Experiment no 2

Overview of User Defined Functions in Python

OBJECTIVES:

- To learn the user defined functions for reusability.
- To learn functions as an object and the function values with their return values.

1. INTRODUCTION:

In Python, a user-defined function (UDF) is a block of code that you create to perform a specific task. You can reuse this function multiple times throughout your program, or even in different programs, by calling it with its name. This promotes code reusability, modularity, and readability.

Here's a breakdown of the key aspects of user-defined functions:

```
Python


def function_name(parameters):
  """Function docstring (optional)"""
  # Function body (code to be executed)
  return output_value (optional)
```

Figure 1: User-Defined Function

- **def:** Keyword that indicates you're defining a function.
- **function_name:** A descriptive name that clearly conveys the function's purpose. Choose names that follow Python's naming conventions (lowercase with underscores for separation).
- **parameters (optional):** A comma-separated list of variables that the function can accept as input. These variables are used within the function's body to perform calculations or operations.
- **"""Function docstring (optional)""":** A brief explanation of what the function does and how to use it. This improves code readability and maintainability.
- **# Function body (code to be executed):** The code that performs the intended task of the function. This can involve calculations, manipulations, or other operations.
- **return output_value (optional):** A value that the function returns after its execution. This value can be used in the code where the function is called. If no return value is specified, the function returns None by default.

2. USER DEFINED FUNCTIONS:

2.1 Creating a function

In Python a function is defined using the **def** keyword.

```python
def calculate_area(length, width):
  """
  This function calculates the area of a rectangle given its length and
  """
  # Calculate the area by multiplying length and width
  area = length * width
  # Return the calculated area
  return area

# Call the function with specific values and store the result
rectangle_area = calculate_area(5, 3)

# Print the result
print(f"The area of the rectangle is: {rectangle_area}")
```

Figure 2: Area Calculation Function

1. **Define the function:**
   - `def calculate_area(length, width):` defines the function named `calculate_area` that takes two arguments, `length` and `width`.
   - The docstring, enclosed in triple quotes (`"""`), provides a brief description of the function's purpose. This improves code readability and maintainability.
2. **Function body:**
   - `area = length * width` calculates the area by multiplying the length and width.
   - `return area` returns the calculated area back to the code where the function is called.
3. **Calling the function:**
   - `rectangle_area = calculate_area(5, 3)` calls the `calculate_area` function with the arguments `5` and `3`, which represent the length and width of the rectangle, respectively. The returned area is stored in the variable `rectangle_area`.
4. **Printing the result:**
   - `print(f"The area of the rectangle is: {rectangle_area}")` prints a message with the calculated area using f-strings for formatted string literals.

2.2 Calling a function

To call a function, use the function name followed by parenthesis:

```python
def my_function():
  print("Hello from a function")

my_function()
```

Figure 3: Function Calling

2.3 Arguments

Information can be passed into functions as arguments. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name.
Arguments are often shortened to *args* in Python documentations.

```
def my_function(fname):
  print(fname + " Refsnes")

my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Figure 4: Function Arguments

- **From a function's perspective:**

A parameter is the variable listed inside the parentheses in the function definition.
An argument is the value that is sent to the function when it is called.

2.4 Number of Arguments

By default, a function must be called with the correct number of arguments. Meaning that if your function expects 2 arguments, you have to call the function with 2 arguments, not more, and not less.

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil", "Refsnes")
```

Figure 5: Multiple Arguments

If you try to call the function with 1 or 3 arguments, you will get an error.

This function expects 2 arguments, but gets only 1:

```
def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Emil")
```

Figure 6: Multiple Arguments getting one value

2.5 Arbitrary Arguments, *args

If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition. This way the function will receive a *tuple* of arguments, and can access the items accordingly.
Arbitrary Arguments are often shortened to **args* in Python documentations.

```
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

Figure 7: Multiple Arguments getting one value

2.6 Keyword Arguments

You can also send arguments with the *key = value* syntax. This way the order of the arguments does not matter.

```
def my_function(child3, child2, child1):
  print("The youngest child is " + child3)

my_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")
```

Figure 8: Keyword Arguments

The phrase *Keyword Arguments* are often shortened to *kwargs* in Python documentations.

2.7 Default Parameter Value

The following example shows how to use a default parameter value. If we call the function without argument, it uses the default value.

```
def my_function(country = "Norway"):
  print("I am from " + country)

my_function("Sweden")
my_function("India")
my_function()
my_function("Brazil")
```

Figure 9: Default Arguments

2.8 Passing a List as an Argument

You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function. E.g. if you send a List as an argument, it will still be a List when it reaches the function.

```
def my_function(food):
  for x in food:
    print(x)

fruits = ["apple", "banana", "cherry"]

my_function(fruits)
```

Figure 9: List as Arguments

2.9 Return Values

To let a function return a value, use the return statement.

```
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

Figure 10: Return Values

3. ACTIVITIES:

A1) Write a code and practice all the type of function written in this lab. Give some examples of all of the functions.
A2) Write a code to make a function which uses for loop and give the arguments as a list/array.
A3) Write a code to make a function which uses while loop and give the arguments as a number.
A4) Write a code to make a function which uses nested for and give the arguments as arrays.

IMPORTANT: All the activities` code should be attached to the manual before summary section.

Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation.

Student's Signature:_____ **Date:**_____

LABORATORY SKILLS ASSESSMENT (Psychomotor)

Total Marks:100

Criteria (Max Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S< 70%	Level 3 70% ≤ S< 90%	Level 4 90%≤ S ≤100%	Score (S)
Procedural Awareness (30)	Selects inappropriate skills and/or strategies Required by thetask.	Selects and applies appropriate skills and/or strategies required by the task with major errors.	Selects and applies the appropriate strategies and/or skills specific to the task without significant errors.	Selects and applies appropriate strategies and/or skills specific to the task without any error.	
Practical Implementation (30)	Makes major critical errors in applying procedural knowledge related to python functions.	Makes numerous critical errors in applying procedural knowledge related to python functions.	Makes some non-critical errors in applying procedural knowledge related to python functions.	Applies the procedural knowledge in optimized ways related to python functions.	
Use of Tool/Equipment (30)	Uses tools, equipment and materials with limited competence.	Uses tools, equipment and materials with some competence.	Uses tools, equipment and materials with considerable competence.	Uses tools, equipment and materials with a high degree of competence.	
Safety (10)	Requires constant reminders to follow safety procedures.	Requires some reminders to follow safety procedures.	Follows safety procedures with only minimal reminders.	Routinely follows safety procedures.	
				Marks Obtained	

Instructor Name:_____ **Sign:** _____

LABORATORY SKILLS ASSESSMENT (Affective)

Total Marks: 40

Criteria (Max. Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S < 70%	Level 3 70% ≤ S < 90%	Level 4 90% ≤ S ≤ 100%	Score
Introduction (5)	Very little background information provided or information is incorrect	Introduction is brief with some minor mistakes	Introduction is nearly complete, missing some minor points	Introduction complete and well-written; provides all necessary background principles for the experiment	
Procedure (5)	Many stages of the procedure are not entered on the lab report.	Many stages of the procedure are entered on the lab report.	The procedure could be more efficiently designed but most stages of the procedure are entered on the lab report.	The procedure is well designed and all stages of the procedure are entered on the lab report.	
Data Record (10)	Data is brief and missing significant pieces of information.	Data provides some significant information and has few critical mistakes.	Data is almost complete but has some minor mistakes.	Data is complete and relevant. Tables with units are provided. Graphs are labeled. All questions are answered correctly.	
Data Analysis (10)	Data is presented in very unclear manner.	Data is presented in ways that are not clear enough.	Data is presented in ways that can be understood and interpreted.	Data is presented in ways that best facilitate understanding and interpretation.	
Report Quality (10)	Report contains many errors.	Report is somewhat organized with some spelling or grammatical errors.	Report is well organized and cohesive but contains some grammatical errors.	Report is well organized and cohesive and contains no grammatical errors. Presentation seems polished.	
				Marks Obtained	

LABORATORY SKILLS ASSESSMENT (Cognitive)

Total Marks: 10

(If any) **Marks Obtained**	

Instructor's Signature:_____ **Date:**_____