

# 1.1P: Preparing for OOP – Answer Sheet

## Introduction

This answer sheet serves two purposes:

- A. It serves as a revision for you of your previous learnings; and
- B. It establishes a baseline understanding of your knowledge in key Computer Science topics.

As such, this answer sheet is divided into the following areas of knowledge:

- A. Your experience with UNIX/DOS console commands;
- B. Your ability to differentiate between data types (e.g., text) and information categories (e.g., title);
- C. Your experience with parsing and evaluating expressions according to rules of precedence;
- D. Your understanding of computer science concepts and various programming language constructs;
- E. Finally we want you to develop a simple function called *Average*. You will develop a fully functional program in three steps:
  - 1) Implement the *Average* function,
  - 2) Define a main function calling *Average*, and
  - 3) Define tests and output the result on calling *Average* on a given array of numbers.

## Section A: Console commands

Explain the following terminal instructions:

- a) ***cd***:

Allow the modification of the current working directory in a command-line interface, facilitating navigation between various directories within the file system.

- b) ***pwd***:

Allow the presentation of the current working directory in a command-line interface.

c) ***mkdir***:

Allow creating a new directory, aiding in the arrangement of files and folders in the file system.

d) ***cat***:

Concatenating and displaying the contents of files. It allows users to read, create, and combine files, presenting the output in the standard output, typically the terminal.

e) ***ls***:

Allow to list files and directories.

## Section B: Data types and Information Classes

1. Consider the following kinds of information, and suggest the most appropriate data type to store or represent each class of information:

Information Category	Suggested Data Type
A person's family name	string
A person's age in years	integer
A person's weight in kilograms	float
A telephone number	string

A temperature on the Kelvin scale	float
The average age of a group of children	integer
Whether a student has passed this task	boolean

2. Aside from the examples already provided in question 1, consider the following list and find an example of information that could be stored as:

Data type	Suggested Information Category
String	A person's email address
Integer	Number of pages in a book
Float	Price of the products
Boolean	If someone has sign-in the form or not

## Section C: Parsing and Evaluating Expressions

Fill out the **last** two columns of the following table. Parse and evaluate each expression. Enter the resulting value in column 3 and specify, in column 4, the data type of the expression in a compiler "*friendly*" form (i.e., in a language format accepted, for example, in C, C#, or Pascal):

Expression	Given	Value	Data Type
6		6	integer

<b>True</b>		True	Boolean
A	a = 2.5	2.5	Float
1 + 2 * 3		7	Integer
<b>a and False</b>	a = True	False	Boolean
<b>a or False</b>	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 = 3	8	Float
a + 2 * b	a = 2.5 b = 3	8.5	Float
(a + b) * c	a = 1 b = 2 c = 3	9	Integer
"Fred" + " Flintstone"		Fred Flintstone	String
a + " Rubble"	a = "Barney"	Barney Rubble	String

## Section D: Computer Science and Programming Language Concepts:

1. Using an example, explain the difference between **declaring** and **initializing** a variable.

The difference between the two is when a variable is declared, it establishes a named memory location with a specific data type, whereas initializing a variable involves assigning an initial value to that storage location.

Insert your example here:

```
full_name: (declare a variable)

full_name = "Tran Duc Thuan" (declare + initialize variable)
```

2. Explain the concept *parameter*. Write some code that demonstrates a simple 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 value that you can pass into the function or procedure when you call it, which allows you to provide inputs to a function so that it can perform its task based on those inputs.

Insert your example here:

```
def name(first_name, middle_name, last_name):
    print("My full name is: " + last_name + middle_name + first_name)

name("Thuan", "Duc", "Tran")
```

3. Using code examples, describe the term *scope* as it is being used in programming (not in business or project management). In your explanation, focus on procedural programming. Ensure that you cover as many kinds of scopes and their respective differences as possible. Your answer must detail at least two kinds of scopes.

A scope is the definition of how variables become visible and accessible in different sections of the code, outlining the conditions and locations where a variable can be used and modified.

There are two kind of scope base on my understanding: global scope and local scope.

Insert your examples here:

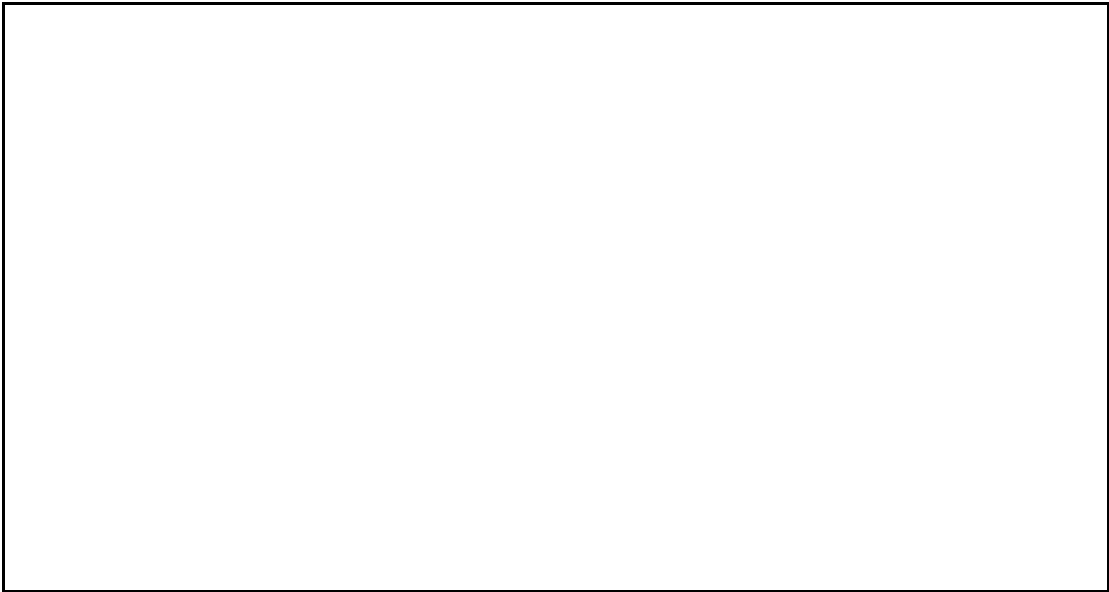
Local scope: refers to variables defined inside a method (function) or a block. They are not visible outside of that method or block and are only available within it.

Global scope: refers to the visibility and availability of a variable within a designated function or code block. A variable declared in a local scope is limited to being accessed and utilized solely within the boundaries of that specific function or block.

Example:

```
name = "Tran" # global scope (availabe inside and outside the function)
def display_name():
    name = "Thuan" # local scope (availabe only inside the funciton)
    print(name)

display_name()
print(name)
```



## Section E: Programming Practice:

1. Using procedural style programming, 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: we want to see your understanding of algorithms. You must demonstrate appropriate use of parameters, returning and assigning values, and the use of control statements like a loop. Note — just write the function *Average* at this point. In the next question we will ask you to *invoke the Average function*. You are not required to develop a complete program or even specify code that outputs anything at this stage. *Average* is a pure function. Input/output and any business logic processing is the responsibility of the (main line) calling the function *Average*.

Insert your code here:

```
def Average(num):
```

```
    if not num:
```

```
        return 0
```

```
    total = 0
```

```
    count = 0
```

```
    for i in num:
```

```
        total += i
```

```
        count += 1
```

```
average = total / count
```

2. Using the same language, write a main function you would to set up data (i.e., declare an array and initialize it with proper values), call the Average function, and print out the result. You are not required to provide input processing logic; you can have an inline instantiate of the collection of data values that the Average function needs to calculate the average of. Please use a reasonable data set, that is, the array must contain at least five elements of numerical type.

Insert your code here:

```
def main():

    data = [12, 14, 16, 18, 20]

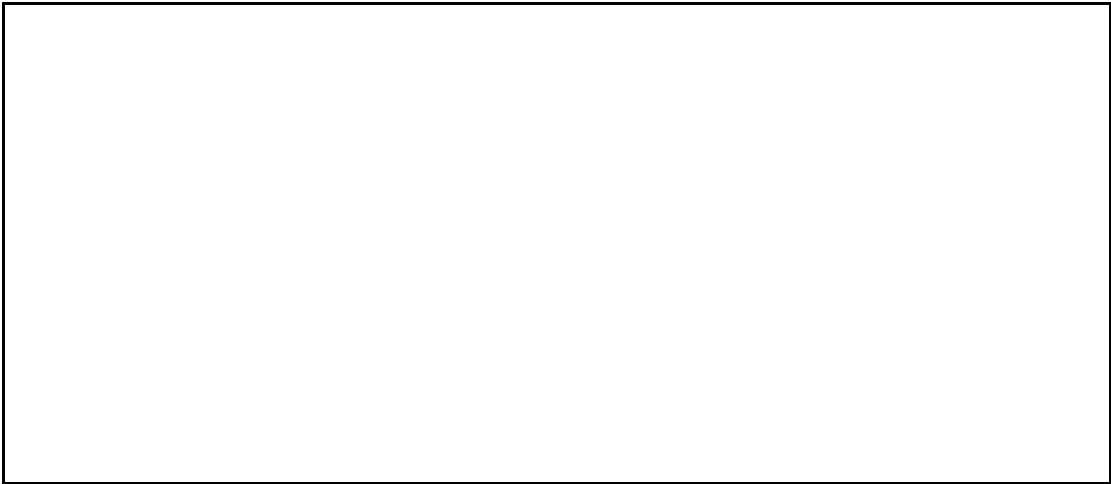
    result = Average(data)

    if result is not None:

        print(f"The average of the numbers is: {result}") # formatting string

main ()
```





3. Again using the same language, extend the main function with some output statements. Print the message “Double digits” if the average is above or equal to 10. Otherwise, print the message “Single digits”. And then, if the average is negative (e.g., the average of a week’s temperature readings at the Australian base in the Antarctic), add an additional line of output highlighting that the “Average value is in the negative”. Provide screenshot(s) of your program running, that is, the code and the run time output.

Insert your code here:

```
def main():

    data = [12, 14, 16, 18, 20]

    result = Average(data)

    if result is not None:
        print(f"The average of the numbers is: {result}") # formatting string

    if result >= 10:
        print("Double digits")
    else:
        print("Single digits")
```

```
    if result < 0:
        print("Average value is in the negative.")
    else:
        print("The array is empty, cannot calculate the average.")

main()
```

Insert your whole program here:

```
11(2,3).py x
Users > thuanduc > Documents > thuan's folder > work > COS20007 > week1 > 1.1 > 11(2,3).py > main
1 def Average(num):
2     if not num:
3
4         return None
5
6     total = 0
7     count = 0
8
9     for i in num:
10         total += i
11         count += 1
12
13     average = total / count
14     return average
15
16 def main():
17
18     data = [12, 14, 16, 18, 20]
19
20
21     result = Average(data)
22
23
24     if result is not None:
25         print(f"The average of the numbers is: {result}") # formatting string
26
27
28         if result >= 10:
29             print("Double digits")
30         else:
31             print("Single digits")
32
Ln 36, Col 10 Spaces: 4 UTF-8 LF Python 3.12.0 64-bit Go Live
```

```
4-Code.log x
Users > thuanduc > Library > Application Support > Code > logs > 20240301T185437 > window4 > exhost > output_logging_20240301T195231 > 4-Code.log
1 [Running] python -u "/Users/thuanduc/Documents/thuan's folder/work/COS20007/week 1/1.1/1.1(2,3).py"
2 The average of the numbers is: 16.0
3 Double digits
4
5 [Done] exited with code=0 in 0.02 seconds
6
7 [Running] python -u "/Users/thuanduc/Documents/thuan's folder/work/COS20007/week 1/1.1/1.1(2,3).py"
8 The average of the numbers is: 16.0
9 Double digits
10
11 [Done] exited with code=0 in 0.017 seconds
12
13
```

The image shows a Visual Studio Code editor window with a Python file named `1.1(2,3).py`. The code defines an `Average` class and a `main` function. The `main` function calculates the average of the list `[12, 14, 16, 18, 20]` and prints the result. The output console shows the execution of the script, displaying the average value and the number of digits.

```
1 1.1(2,3).py
2
3
4     return None
5
6     total = 0
7     count = 0
8
9     for i in num:
10         total += i
11         count += 1
12
13     average = total / count
14     return average
15
16 def main():
17
18     data = [12, 14, 16, 18, 20]
19
20
21     result = Average(data)
22
23
24     if result is not None:
25         print(f"The average of the numbers is: {result}") # formatting string
26
27
28         if result >= 10:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

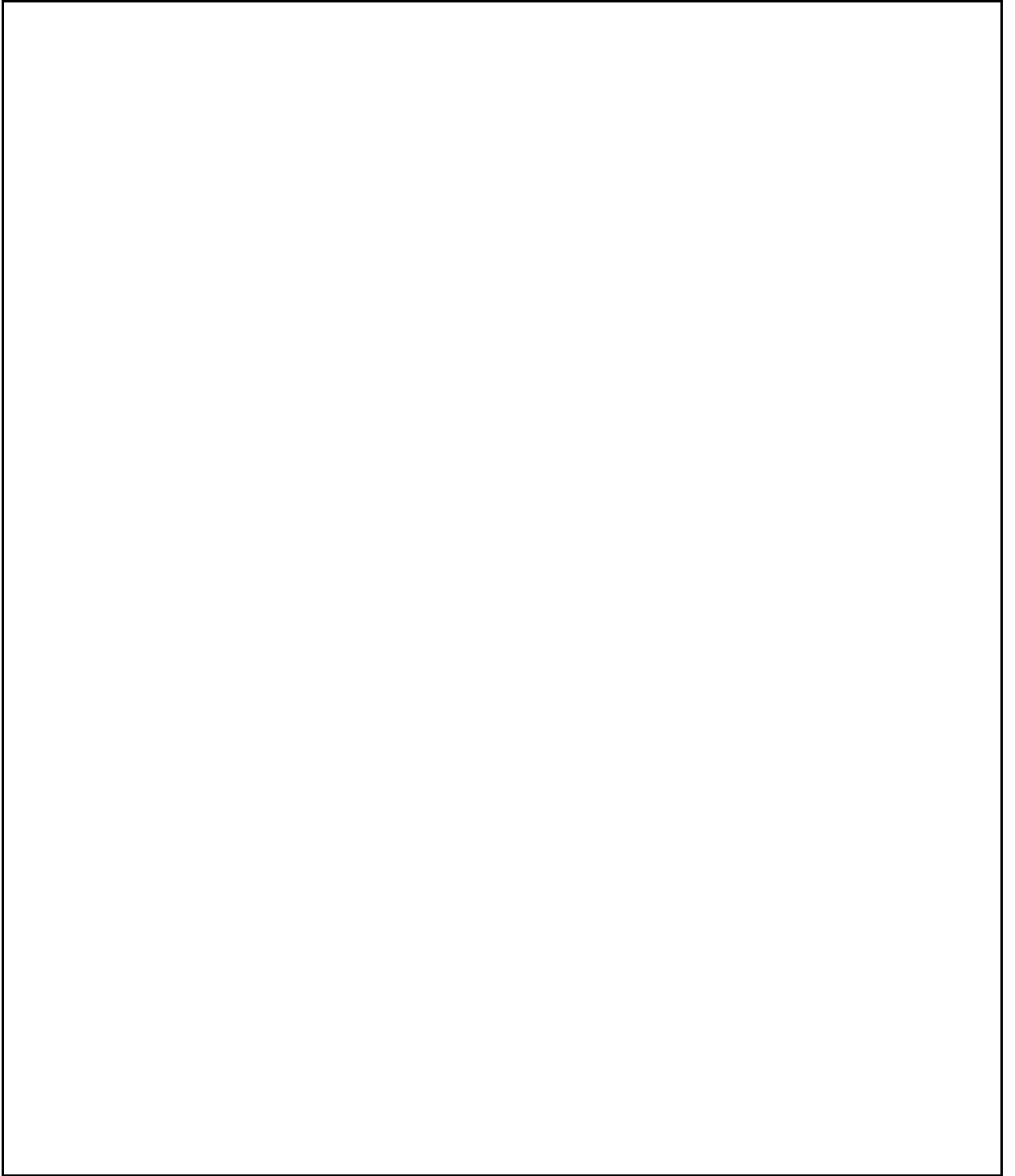
[Running] python -u "/Users/thuanduc/Documents/thuan's folder/work/COS20007/week 1/1.1(2,3).py"

The average of the numbers is: 16.0

Double digits

[Done] exited with code=0 in 0.065 seconds

Ln 27, Col 9 Spaces: 4 UTF-8 LF Python 3.12.0 64-bit Go Live Prettier



## End of Task

All students have access to the Adobe Acrobat tools. Please print your solution to PDF and submit via Canvas.