National University of Computer and Emerging Sciences

Department of Computer Science

FAST-NU, Lahore, Pakistan

Lab 1

Instructor: Mariam Nasim

1 Objectives

After performing this lab, students shall be able to understand:

- ✓ Python data types.
- ✓ Python operators (math, comparison, boolean)
- ✓ Python condition and loops
- ✓ Python functions

Task Distribution

Total Time	170 Minutes
Python data types	15 Minutes
Python operators	10 Minutes
Python if-else conditions	10 Minutes
Python loops	15 Minutes
Python functions	20 Minutes
Exercise	90 Minutes
Online Submission	10 Minutes

2 Python Interpreters

a) Google Colab

Google Colab (short for Colaboratory) is a free, cloud-based Jupyter notebook environment provided by Google. It allows you to:

- Write and execute Python code.
- Use GPUs/TPUs for faster computations.
- Collaborate on projects in real time.
- Access powerful libraries without installation.

Step 1: Access Google Colab

1. **Sign in to Google Account:** Ensure you are logged into your Google account (sign in with your FAST LHR email)
2. **Open Google Colab:**
    - Go to: Google Colab.
    - Alternatively, you can access it via Google Drive:
        - Open Google Drive.
        - Click on **"New" > "More" > "Google Colaboratory"** to create a new notebook.

Step 2: Understand the Notebook Interface

The Google Colab interface is similar to Jupyter Notebook. Key components include:

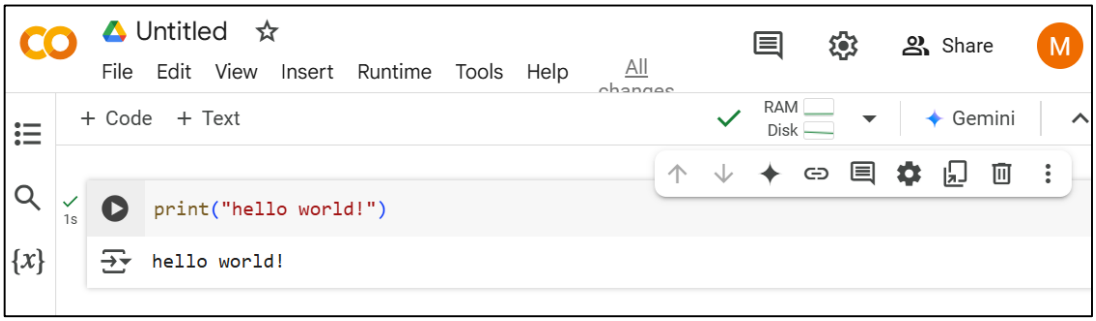
1. **Cells:**
    - **Code Cell:** For writing and running Python code.
    - **Text Cell:** For writing markdown or plain text (e.g., instructions or explanations).
2. **Toolbar:**
    - **File:** To create, open, save, or download notebooks.
    - **Runtime:** To manage the computational backend (e.g., restart runtime, select GPU).
    - **Insert:** To add new cells (code or text).
3. **Sidebar:**
    - Manage files and datasets in the notebook environment.

Step 3: Create a New Notebook

1. **Start a New Notebook:**
    - Open Colab and click on **"New Notebook"**.
    - Alternatively, in Google Drive, go to **"New > More > Google Colaboratory"**.
2. **Rename the Notebook:**
    - Click on the notebook's default name (**Untitled.ipynb**) in the top-left corner.
    - Provide a meaningful name for the notebook.

Step 4: Write and Run Code

1. **Add a Code Cell:**
    - Click on the "+ Code" button to add a new cell.
2. **Write Python Code:**
3. **Run the Code:**
    - Click the **Play** button on the left of the cell or press **Shift + Enter**.



b) Jupyter Notebooks:

Step 1: Download Anaconda

1. Visit the official Anaconda website.
2. Download the appropriate version for your operating system (Windows, macOS, or Linux).
    - Choose the **Python 3.x** version.

Step 2: Install Anaconda

Step 3: Launch Jupyter Notebook via Anaconda Navigator

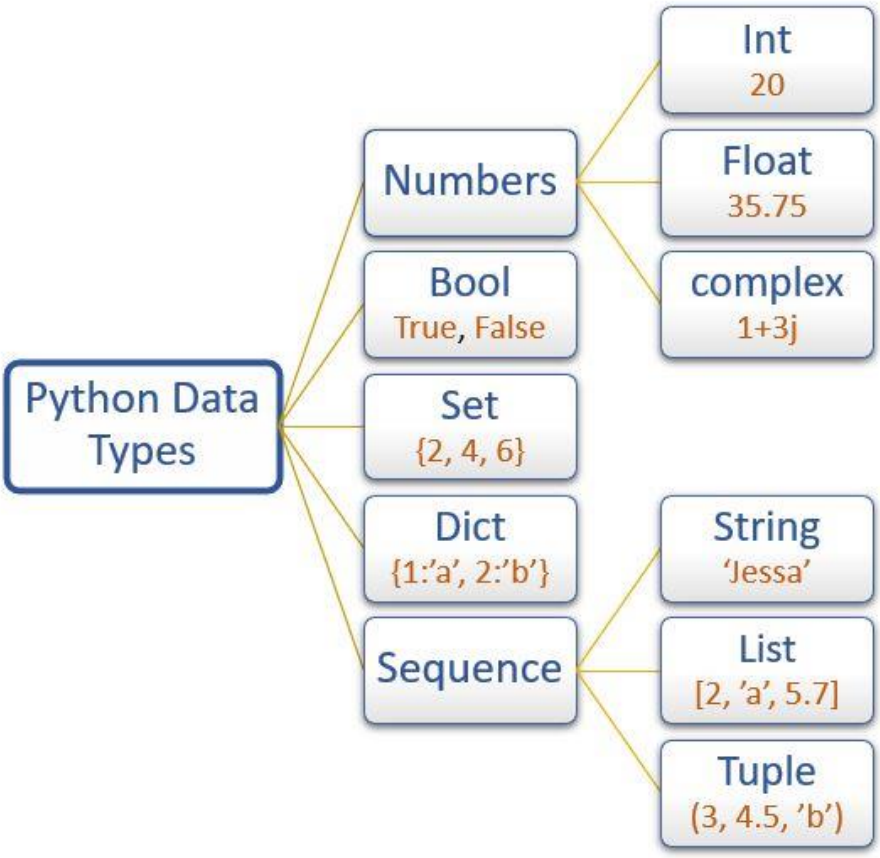
1. Open the **Anaconda Navigator** (search for it in your start menu or applications folder).
2. In Anaconda Navigator, you'll see an option for **Jupyter Notebook**.
3. Click the **Launch** button next to Jupyter Notebook. It will open in your default web browser.

3 Lab Material:

You can refer to the following links, if any details have been missed in the in manuals.

a) Data Types
    a. Built in types: https://www.w3schools.com/python/python_datatypes.asp
    b. Typecasting: https://www.w3schools.com/python/python_casting.asp
b) Operators https://www.w3schools.com/python/python_operators.asp
    a. Math
    b. Comparison
    c. Boolean
c) If-else conditions: https://www.w3schools.com/python/python_conditions.asp
d) Loops: https://www.w3schools.com/python/python_for_loops.asp
    https://www.w3schools.com/python/python_while_loops.asp
e) Functions : https://www.w3schools.com/python/python_functions.asp
f) main(): https://www.geeksforgeeks.org/python-main-function/

Python Data types:



Python has no command for declaring a variable for any datatype. A variable is created the moment you first assign a value to it. Variable names are casesensitive. Just like in other languages, Python allows you to assign values to multiple variables in one line.

The process of explicitly converting the value of one data type (int, str, float, etc.) to another data type is called type casting. In Type Casting, loss of data may occur as we enforce the object to a specific data type.

Example

Integers:

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Operators

This section contains the details of different Python operators i.e. Math operators, comparison operators and Boolean operators.

Arithmetic operators are used with numeric values to perform common mathematical operati

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Operator Precedence:

Operator	Description
`()`	Parentheses
`**`	Exponentiation
`+x` `-x` `~x`	Unary plus, unary minus, and bitwise NOT
`*` `/` `//` `%`	Multiplication, division, floor division, and modulus
`+` `-`	Addition and subtraction
`<<` `>>`	Bitwise left and right shifts
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`==` `!=` `>` `>=` `<` `<=` `is` `is not` `in` `not in`	Comparisons, identity, and membership operators
`not`	Logical NOT
`and`	AND
`or`	OR

If-else:

Python supports conditional statements i.e.if,elif,else. Comparison operators and Boolean operators written in the previous section can be used in if-elif-else statements.

An "if statement" is written by using the if keyword.
Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".

The else keyword catches anything which isn't caught by the preceding conditions.

Example

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Python conditional statements only require `if` statement to execute. Bothelif andelse are optional and used as per requirement.

Loops:

- **While loop**
  - Iterate while the specified condition is True

In [1]:
```
counter = 0
while counter < 5:
    print (f"The value of counter is {counter}")
    counter += 2    # increment counter of 2
```

Out [1]:
```
The value of counter is 0
The value of counter is 2
The value of counter is 4
```

With the `break` statement we can stop the loop even if the while condition is true:

Example

Exit the loop when i is 3:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

With the `continue` statement we can stop the current iteration, and continue with the next:

Example

Continue to the next iteration if i is 3:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example

Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Loop through the letters in the word "banana":

```
for x in "banana":
  print(x)
```

With the `break` statement we can stop the loop before it has looped through all the items:

Example

Exit the loop when `x` is "banana":

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

With the `continue` statement we can stop the current iteration of the loop, and continue with the next:

Example

Do not print banana:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example

Using the start parameter:

```
for x in range(2, 6):
  print(x)
```

Functions

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

In Python a function is defined using the def keyword. To call a function, use the function name followed by parenthesis:

Example

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

my_function()
```

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:

This function expects 2 arguments, and gets 2 arguments:

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

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

Python has a set of built-in functions.

Refer to this link to see a list. https://www.w3schools.com/python/python_ref_functions.asp

Main():

In Python, a “main function” is typically used to denote the entry point of a Python script or program, similar to the main() function in languages like C and C++. While Python doesn’t require a main function to execute scripts, it’s a common practice to define a main function in a script for better organization and readability.

1. Without a Main Function

If you don't use a `main()` function, the code is executed directly from top to bottom. Here's an example:

python Copy Edit

```python
# Without a main function

print("This is a simple Python script.")
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
print(f"The sum is: {num1 + num2}")
```

How it works:

- The script runs as soon as it is executed, starting from the first line.
- The code is executed immediately, and there is no need to call a separate function.

2. With a Main Function

Using a `main()` function is a better practice, especially for larger programs. It allows you to better organize and modularize your code.

Here's an example with a `main()` function:

python Copy Edit

```python
def main():
    # Main program logic here
    print("Welcome to the Python program!")
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    print(f"The sum is: {num1 + num2}")

if __name__ == "__main__":
    main()
```

How it works:

- The `main()` function holds the main logic of the program.
- The check `if __name__ == "__main__":` ensures that `main()` is called only when the script is executed directly (not imported as a module in another script).
- This helps in organizing code, making it modular and reusable.

With main(): Provides better structure, modularity, and organization, especially for larger programs or reusable scripts.

Without main(): Suitable for small scripts or one-time tasks where simplicity is key.

4 Lab Tasks:

Task 1: (3 marks)

Ask the user to provide two float inputs: the base and height of a triangle.

Calculate the area of the triangle.

Print the area (type-cast to an integer).

Task 2: (6 marks)

Use if-else conditions for this question. You are developing a simple program for a parking system. Write a Python program to:

1. Take the vehicle type as input (e.g., "car," "motorcycle," or "truck").
2. Take the parking hours as input (a positive integer).
3. Calculate the parking fee based on the following rules:
    - Car: $5 per hour
    - Motorcycle: $2 per hour
    - Truck: $10 per hour
4. If the parking hours exceed 10 hours, give a 20% discount on the total fee.
5. Print the total fee with an appropriate message.

```
Enter vehicle type (car/motorcycle/truck): car
Enter parking hours: 5

The total parking fee is $25.
```

Input:

```
bash

Enter vehicle type (car/motorcycle/truck): car
Enter parking hours: 12
```

Output:

```
swift

The total parking fee is $48.00 (20% discount applied).
```

Task 3: (5 marks)

Ask the user to input a positive integer. Then Calculate and display the sum of all natural numbers from 1 to that number using a **for** loop.

Task 4: (5 marks)

Repeat task 4 using a **while** loop.

Task 5: (6 marks)

Make a python function named **check_palindrome().**

Function takes a string as input from the user and checks if it is a palindrome. Use the following built-in functions to accomplish the task:

- len() to calculate the length of the string
- str.lower() to handle case insensitivity
- String slicing ([::-1]) to reverse the string

Later call the function check_palindrome() and show some outputs.

Example 1:

Input: `Madam`

Output: `The string is a palindrome.`

Example 2:

Input: `Hello`

Output: `The string is not a palindrome.`

Task 6 (5 marks):

Write a Python function that takes a string and returns the string with all vowels removed. The vowels are 'a', 'e', 'i', 'o', 'u' (both uppercase and lowercase). You can assume that the input string will not be empty.

```
Enter a string: Hello World
```

```
String with vowels removed: Hll Wrld
```

5 Submission Instructions:

- Rename your Jupyter notebook to your "**roll number_Name**" and download the notebook as .ipynb extension.
- To download the required file, go to File->Download .ipynb
- Only submit the .ipynb file. DO NOT zip or rar your submission file
- Submit this file on Google Classroom under the relevant assignment.
- Late submissions will **NOT AT ALL** be accepted.