1.1P: Preparing for OOP – Answer Sheet

1. Explain the following terminal instructions:
    a. cd: It is a command that is used to change the current working directory.
    b. ls: A command that lists computer files and directories in the operating system.
    c. pwd: It writes full pathname of existing directory to the standard output.

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
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	Any kind of name
Integer	Number of students in a class
Float	Average marks of students in four units.
Boolean	Students passed or fail

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`	False	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	Float
`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"`	Wilam smith	string

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

The difference between the two is storing a value. Declaring a variable reserves a name in the memory but does not give a value. While in initialisation, a value is given after a variable is declared.

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 value that is used in a function when it gets called. This is to pass a value to the function so that the value can be used whenever needed.

main.rb

```ruby
def welcome(name)

    puts "Hello, #{name}!!"
end

welcome("Nur E Siam")

```

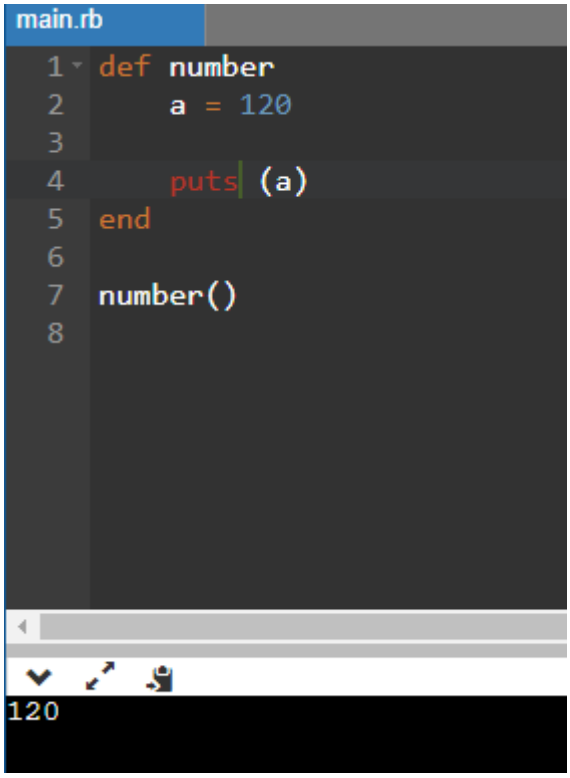
```
Hello, Nur E Siam!!
```

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.

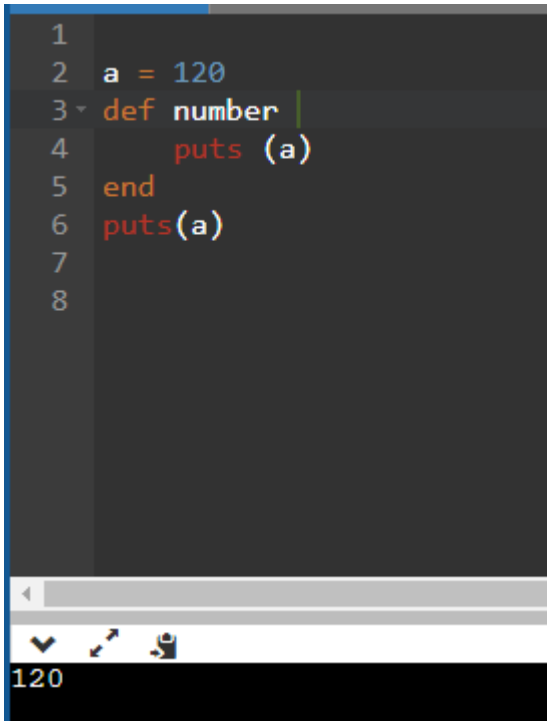
Scope is a region in a code where a variable can be declared, used, or modified. Scope can be done locally and globally. If a variable is declared within a function, it is

called local variable, which cannot be accessed outside the function. If a variable is declared outside the function, it becomes a global variable and can be accessed any time.

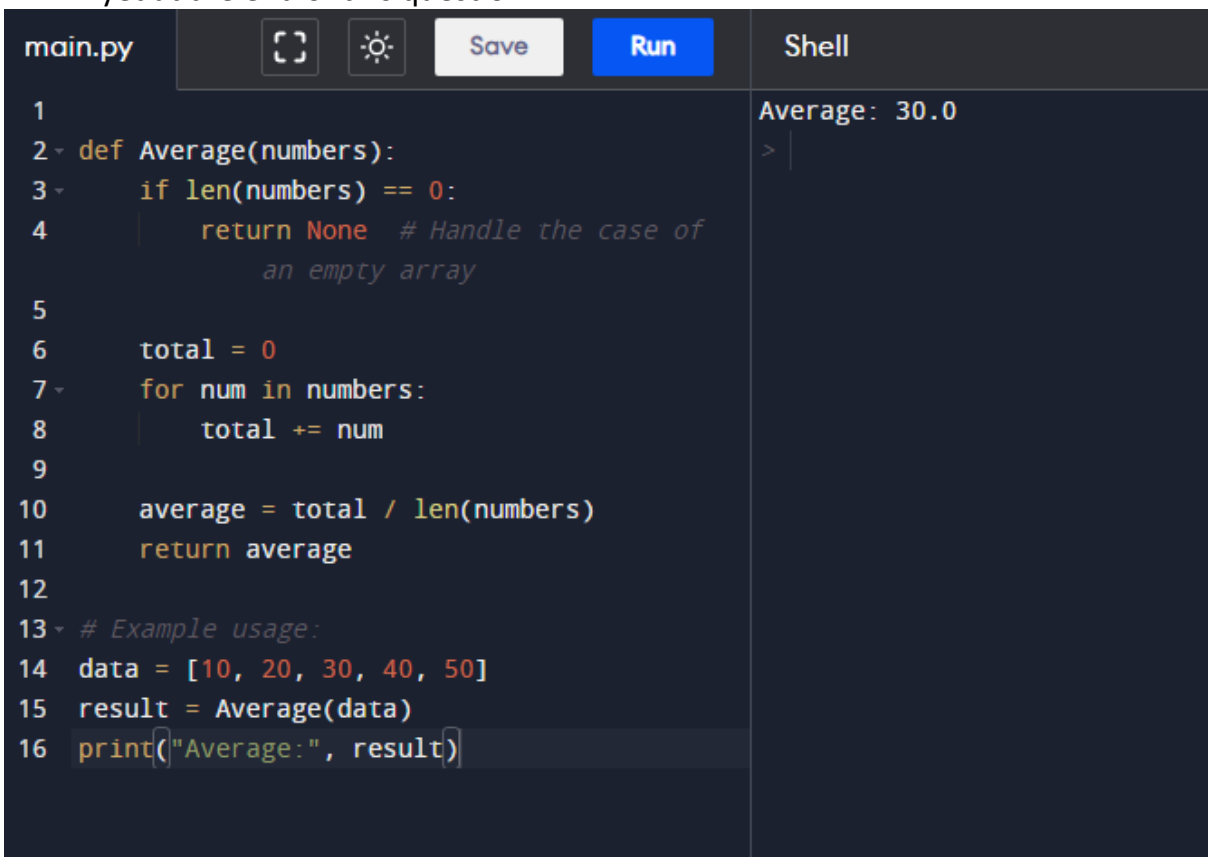
Local:



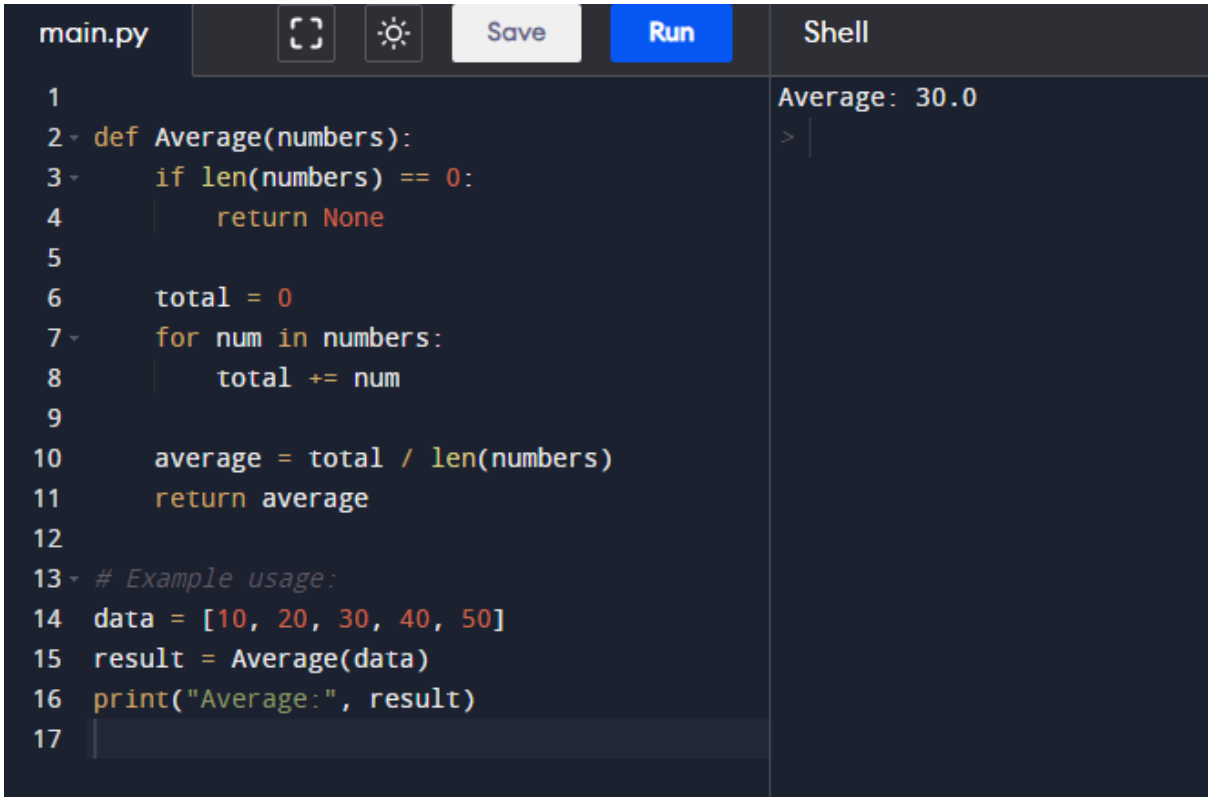
Global:



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.



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

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

main.py

Save Run

```python
def Average(numbers):
    if len(numbers) == 0:
        return None

    total = 0
    for num in numbers:
        total += num

    average = total / len(numbers)
    return average

# Example usage:
data = [10, 20, 30, 40, 50]
result = Average(data)

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

print("Average:", result)
```

Shell

```
Double digits
Average: 30.0
>
```