1.1P: Preparing for OOP – Answer Sheet

1. Explain the following terminal instructions:
    a. cd: Change Directory. Used to navigate between directories.
    b. ls: List Files and Directories. Displays the contents of the current directory.
    c. pwd: Print Working Directory. Shows the full path of the current directory.

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	Integer
A phone number	Integer or String
A temperature in Celsius	Float
The average age of a group of people	Integer
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	Student name
Integer	Number of students
Float	Height of student (m)
Boolean	Whether male students more than female students

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	Integer
`True`		True	Boolean
`a`	`a = 2.5`	2.5	Float
`1 + 2 * 3`		7	Integer
`a and False`	`a = True`	True	Boolean
`a or False`	`a = True`	True	Boolean
`a + b`	`a = 1` `b = 2`	3	Integer
`2 * a`	`a = 3`	6	Integer
`a * 2 + b`	`a = 2.5` `b = 2`	7	Integer
`a + 2 * b`	`a = 2.5` `b = 2`	6.5	Float

`(a + b) * c`	`a = 1`
	`b = 1`
	`c = 5`
		10	Integer

`"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.

The difference between the two is that declaring involves defining the data type for a variable. On the other hand, initialising involves assigning an initial value to the variable, which must be compatible with the declared data type.

- ***Declaring a Variable:***

```
int myNumber;
```

- ***Initializing a Variable:***

```
int myNumber = 42;
```

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 a variable used to pass information to a method, and it can be of any type. Arguments are passed to the parameters when a method is called, and there are different method parameters.

```
using System;

class Program
{
    // Define a method that takes a parameter
    static void Greet(string name)
    {
        Console.WriteLine($"Hello, {name}!");
    }

    static void Main()
    {
        // Call the method with an argument (value for the parameter)
        Greet("Alice");  // Output: Hello, Alice!
        Greet("Bob");    // Output: Hello, Bob!
    }
}
```

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 area within a program where variables can be accessed or referred to.
Local scope limits a variable's visibility to a specific part of the code, like a function or block, and it can't be accessed from outside that specific area.
Global scope where variables are declared outside of any specific function or code block. Variables with global scope can be accessed and used from any part of the program, including within functions or code blocks.

Example for local scope:

```
def my_function():
x = 10 # x is in local scope and can only be used within
my_function
print(x) # This would result in an error because x is not
defined in the global scope
my_function()
```

Example for global scope:

```
y = 20 # y is in global scope and can be accessed from
anywhere in the program
def another_function():
print(y) # y can be used inside this function because
it's in global scope
another_function()
print(y) # y can also be used here
```

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.

```python
def Average(arr):
    sum = 0
    for i in arr:
        sum += i
    return sum / len(arr)
```

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

```python
# Define the Average function (as given previously)
def Average(arr):
    sum = 0
    for i in arr:
        sum += i
    return sum / len(arr)

# Create an example array of integers
numbers = [5, 10, 15, 20, 25]

# Call the Average function and store the result in a variable
average_result = Average(numbers)

# Print out the result
print("Average:", average_result)
```

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.

```python
# Define the Average function (as given previously)
def Average(arr):
    sum = 0
    for i in arr:
        sum += i
    return sum / len(arr)

# Create an example array of integers
numbers = [5, 10, 15, 20, 25]

# Call the Average function and store the result in a variable
average_result = Average(numbers)

# Check if the average is greater than or equal to 10 and print the appropriate message
if average_result >= 10:
    print("Double digits")
else:
    print("Single digits")

# Print out the result
print("Average:", average_result)
```

```
vufanity@Vus-MacBook-Pro ~ % /usr/local/bin/python3 /Users/vufanity/COS20007/1.1.py
Double digits
Average: 15.0
vufanity@Vus-MacBook-Pro ~ %
```