# Worksheet 1 Arrays, tuples and records

**Task 1**

1. Write a program to read 6 numbers into an array **numbers[0]** to **numbers[5**], ouput them in reverse order and then output the total and average.

numbers = []

total = 0

count = 0

for i in range(1, 7):

    numbers.append(i)

    total += i

    count += 1

print(numbers[::-1])

print(total)

print(total/count)

2. A teacher uses a program that stores pupil names in an array. The array is indexed from 0, so the first element in the array is **name[0]**. Occasionally the teacher needs to search for a name to find the student’s record number, which is **index + 1**.

Write a pseudocode algorithm that will search an array **name** containing **max** elements, to find a name and output record number if it exists. If the name does not exist the user should be told the term was not found. Use appropriate prompts for input and output in your solution.

pupil\_name = input()

name = []

index = 0

function searchForName(wanted\_name)

    for i in range(0, max) do

        if name[i] == wanted\_name then

            index = i

            return index + 1

            break

        endif

    endfor

endfunction

3. Sales quantities of a certain item, calculated to the nearest thousand, for Jan-March, April-June, July-Sep and Oct-Dec are held in separate arrays for each of 3 outlets. The sales figures for each quarter are to be totalled and output in the format

Total for quarter 1 xxxx

Total for quarter 2 xxxx

Total for quarter 3 xxxx

Total for quarter 4 xxxx

Write a pseudocode algorithm for this program. Initialise the array with the following test data:

Outlet1Sales = [10, 12,15,10]

Outlet2Sales = [5,8,3,6]

Outlet3Sales = [10,12,15,10]

**quarter1 = 0**

**quarter2 = 0**

**quarter3 = 0**

**quarter4 = 0**

**outlet1Sales = [10, 12, 15, 10]**

**outlet2Sales = [5, 8, 3, 6]**

**outlet3Sales = [10, 12, 15, 10]**

**salesCombined = []**

**salesCombined.extend([outlet1Sales, outlet2Sales, outlet3Sales])**

**for outlet in salesCombined do**

**quarter1 += outlet[0]**

**quarter2 += outlet[1]**

**quarter3 += outlet[2]**

**quarter4 += outlet[3]**

**print("Total for quarter 1", quarter1)**

**print("Total for quarter 2", quarter2)**

**print("Total for quarter 3", quarter3)**

**print("Total for quarter 4", quarter4)**

**A computer screen shot of a program

Description automatically generated**

**Task 2**

4. (a) Now suppose, in question 3, there were 50 outlets. Assuming the array outletSales[4,50] holds the sales values for each quarter, complete the following program to print the total sales figures for each quarter

Fill array outletSales with sales values

Initialise each element of array total[4] to zero

for quarter = 0 to 3

*total = [0 for i = 0 to 3]*

*for outlet = 0 to 49*

total[quarter] = total[quarter] + outletSales[quarter,outlet]

next *outlet*

*print(total[quarter])*

next quarter

A computer screen shot of a program code

Description automatically generated

5. A grid game draws a 6 by 4 grid with each square denoted by “x”. A character “O” can move by entering a row coordinate from 1 to 6 and column co-ordinate from 1 to 4. The character starts at array poistion [0,0] (Figure 1) and will move, for example, to row 0 column 1 (Figure 2) if the user enters 1, 2 for the row and column coordinates. **Remember that the indices of the array both start at 0.**

Write a pseudocode algorithm that creates a 2-D grid[row, column], drawn as shown in Figure 1.

Prompt the user to enter a row and column value. Update the character position and draw the new grid.

O x x x x O x x

x x x x x x x x

x x x x x x x x

x x x x x x x x

x x x x x x x x

x x x x x x x x

Figure 1 Figure 2

grid = [["x" for i in range(0, 4)] for x in range(0, 6)]

row = int(input("enter new row value"))

column = int(input("enter new column value"))

grid[row-1, column-1] = "O"

print(grid)

A screen shot of a computer code

Description automatically generated

6. A company runs a private car park near an airport. The car park has 10 rows numbered 1-10 and each row has spaces (referred to as columns) numbered 1-6 for 6 cars. Customers leave their cars with keys at the car park office, and a driver parks it in a free space and then records where it is parked.

The space is referenced by its grid coordinates row and column. E.g. a car parked in the 3rd row, 5th space would have the grid reference [3,5].

The driver enters the car registration into the computer. A car with registration AVH 61 HU parked at grid reference [3,5] would assign “AVH 61 HU” to **park[3,5]**. Empty spaces are denoted, for example, by **park[3,5] = “empty”**

Write pseudocode for a program which :

Initialises the grid, with each element holding “empty”.

“Parks a car”. This option asks the user to enter the registration number of a car and the grid reference (row and column number) where it has been parked.

Validates the user entry row between 1 and 10, column between 1 and 6 and asks user to re-enter until entry is valid.

Checks that this is an empty space, and if it is, puts the registration number in the appropriate element of the array. If it is not, displays “That space is taken” and asks the user to re-enter the grid reference.

Displays the grid.

park = [["empty" for i in range(0, 5)] for x in range(0, 10)]

optionChosen = int(input("choose which option you want: \n1. set spaces to 'empty' \n2. park car \n3. remove acar \n4. display park grid \n5. quitz \n"))

function setSpacesToEmpty(list)

    for row in list do

        for i in range(0, len(row)) do

            row[i] = "empty"

    return list

endfunction

function parkCar(grid)

    end = False

    while end == False do

        registration = input("enter car's registration number")

        gridRefCol = input("enter column the car is in")

        gridRefRow = input("enter row car is parked")

        gridSpot = grid[gridRefRow][gridRefCol]

        if gridSpot == "empty" then

            gridSpot = registration

            break

        else

            print("try again")

endprocedure

function removeCar(grid)

    registration = input("enter car's registration number")

    for row in grid do

        for i in range(0, len(row)) do

            if row[i] == registration then

                row[i] = "empty"

endprocedure

function displayGrid(grid)

    print(grid)

endprocedure

if optionChosen == 1 then

    setSpacesToEmpty(park)

elif optionChosen == 2

    parkCar(park)

elif optionChosen == 3

    removeCar(park)

elif optionChosen == 4

    displayGrid(park)

A screen shot of a computer program

Description automatically generated

**Task 3**

7. An application is plotting points in 3D space, using x, y, and z coordinates. Explain how the coordinates could be stored in each of the following and justify if the structure is a good choice for this task.

(a) 1D array

1d array is one dimensional hence the name, meaning it can only store separate values. Meaning you would only be able to store coordinates of a certain axis at a time, for example one 1d array can be used to store the x coordinates, and another can store the y coordinates and another the z coordinates. Not a good choice as it is really inefficient since you have to make 3 separate arrays each for a different axis’ coordinates. Then you have to match them with each other for the coordinate points.

(b) 2D array

2d array stores 1d arrays. So it can store an array of values in each position. This can be viable as in each position you can store a set of coordinates. For example, in index 0 you can store an array that houses a 1d array that houses the x y and z coordinate of one full coordinate. E.g [[1, 2, 3]], index 0 in the first array being the x, index 1 being y, and index 2 being z.

(c) 3D array

You could use a 3d array to store a lot of coordinate sets. For example in each 2d array inside the 3d array, you can use the 2d array as a set of coordinates for like a drawing for example, then if you want to have another object drawn you can use another position to store a 2d array which acts as another set of multiple coordinates for the object. Very viable if you want to make a lot of things in the 3d space at once.

(d) tuple

A tuple is used to store multiple data types of values. If you wanted to store a coordinate point that has a mixture of float values and int values, like x = 2 y = 1.4 z = 2.8 you could use a tuple since it is able to store different data types. Though if you wanted to add multiple coordinates you would need multiple tuples which can be somewhat inefficient.

(e) list of tuples

A list of tuples would be good because you can then store multiple full coordinates at the same time that also have different values, like integers and floats. A good choice since you can add multiple coordinates.

(f) 1D array of tuples

You can store coordinates just like in a list of tuples except it is static because its an array, so if you wanted to change a coordinate at some point in the program, you cant because its an array and not a list. Less viable than list of tuples.

(g) array of records

An array of records would store the x y and z coordinates on 3 separate fields which would be very organised and easy to tell which is the x y and z coordinate. Good choice because it is quite orgarnised though its an array so it cant be changed mid program which makes it somewhat inefficient.

(h) file of records

A file of records contains multiple records so you can store multiple sets of coordinates through multiple records which is very efficient since you can plot multiple points at once. Very good choice because it is like array of records but you can change the records if you want to fix a point’s coordinates.