Optimization Algorithms: Theory and Software Implementation

Prof. Thirumulanathan D

Department of Mathematics

Institute of IIT Kanpur

Lecture: 6

Introduction to Python Coding

In the first week, we had a brief introduction to optimization. We covered the basics of constrained and unconstrained optimization, the existence and uniqueness of solutions, convex sets, convex functions, the analytical method of computing critical points, and how to determine whether these points are local minima, local maxima, or saddle points.

In the second week, we start with an introduction to Python coding. This is aimed at students who are not familiar with Python. We begin from scratch and rapidly cover the required basics. The goal is not to teach Python exhaustively, but only the parts necessary for coding optimization algorithms.

Getting Started with Python

We will use Google Colab for running Python code. Anyone with a Gmail account can use it. Alternatively, other editors such as Anaconda or PyCharm can also be used.

Print Statements

To print a string:

```
print("Hello world")
print("I am in optimization class")
#output
Hello world
I am in optimization class
```

To print multiple lines using a single print statement:

```
print("Hello world \n I am in
    optimization class ")
#output
Hello world
I am in optimization class
```

The escape sequence \n is used for a newline. Another escape sequence is \t, which introduces a tab:

```
print("Hello\t  World")
        #output
      Hello   World
```

Assignment Operation

To assign values to variables:

```
a = 5
print(a)
# output
The value of a is 5
```

Multiple assignments:

```
a, b, c = 5, 6, 7.2
print (a, b, c)
#output
5 6 7.2
```

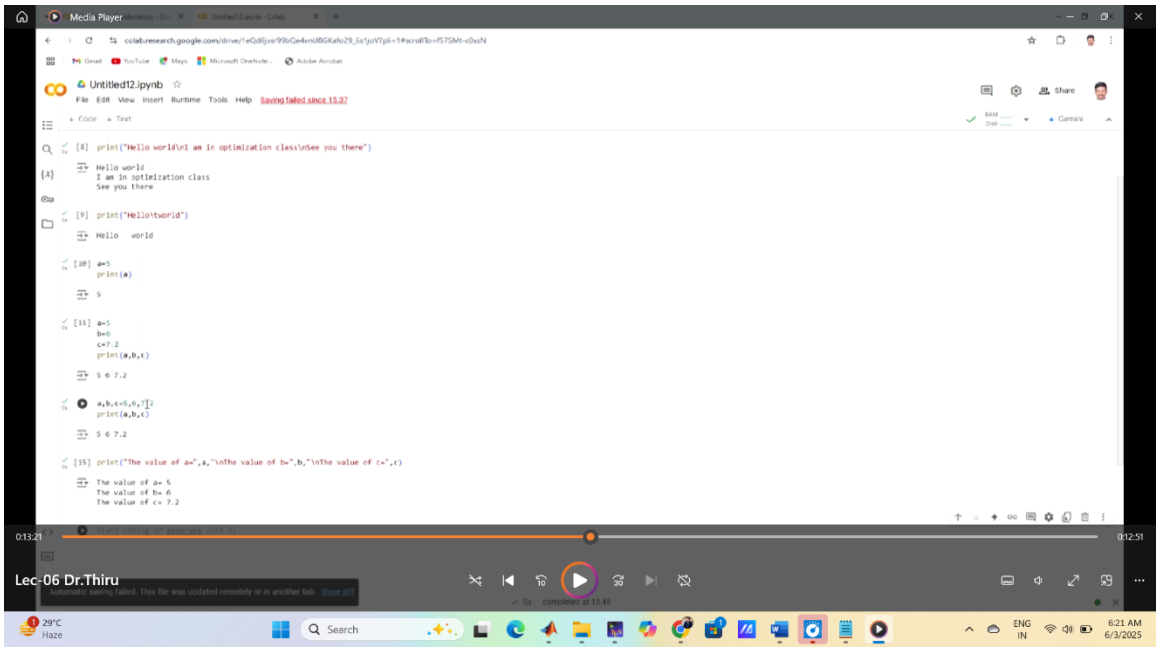
Combining strings and variables in a print statement:

```
print("The value of a is", a)
```

Using escape sequences:

```
  print("The value of a is", a, "\n The value of b is", b, "\n The value of c is", c)
#output
The value of a is 5
The value of b is 6
The value of c is 7.2
```

(Slide time refer 13:21)



Data Types

There are different data types in Python:

Integer: Whole numbers.

Float: Real numbers.

Other types such as strings, hexadecimal, unsigned integers, and long integers are not required for this course.

Floating Point Numbers

Assign and print a floating-point number:

```
x = 1 / 19
print(x)
```

To format the output to three decimal places:

```
print("x = %.3f" % x)
#output
0.053
```

To format with five decimal places:

```
print("x = %.5f" % x)
#output
0.05263
```

This rounds the output rather than truncating it.

Formatting Output

Using format specifiers:

%d: For integers.

%.nf: For floating-point with n digits after the decimal.

%.ne: For scientific/exponential notation. Example

of mixed formatting:

```
x = 1/19
y = 2
print("The values of x and y are: %.4f and %d" % (x, y))
#output
x = 0.0526, y = 2
```

For very small numbers:

```
x = 9e-9
print("x = %.5f" % x)
# Outputs
0.00000, not useful
print("x = %.5e" % x)
# Outputs
9.00000e-09
```

Summary of Format Specifiers

%d: Integer format.

%.nf: Float with n digits after decimal.

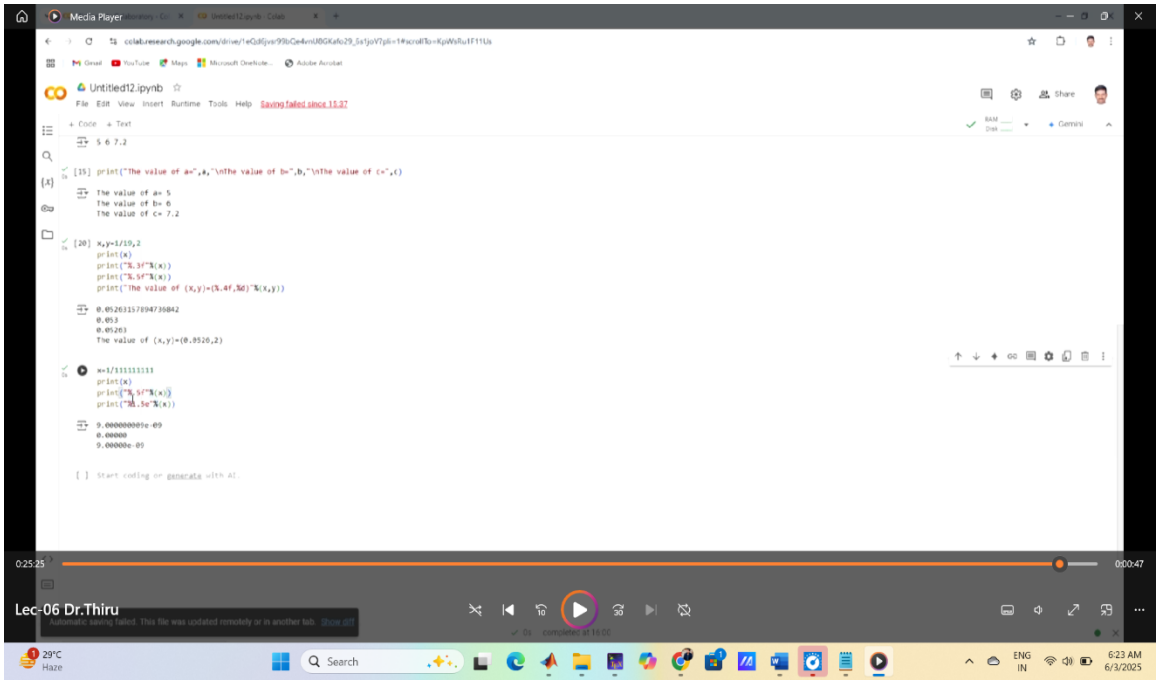
%.ne: Scientific notation (e.g., %.3e gives one digit before decimal, three after).

When using format specifiers, avoid commas within the parentheses. The following is incorrect:

```
# Incorrect
print("x = %.3f", x)

# Correct
print("x = %.3f" % x)
```

(Slide time refer 25:25)



These formatting tools will be useful when presenting numerical results. In the next lecture, we will discuss arithmetic operations.

Thank you.