

# What is a record?

- A record is a data structure that groups together related items of data
- Instead of storing data in parallel arrays
- You can store more than one type of data together
- A record is an unordered data structure
- Can have multiple instances

## Parallel array

Player Array			Score Array	
Index	Data		Index	Data
0	Olivia	→	0	35
1	Luke	→	1	40
2	Adam	→	2	25
3	Alex	→	3	30



Creating a record structure

**recordStructure recordstructurename**

*Adding data to the record*

***fieldname : datatype***

***recordidentifier:recordstructurename***

**...  
endRecordStructure**

***recordidentifier.fieldname = data***

The pseudocode to define a new complex data structure called player:

**RECORD player**

**name: String**

**score: int**

**ENDRECORD**

The pseudocode to define a new complex data structure called Player:

**Player1: player**

**Player1.name = 'Olivia'**

**Player1.score = 35**



# Arrays of Records

Array: Players

0	1	2	3
Olivia 35	Luke 40	Adam 25	Alex 35

**Records** are treated as data types, so they can be held within a single **array**.

This allows for storage of more than one **record** within the same structure. This structure is essentially an **array of records**.

The table below simplifies the way that this data would be held in memory.

The records for the players could be stored in a 1D array.

It allows easy access/indexing/manipulation of each data item in turn

1D Array can hold multiple items of same data type – record

Maximum number of array elements is known

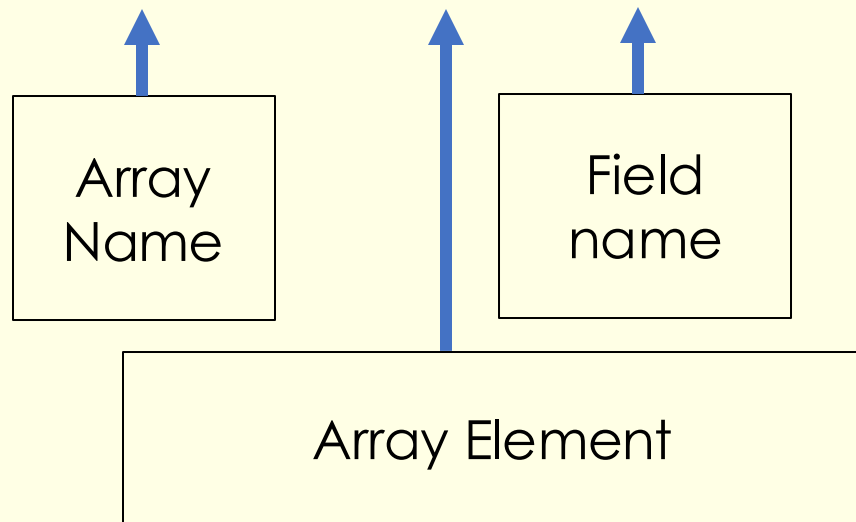


# Pseudocode of Arrays of records

Players(100) As player

We can then reference any element of the array:

Players(3).name = "Jane"



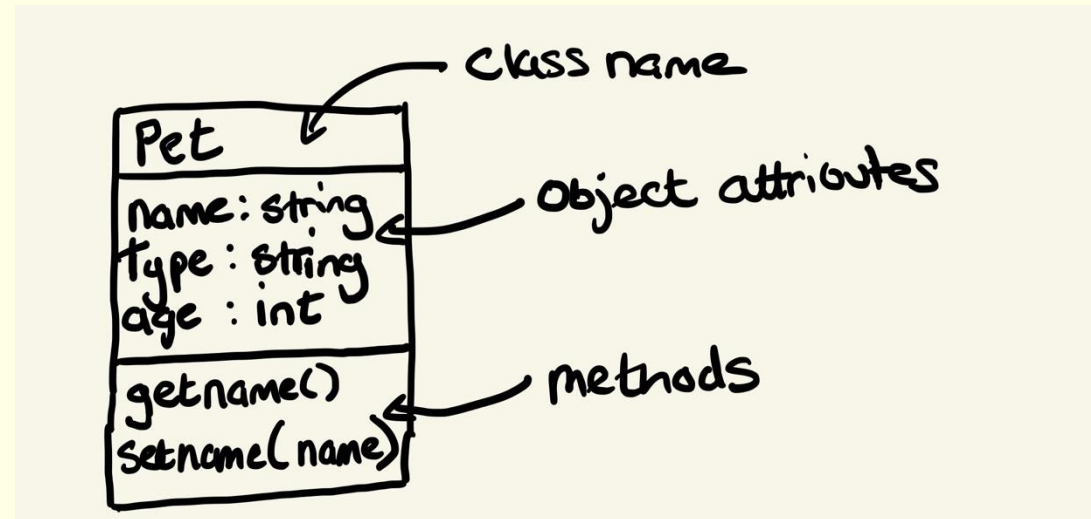
# You are writing a program to manipulate data about a pet that is used in a virtual pet game.

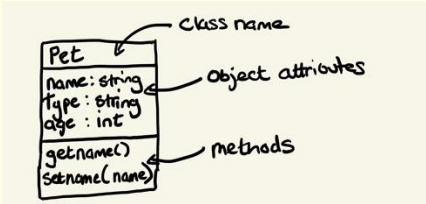
For each animal, the program needs to store:

- name (e.g. Buster)
- type (e.g. Dog)
- age (e.g. 7)

The **data** about the items can be stored using either a **record structure**, or as **objects of a class**.

```
recordStructure pets
  name : String
  type : String
  age : Int
endRecordStructure
```



	Record	Class
Similarities		
Data structure	A record is a data structure that stores data together, organised by attributes.	A <i>class</i> is a record with associated methods. Each object is a data structure with attributes stored together
Set up in advance	Attributes and structure for the record are set up. Meaning that it is created by the programmer for a particular purpose.	Constructor method defines the class object
Store data of different types	<pre>recordStructure <b>pets</b>     name : <b>String</b>     <b>type</b> : <b>String</b>     age : <b>Int</b> endRecordStructure</pre>	 A hand-drawn diagram of a class structure. It is a vertical rectangle divided into three sections. The top section is labeled 'Pet' and has an arrow pointing to it from the text 'Class name'. The middle section contains 'name: string', 'type: string', and 'age: int', with an arrow pointing to it from the text 'Object attributes'. The bottom section contains 'getname()' and 'setname(name)', with an arrow pointing to it from the text 'methods'.
Both can have multiple instances	Yes	Yes
Accessed by their names	Yes	Yes
Differences		
Class also has methods Class can include visibility of properties / private		



# PyRecord allows you to use **records** in python:

Task: Try this out this using pyrecord. In Thonny you will need to install the pyrecord package.

```
password.py × records.py ×  
1 from pyrecord import Record  
2 Person = Record.create_type("Person", "name", "email_address")  
3  
4 John = Person("John Smith", "jsmith@example.org")  
5 print(John.name)  
6 John.name = "Jon"  
7 print(John.name)  
8 |
```

