



# Fourth Industrial Summer School

Day 1

## Python programming I

# Session Objectives

- ✓ Introduction to Python programming
  - History
  - IDE
- ✓ First Python Program
- ✓ Numbers
  - Integer
  - Float
  - Complex
- ✓ Operators
  - Arithmetic
  - Logical
  - Bitwise



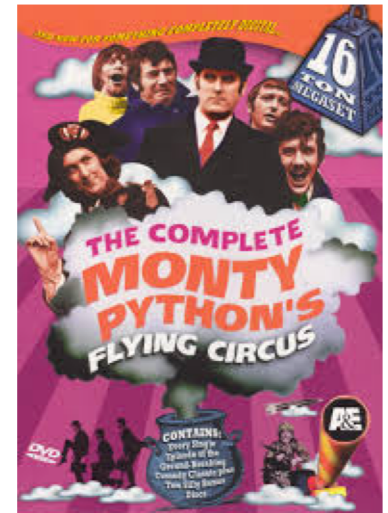


# Python History

- Guido Van Rossum
  - First version of Python code (version 0.9.0) in 1991
  - Python version 1.0 was released in January 1994
  - .....
  - Recent major release version **3**



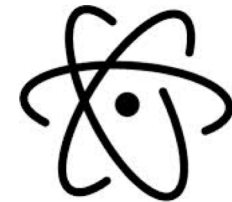
- Python? A snake?



# Python Programming - IDEs



PyCharm



Atom



# Python Programming - IDE

- Google Colab
  - Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.



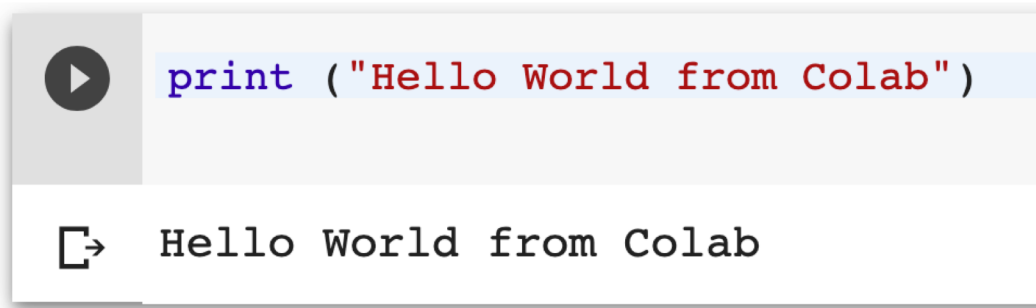
[colab.research.google.com](https://colab.research.google.com)

- Benefits
  - Free GPU 😊
  - Pre-installed libraries
  - Jupyter Notebook
  - Collaboration feature
  - Stored on the cloud (e.g. Google Drive)

# First Python Program in Colab

- The famous “Hello World” example

1. Start Colab notebook
2. Output the String “Hello World from Colab”
  - using the `print` function



A screenshot of a Google Colab code cell. The top part shows a play button icon and the code `print ("Hello World from Colab")` in a light blue background. The bottom part shows the output `Hello World from Colab` in a white background with a copy icon to the left.

```
print ("Hello World from Colab")
```

```
Hello World from Colab
```

Must do  
before  
proceeding

# Comments

- The symbol `#` is used to indicate comments in a python program



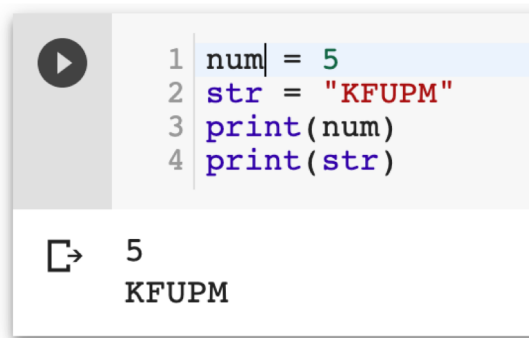
```
# My first comment in Python  
print ("Hello World from Colab") #Inline comment
```

```
☞ Hello World from Colab
```

# Python Variables

## ■ Creating Variables

- A variable is created when you assign a value to it.
- No variable declaration

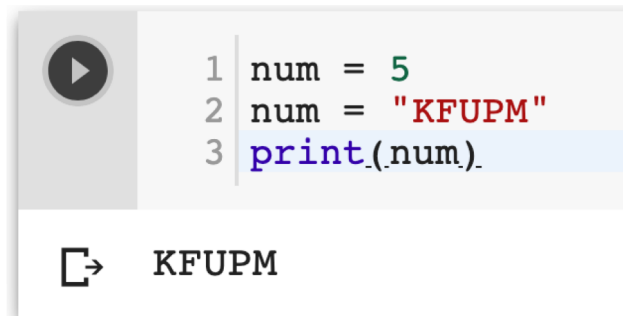


```
1 num = 5
2 str = "KFUPM"
3 print(num)
4 print(str)
```

5  
KFUPM

This code block shows a Python interpreter window. It contains four lines of code: line 1 assigns the value 5 to the variable 'num'; line 2 assigns the string 'KFUPM' to the variable 'str'; line 3 prints the value of 'num'; and line 4 prints the value of 'str'. Below the code, the output is displayed: the number 5 on the first line and the string 'KFUPM' on the second line.

## ■ Change type – Simple !!



```
1 num = 5
2 num = "KFUPM"
3 print(num)
```

KFUPM

This code block shows a Python interpreter window. It contains three lines of code: line 1 assigns the value 5 to the variable 'num'; line 2 reassigns the variable 'num' with the string 'KFUPM', effectively changing its type; line 3 prints the value of 'num'. Below the code, the output is displayed: the string 'KFUPM'.



# Python Variables



- Naming Variables
- Similar to other programming languages. Python has rules for naming variables
  - Must start with a letter or the underscore character
  - Can't start with a number
  - Name can only contain alpha-numeric characters and underscores
  - Names are case-sensitive

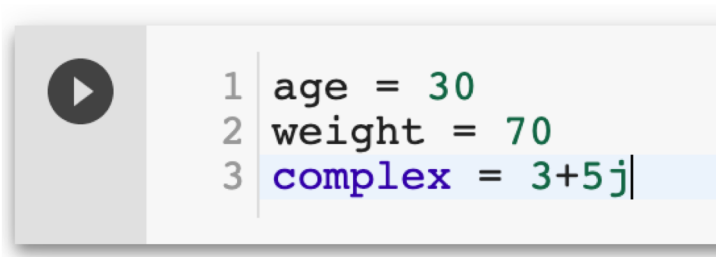
# Python Variables



- Numbers
  - Integer
  - Float
  - Complex
  
- Collections (Arrays)
  - List
  - Tuple
  - Dictionary
  
- Strings

# Python Numbers

- Numeric types in Python
  - `int`
  - `float`
  - `complex`



```
1 age = 30
2 weight = 70
3 complex = 3+5j|
```

# Input

- Read from user using the input function



```
1 name = input("What's your name? ")  
2 print("Nice to meet you " + name.)
```

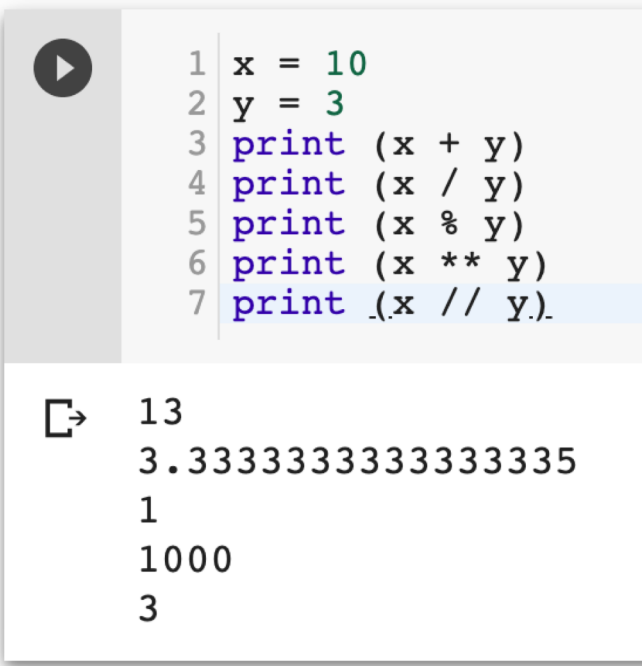


```
What's your name? Ahmed Alahmed  
Nice to meet you Ahmed Alahmed
```

# Operators

## ■ Arithmetic operators

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Power
//	Floor division

A screenshot of a Python code execution environment. It shows a list of seven lines of code: 1 x = 10, 2 y = 3, 3 print (x + y), 4 print (x / y), 5 print (x % y), 6 print (x \*\* y), and 7 print (x // y). The output of these operations is displayed below the code: 13, 3.3333333333333335, 1, 1000, and 3. The code is in a light gray box with a play button icon on the left. The output is in a white box with a copy icon on the left.

```
1 x = 10
2 y = 3
3 print (x + y)
4 print (x / y)
5 print (x % y)
6 print (x ** y)
7 print (x // y)
```

13  
3.3333333333333335  
1  
1000  
3

# Operators

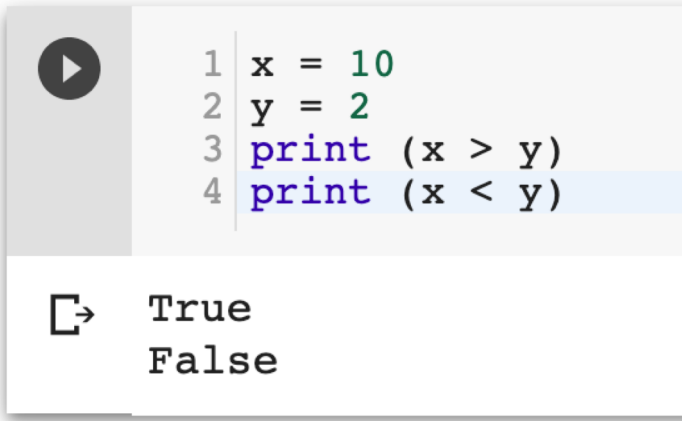
- Assignment operators
  - $x -= 10$  means  $x = x - 10$

Operator	Name
=	Assignment
+=	Addition then Assignment
-=	Subtraction then Assignment
*=	Multiplication then Assignment
/=	Division then Assignment
%=	Modulus then Assignment
**=	Power then Assignment
//=	Floor division then Assignment

# Operators

- Comparison operators
  - Widely used in **conditional** statements and **loops**

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

A screenshot of a code editor or terminal window showing a Python script being executed. The script consists of four lines: line 1 sets x to 10, line 2 sets y to 2, line 3 prints the result of x > y, and line 4 prints the result of x < y. The output shows 'True' for the first comparison and 'False' for the second. The code is color-coded: numbers are grey, variables are green, and print statements are blue.

```
1 x = 10
2 y = 2
3 print (x > y)
4 print (x < y)
```

True  
False

# Operators

- Logical operators
  - **Compound** statements

Operator	Usage
and	True if both True
or	True if either True
not	Reverse



```
1 x = 10
2 y = 2
3 print (x > 5 and y > 10)
4 print (x > 5 or y > 10)
5 print (not (x > 5)).
```



```
False
True
False
```



# Operators

- Bitwise operators
  - used to compare **binary** numbers

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits

Hands on session

Problem Solving