

1.1P: Preparing for OOP – Answer Sheet

1. Explain the following terminal instructions:
 - a. cd: change directory, mean change to a folder path.
 - b. ls: list files, return a list of files
 - c. pwd: return the current location
2. Consider the following kinds of information, and suggest the most appropriate data type to store or represent each:

Information	Suggested Data Type
A person's name	string
A person's age in years	Int
A phone number	String
A temperature in Celsius	Float
The average age of a group of people	Float
Whether a person has eaten lunch	boolean

3. Aside from the examples already provided in question 2, come up with an example of information that could be stored as:

Data type	Suggested Information
String	Address
Integer	Quantity
Float	Length
Boolean	Whether you do homework or not.

4. Fill out the last two columns of the following table, evaluating the value of each expression and identifying the data type the value is most likely to be:

Expression	Given	Value	Data Type
6		6	int
True		true	boolean

a	a = 2.5	2.5	Float
1 + 2 * 3		7	Int
a and False	a = True	False	Boolean
a or False	a = True	True	Boolean
a + b	a = 1 b = 2	3	Int
2 * a	a = 3	6	Int
a * 2 + b	a = 2.5 b = 2	7.0	Float
a + 2 * b	a = 2.5 b = 2	6.5	Float
(a + b) * c	a = 1 b = 1 c = 5	10	Int
"Fred" + " Smith"		Fred Smith	String
a + " Smith"	a = "Wilma"	Wilma Smith	String

5. Using an example, explain the difference between **declaring** and **initialising** a variable.

For example, given a variable name "number". If we assign a variable, but do not give "number" a value, so it is declaring, and we can't leverage number at that time, we have to define the value to "number" later to use. If we assign a variable with a value, eg "number = 10", so it is initialising, and can be use at anytime.

The difference between the two is define the value right after assign variable or later.

6. Explain the term **parameter**. Write some code that demonstrates a simple of use of a parameter. You should show a procedure or function that uses a parameter, and how you would call that procedure or function.

A parameter is variables that is created to received data for a procedure or function.
<insert a screenshot of your code here>

```
1 def calculate_rectangle_area(length, width):
2     area = length * width
3     return area
4 result = calculate_rectangle_area(5, 8)
```

7. Using an example, describe the term **scope** as it is used in procedural programming (not in business or project management). Make sure you explain the different kinds of scope.

Scope is the state of variable that the region of a program where a variable can be accessed or modified, local and global.

```
1 using System;
2
3 public class ScopeExample
4 {
5     public static int globalVariable = 20;
6     static void Main()
7     {
8         AnotherMethod();
9     }
10    static void AnotherMethod()
11    {
12
13        int localVarAnotherMethod = 30;
14    }
```

8. In a procedural style, in any language you like, write a function called Average, which accepts an array of integers and returns the average of those integers. Do not use any libraries for calculating the average. You must demonstrate appropriate use of parameters, returning and assigning values, and use of a loop. Note — just write the function at this point, we'll use it in the next task. You shouldn't have a complete program or even code that outputs anything yet at the end of this question.

<insert a screenshot of your code here>

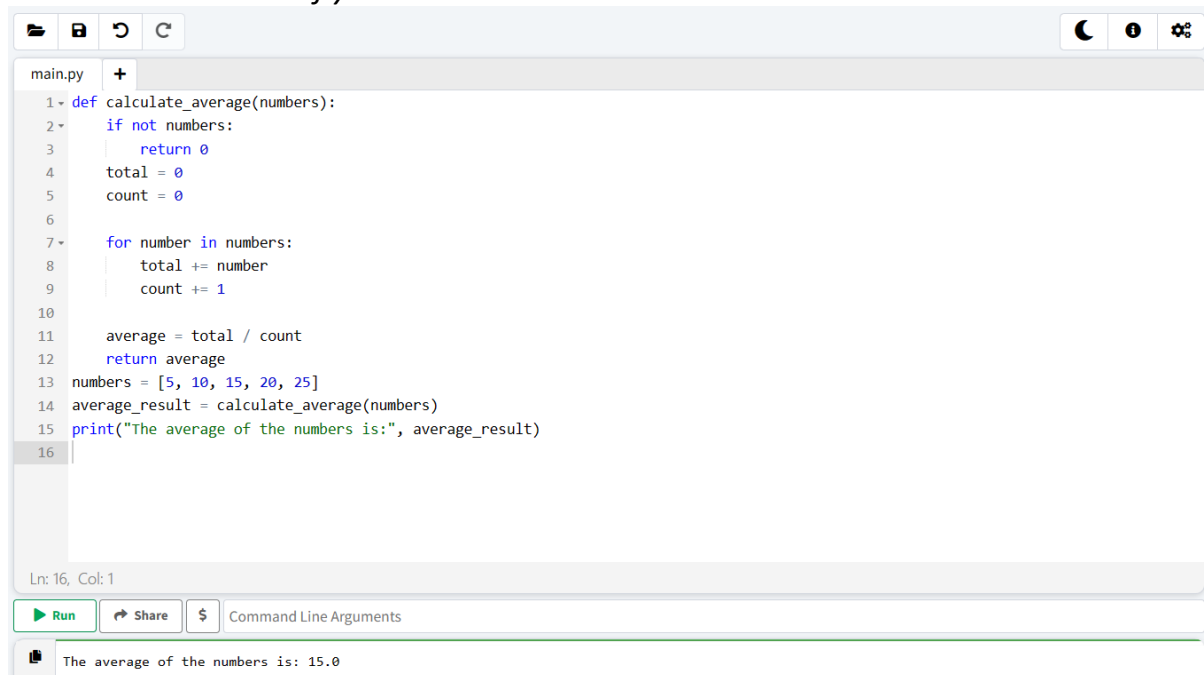
```
def calculate_average(numbers):
    if not numbers:
        return 0
    total = 0
    count = 0

    for number in numbers:
        total += number
        count += 1

    average = total / count
    return average
```

9. In the same language, write the code you would need to call that function and print out the result.

<insert a screenshot of your code here>



The screenshot shows a code editor window with a file named 'main.py'. The code defines a function 'calculate_average' and then calls it with a list of numbers. The output of the program is displayed at the bottom.

```
main.py
1 def calculate_average(numbers):
2     if not numbers:
3         return 0
4     total = 0
5     count = 0
6
7     for number in numbers:
8         total += number
9         count += 1
10
11     average = total / count
12     return average
13 numbers = [5, 10, 15, 20, 25]
14 average_result = calculate_average(numbers)
15 print("The average of the numbers is:", average_result)
16
```

Ln: 16, Col: 1

Run Share Command Line Arguments

The average of the numbers is: 15.0

10. To the code from 9, add code to print the message "Double digits" if the average is above or equal to 10. Otherwise, print the message "Single digits". Provide a screenshot of your program running.

<insert a screenshot of your code here>

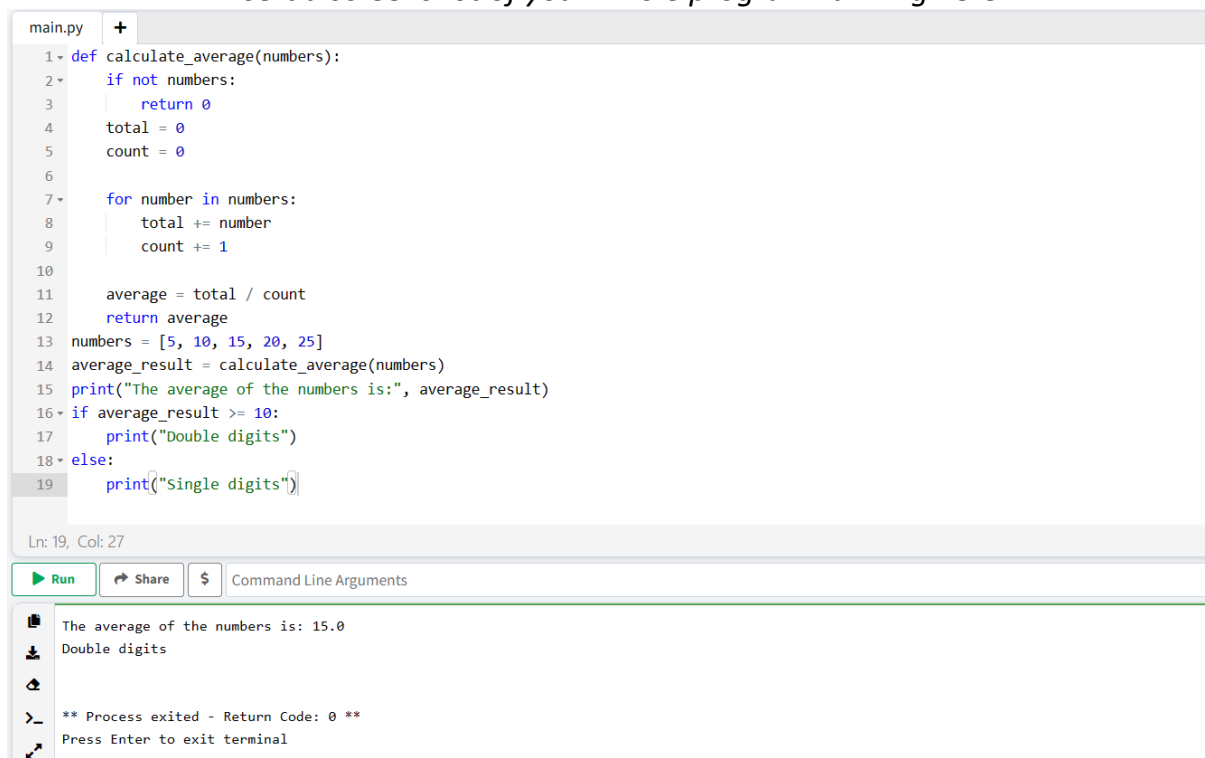
```
def calculate_average(numbers):
    if not numbers:
        return 0
    total = 0
    count = 0

    for number in numbers:
        total += number
        count += 1

    average = total / count
    return average

numbers = [5, 10, 15, 20, 25]
average_result = calculate_average(numbers)
print("The average of the numbers is:", average_result)
if average_result >= 10:
    print("Double digits")
else:
    print("Single digits")
```

<insert a screenshot of your whole program running here>



The screenshot shows a code editor with a file named 'main.py'. The code is a Python function 'calculate_average' that takes a list of numbers and returns their average. It then uses this function to calculate the average of the list [5, 10, 15, 20, 25] and prints the result. The code is as follows:

```
1 def calculate_average(numbers):
2     if not numbers:
3         return 0
4     total = 0
5     count = 0
6
7     for number in numbers:
8         total += number
9         count += 1
10
11     average = total / count
12     return average
13
14 numbers = [5, 10, 15, 20, 25]
15 average_result = calculate_average(numbers)
16 print("The average of the numbers is:", average_result)
17 if average_result >= 10:
18     print("Double digits")
19 else:
20     print("Single digits")
```

The editor shows the cursor at line 19, column 27. Below the code editor, there is a 'Run' button, a 'Share' button, and a 'Command Line Arguments' field. The output of the program is displayed in a terminal window at the bottom:

```
The average of the numbers is: 15.0
Double digits
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```