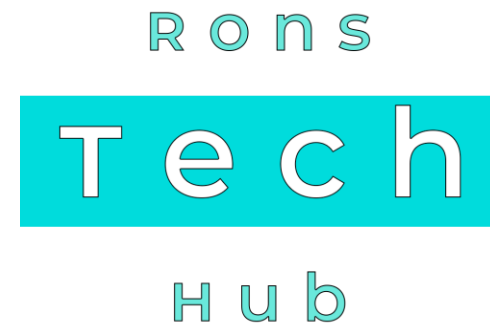


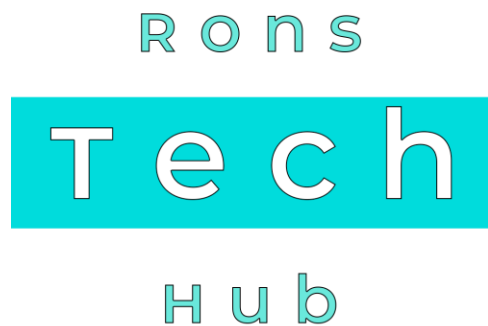
BTEC Level 3 Computing

Unit 1 - Principles of Computer Science

Data Structures



Selecting, applying, using and interpreting common data structures within a computer program to store and process data.



Data Structures

What are Data Structures?



Data structures are ways to organise and store data in a computer so it can be used efficiently.



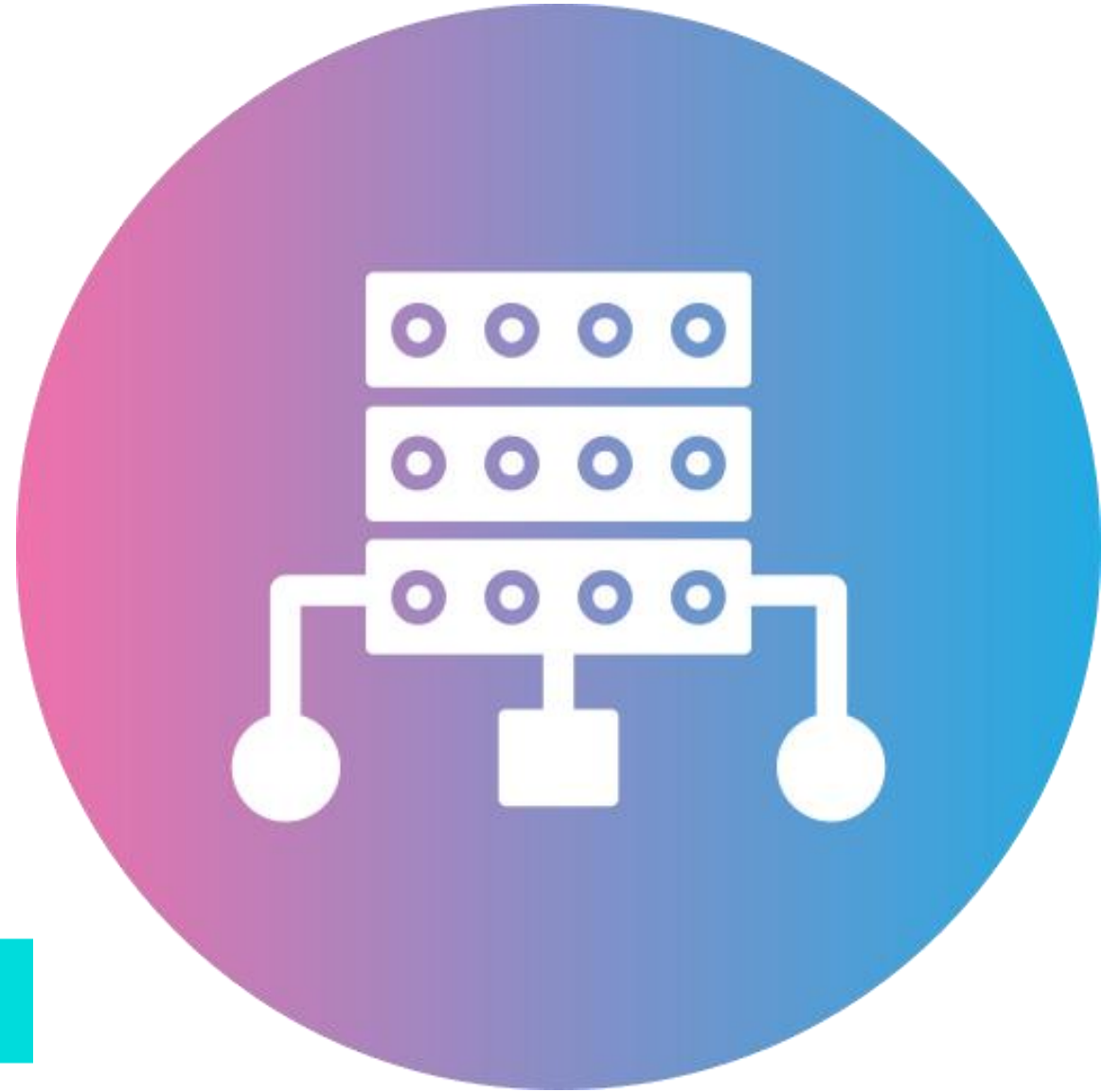
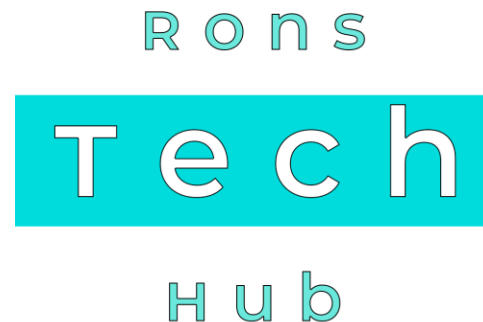
Think of them like containers: some are good for storing lists of things, others are better for looking up information quickly, and so on.



They help you manage your data effectively.

Types Of Data Structures

- Lists.
- Arrays.
 - Single Dimensional Array.
 - Multi-Dimensional Array
- Records.
- Sets.



Lists

- A list in Python is like a container that can hold a bunch of items, in a specific order.
- Think of it like a shopping list: you can put different things on it (numbers, words, even other lists!), and the order matters.
- You can easily add, remove, or change items on your list.
- Python lists:
<https://docs.python.org/3/tutorial/datastructures.html>



Python 3 List Example - 1D

- # creating the list
colors = ["red", "green", "blue", "yellow", "purple"]

add colour to the list
colors.append("turquoise")

remove colour
colors.remove("red")

R o n s

T e c h

H u b

Arrays

- In Python Lists and Arrays are more or less the same thing.
- Other languages like Java use the term Array or Array List.
- See the next page, that is why many newcomers prefer Python.



Arrays Java Example

- `import java.util.ArrayList;`
- `public class ColorList {`
- `public static void main(String[] args) {`
- `// Create an ArrayList to hold Strings (colors)`
- `ArrayList<String> colors = new ArrayList<>();`
- `// Add the colors to the list`
- `colors.add("red");`
- `colors.add("green");`
- `colors.add("blue");`
- `colors.add("yellow");`
- `colors.add("purple");`

Python 3 List Example - 2D

```
• colors_2d = [  
    ["red", "green", "blue"], # List of primary colors  
    ["yellow", "orange", "purple"], # List of secondary colors  
    ["black", "white", "gray"] # List of neutral colors  
]
```

Accessing elements:

```
print(colors_2d[0][1]) # Output: green (row 0, column 1)  
print(colors_2d[1][0]) # Output: yellow (row 1, column 0)  
print(colors_2d[2][2]) # Output: gray (row 2, column 2)
```

Records or Tuple

- Records group related data together, like a container for different types of information.
- They have named fields for easy access.
- Think of them like rows in a spreadsheet.
- Different languages have different names for them (struct, tuple, etc.), but the idea is the same.
- <https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences>

R O N S

T e c h

H u b



Records or Tuple Example

- # Using named tuples (good for simple, immutable records):
from collections import namedtuple

ColorRecord = namedtuple("ColorRecord", ["color1", "color2", "age"])

my_record = ColorRecord("red", "blue", 25)

print(my_record.color1) # Output: red
print(my_record.color2) # Output: blue
print(my_record.age) # Output: 25

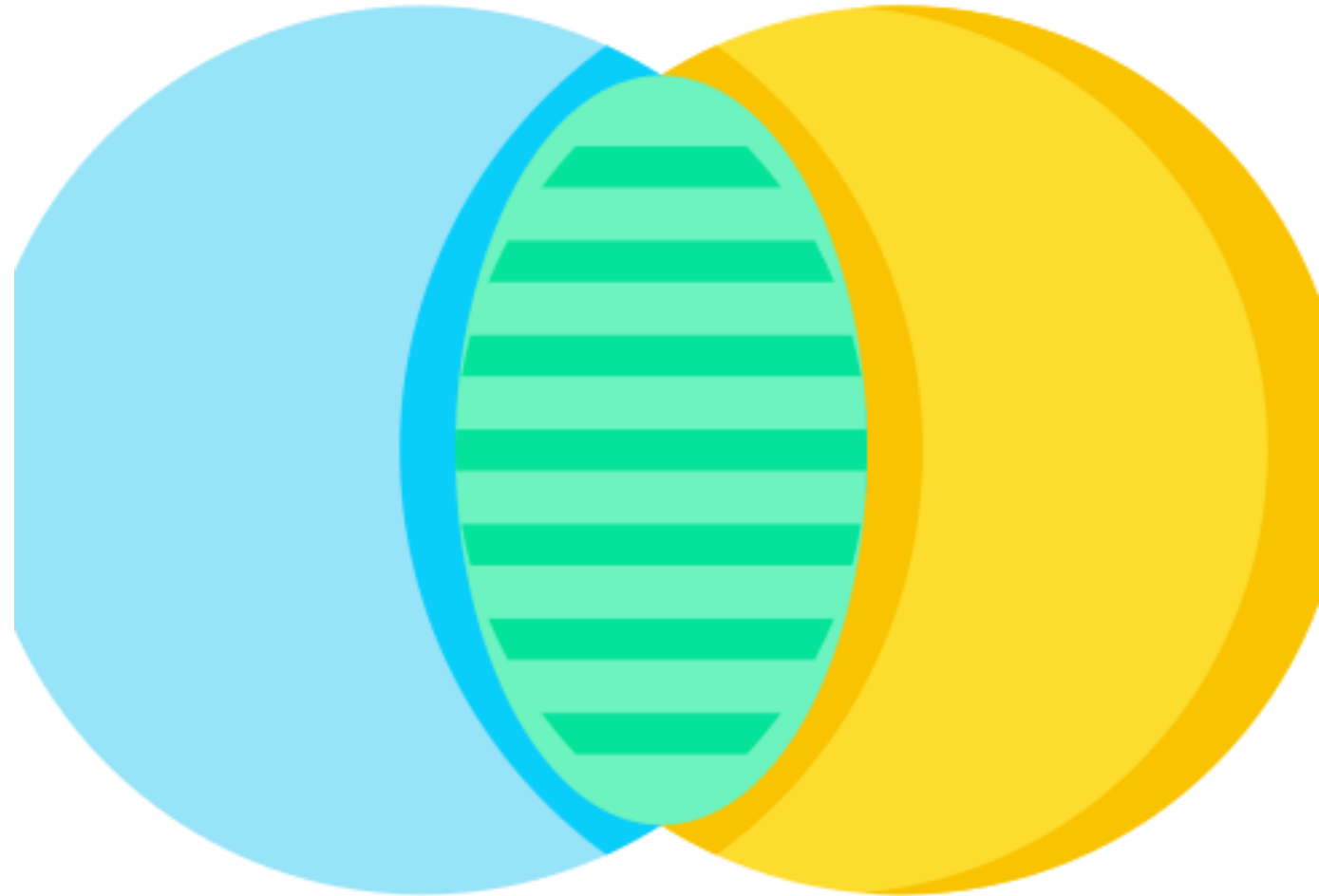
R o n s

T e c h

H u b

Sets

- A set is a data structure that stores a collection of unique elements.
- Think of it like a mathematical set: it can contain various items, but it doesn't allow duplicates, and the order of the items doesn't matter.
- <https://docs.python.org/3/tutorial/datastructures.html#sets>



Sets Example Python 3

- # Creating a set:
my_set = {1, 2, 3, 4, 5} # Using curly braces
another_set = set([3, 4, 5, 6, 7]) # Using the set() constructor with a list
- # Adding elements:
my_set.add(6) # Adds 6 to the set (if it's not already there)
my_set.add(2) # Adding a duplicate does nothing
print(my_set) # Output: {1, 2, 3, 4, 5, 6} (order may vary)
- # Removing elements:
my_set.remove(3) # Removes 3 from the set
print(my_set) # Output: {1, 2, 4, 5, 6}

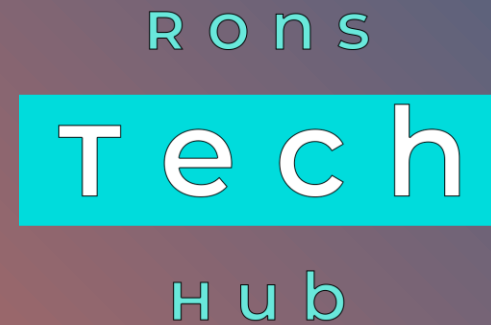
R o n s

T e c h

H u b



Next Time



Common/standard algorithms

