# Artificial Intelligence Lab

# LAB 2: Python Basics

**Lab Instructor:**
**Ms. Wardah Asad**
**Date: 23-September-2025**

# Python Programming

## Objective:

The objective of this lab will be to learn about variables and the basics of Python and different data types in python with the help of examples and learning tasks.

## Activity Outcomes:

The activities provide hands-on practice with the following topics

- Basics of programming
- Variables
- Print & Eval functions
- Data types

## Instructor Note:

As a pre-lab activity, read the textbook **"Python Basics: A Practical Introduction to Python 3, 2021"**

**Demo of How Python is Handy and useful:** **C++ Program**

main.cpp Share Run

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;

    cout << "Enter the size of the array: ";
    cin >> n;

    int arr[n];

    cout << "Enter " << n << " elements:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    int largest = arr[0];

    for (int i = 1; i < n; i++) {
        if (arr[i] > largest) {
            largest = arr[i];
        }
    }

    cout << "The largest element in the array is: " << largest << endl;
    return 0;
}
```

**Python Program**

```
n = int(input("Enter the size of the list: "))

# Taking list elements
lst = []
for i in range(n):
    num = int(input(f"Enter element {i+1}: "))
    lst.append(num)

largest = max(lst)

print("The largest element in the list is:", largest)
```

**2nd Approach could be:**

```
n = int(input("Enter the size of the list: "))

# Taking list elements
lst = []
for i in range(n):
    num = int(input(f"Enter element {i+1}: "))
    lst.append(num)

# Finding largest
largest = lst[0]
for num in lst:
    if num > largest:
        largest = num

print("The largest element in the list is:", largest)
```

```
Enter the size of the list: 4
Enter element 1: 23
Enter element 2: 67
Enter element 3: 89
Enter element 4: 43
The largest element in the list is: 89
```

- **Python First Program:**

Print function is used to print our output.

```
print("Hello, World!")
```

- **Python Variables:**

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

## Example

```
x = 5 y = "John"
print(x)
print(y)
```

### Get the Type

You can get the data type of a variable with the `type()` function.

## Example

```
x = 5 y = "John"
print(type(x))
print(type(y))
```

- **Python Data Types:**

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default:

```
Str, Int, float, tuple, set, dict, bool
```

- **Python Numbers:**

Variables of numeric types are created when you assign a value to them:

```
x = 1    # int
 y = 2.8  # float
```

**Numeric Operators:**

| Name | Meaning | Example | Result |
|---|---|---|---|
| + | Addition | 34 + 1 | 35 |
| - | Subtraction | 34.0 - 0.1 | 33.9 |
| * | Multiplication | 300 * 30 | 9000 |
| / | Float Division | 1 / 2 | 0.5 |
| // | Integer Division | 1 // 2 | 0 |
| ** | Exponentiation | 4 ** 0.5 | 2.0 |
| % | Remainder | 20 % 3 | 2 |

## Python Strings:

Strings in python are surrounded by either single quotation marks, or double quotation marks.

`'hello'` is the same as `"hello"`.

### Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

Example

```
a = "Hello"
print(a)
```

### String Functions:

1. Length function
2. Slicing
3. Upper case
4. Lower case
5. Remove whitespace
6. Replace string
7. Strip string
8. 8. String Concatenate

## String Formatting (ways of writing):

```
name = "Yahya-al-Sinwar"
marks = 85

#Using string concatenation
print("Hello " + name + ", you scored " + str(marks) + " marks.")

#Using .format()
print("Hello {}, you scored {} marks.".format(name, marks))

#Using f-strings
print(f"Hello {name}, you scored {marks} marks.")
```

```
Hello Yahya-al-Sinwar, you scored 85 marks.
Hello Yahya-al-Sinwar, you scored 85 marks.
Hello Yahya-al-Sinwar, you scored 85 marks.
```

## Python Lists:

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data

### Example

Create a List: `thislist = ["apple", "banana", "cherry"]`

```
print(thislist)
```

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index `[0]`, the second item has index `[1]` etc.

- ## Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the `if` keyword.

## Example

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

# Beauty of Python (List Comprehension in Python)

## Simple Way:

```
squares = []
for x in range(5):
    squares.append(x*x)
```

## Compact Way:

```
squares = [x*x for x in range(5)]

print(squares)
```

**Output:**


# Python While Loop

## Python Loops

Python has two primitive loop commands:

- `while` loops
- `for` loops

## The while Loop

With the `while` loop we can execute a set of statements as long as a condition is true.

### Example

Print i as long as i is less than 6:

```
i = 1
```

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

**Note:** remember to increment i, or else the loop will continue forever.

The `while` loop requires relevant variables to be ready, in this example we need to define an indexing variable, `i`, which we set to 1.

# The break Statement

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
```

# The continue Statement

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)
```

## The else Statement

With the `else` statement we can run a block of code once when the condition no longer is true:

### Example

Print a message once the condition is false:

```
i = 1
while  i <  6 :
  print   (i )          i += 1 else:   print("i is no
longer less than 6")
```

# Python For Loop

## Python For Loops

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

This is less like the `for` keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.
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)
```

The for loop does not require an indexing variable to set beforehand.

# Looping Through a String

Even strings are iterable objects, they contain a sequence of characters:

### Example

Loop through the letters in the word "banana":

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

# The break Statement

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
```

### Example

Exit the loop when x is "banana", but this time the break comes before the print:

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

```
for x in fruits:   if x == "banana":
    break
print(x)
```

# The continue Statement

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)
```

# The range() Function

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 range() function:

```
for x in range(6):

   print(x)
```

Note that `range(6)` is not the values of 0 to 6, but the values 0 to 5.

The `range()` function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: `range(2, 6)`, which means values from 2 to 6 (but not including 6):

### Example

Using the start parameter:

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

   print(x)
```

The `range()` function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: `range(2, 30, 3)` :

### Example

Increment the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):
  print(x)
```

### Example

Print all numbers from 0 to 5, and print a message when the loop has ended:

# Else in For Loop

The `else` keyword in a `for` loop specifies a block of code to be executed when the loop is finished:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

**Note:** The `else` block will NOT be executed if the loop is stopped by a `break` statement.

### Example

Break the loop when `x` is 3, and see what happens with the `else` block:

```
for x in range(6):
  if x == 3: break
  print(x)
else:
  print("Finally finished!")
```

# Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

### Example

Print each adjective for every fruit:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]
for x in adj:
  for y in fruits:
    print(x, y)
```

# The pass Statement

`for` loops cannot be empty, but if you for some reason have a `for` loop with no content, put in the `pass` statement to avoid getting an error.

### Example

```
for x in [0, 1, 2]:
  pass
```

**Lab Tasks:**

1. Write a program that takes three numbers and prints their sum. Every number is given on a separate line.
2. Write a program that takes a number and print its square.
3. Write a program that reads the length of the base and the height of a right angled triangle and prints the area. Every number is given on a separate line.

4. Write a program to check whether an integer is positive, negative, or zero
5. Write a program to input marks of five subjects Physics, Chemistry, Biology, Mathematics, and Computer. Calculate percentage and grade according to following: Percentage >= 90% : Grade A

   Percentage >= 80% : Grade B

   Percentage >= 70% : Grade C

   Percentage >= 60% : Grade D

   Percentage >= 40% : Grade E

   Percentage < 40% : Grade F

6. Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string Sample String : 'abc', 'xyz'

   Expected Result : 'xyc abz'

7. Write a Python program that prints all the numbers from 0 to 6 except 3 and 6.

   Note : Use 'continue' statement.

   Expected Output : 0 1 2 4 5

**8.** Write a Python program to construct the following pattern using for loop.

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

**9.** Write a Python program that iterates the integers from 1 to 50. For multiples of three print "Fizz" instead of the number and for multiples of five print "Buzz". For numbers that are multiples of three and five, print "FizzBuzz".

**Sample Output (starting lines):**

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
```

**10.** Write a Python program that accepts a string and calculates the number of digits and letters.

Sample Data: Python 3.2

**Expected Output:**

Letters: 6

Digits: 2

**11.** Write a Python program to check the validity of passwords input by users.

Validation:

- At least 1 letter between [a-z] and 1 letter between [A-Z].
- At least 1 number between [0-9].
- At least 1 character from [$#@].
- Minimum length 6 characters.
- Maximum length 16 characters.

**12.** Write a Python program that removes and prints every third number from a list of numbers until the list is empty.