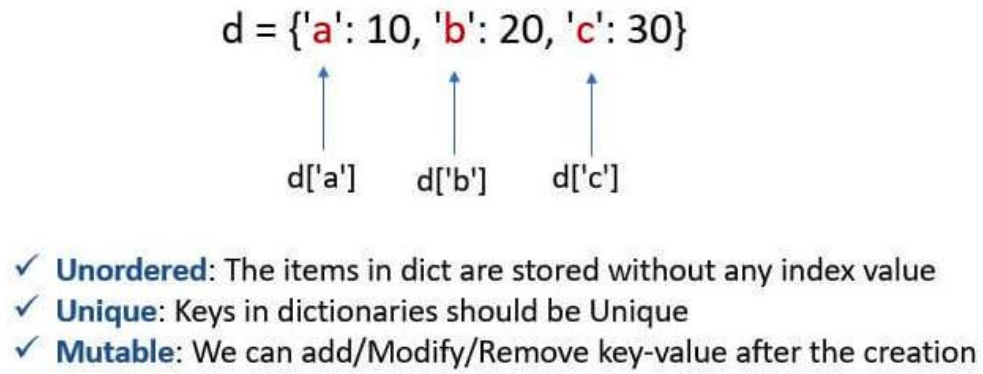
```
Difference of two sets using difference() function
{0, 1, 2}

Difference of two sets using '-' operator
{0, 1, 2}
```

```python
set1 = {1, 2, 3}
set2 = {3, 4, 5}

# Union
print(set1 | set2)  # Output: {1, 2, 3, 4, 5}

# Intersection
print(set1 & set2)  # Output: {3}

# Difference
print(set1 - set2)  # Output: {1, 2}

# Symmetric Difference
print(set1 ^ set2)  # Output: {1, 2, 4, 5}
```

[1] https://www.geeksforgeeks.org/introduction-to-set-data-structure/
[2] https://www.geeksforgeeks.org/sets-in-python/

Map

- Map data structure is also known as a dictionary, associative array, or hash map and stores a collection of key-value pairs.
- Each key is associated with a single value.
- A Python dictionary stores the value in key:value pairs
- Efficiently store and retrieve data based on a unique identifier (the key)



[1] https://www.geeksforgeeks.org/introduction-to-map-data-structure/
[2] https://pynative.com/python-dictionaries/

Operations of Map

- Different ways to create a Python dictionary

```
Dict = {}
print("Empty Dictionary: ")
print(Dict)

Dict = dict({1: 'Geeks', 2: 'For', 3: 'Geeks'})
print("\nDictionary with the use of dict(): ")
print(Dict)

Dict = dict([(1, 'Geeks'), (2, 'For')])
print("\nDictionary with each item as a pair: ")
print(Dict)
```

Output

```
Empty Dictionary:
{}

Dictionary with the use of dict():
{1: 'Geeks', 2: 'For', 3: 'Geeks'}

Dictionary with each item as a pair:
{1: 'Geeks', 2: 'For'}
```

```
# Creating a map
d = {'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}

# Adding a new key-value pair
d['key4'] = 'value4'

# Retrieving the value associated with a key
print(d['key2'])  # Output: value2

# Updating the value associated with a key
d['key2'] = 'new_value2'

# Removing a key-value pair
del d['key3']

# Iterating over the key-value pairs in the map
for key, value in d.items():
    print(key, value)
```

Output

```
Value for key 'banana': 200
Updated value for key 'banana': 250
Key-value pairs in the map:
apple: 100
banana: 250
```

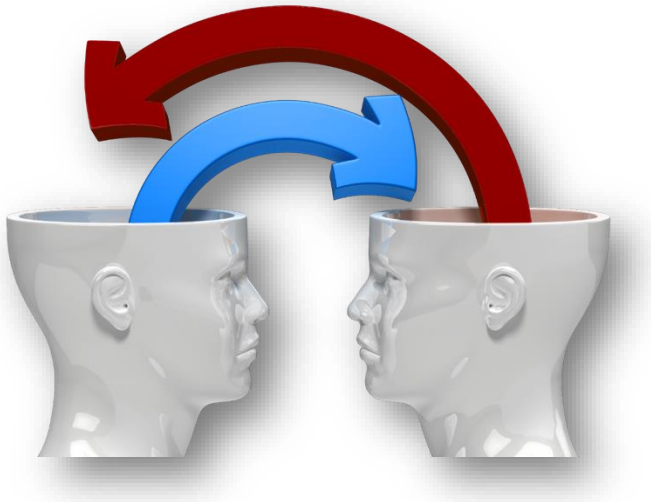
[1] https://www.geeksforgeeks.org/introduction-to-map-data-structure/
[2] https://www.geeksforgeeks.org/python-dictionary/?ref=header_outind

Difference between array, set, and map data structures

Features	Array	Set	Map/Dictionary
Duplicate values	Duplicate values	Unique values	Keys are unique, but the values can be duplicated
Order	Ordered collection	Unordered collection	Unordered collection
Retrieval	Elements in an array can be accessed using their index.	Iterate over the set to retrieve the value.	Elements can be retrieved using their key.
Operations	Adding, removing, and accessing elements	Set operations like union, intersection, and difference.	Maps are used for operations like adding, removing, and accessing key-value pairs.
Memory	Stored as contiguous blocks of memory	Implemented using linked lists or trees	Implemented using linked lists or trees

[1] https://www.geeksforgeeks.org/introduction-to-set-data-structure/

Discussion



Q & A!