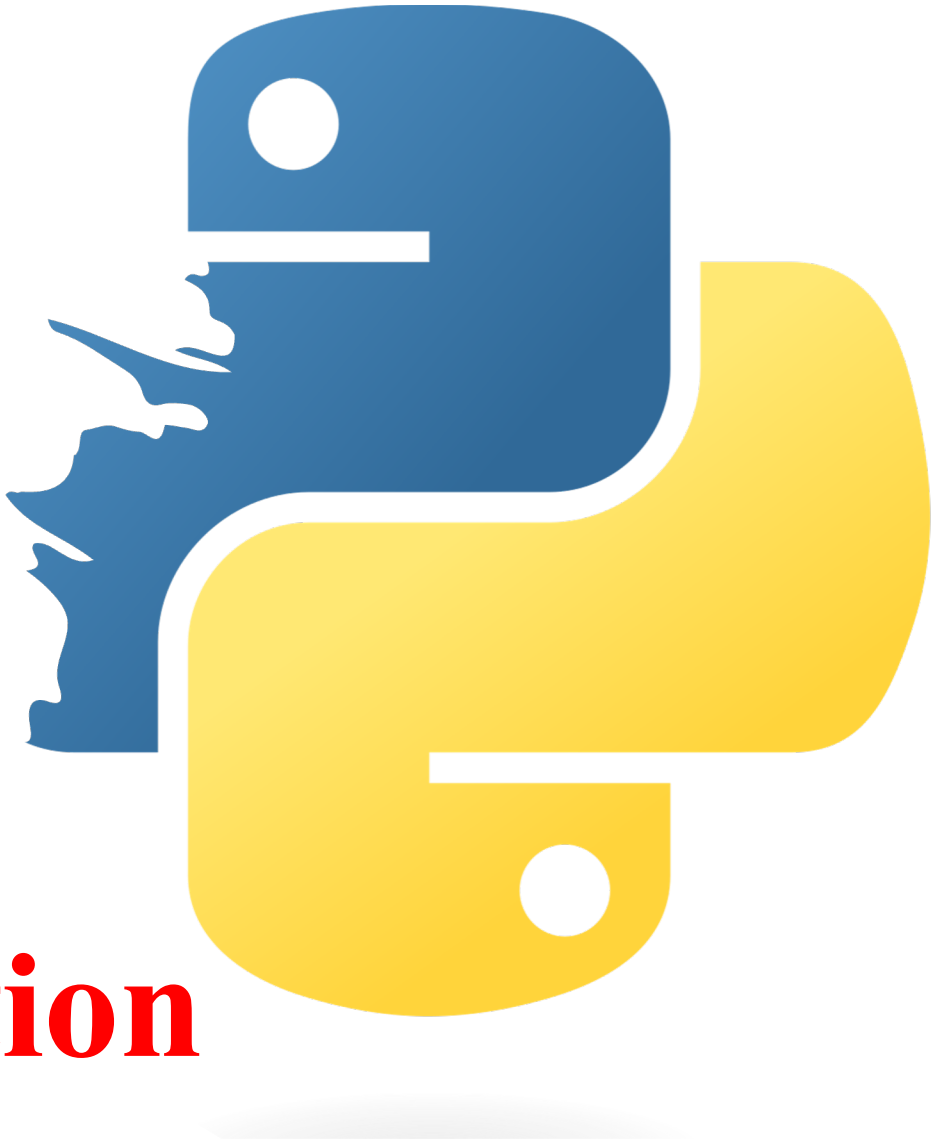
DATA 3401

Basic Python

Dr. Max (Masoud) Rostami

Department of Science / College of Science
Data Science Program / College of Science
The University of Texas Arlington, Texas

Python Installation



Anaconda and Google Colab

Anaconda: Anaconda is a free and open-source distribution of Python and R, specifically designed for data science and machine learning. Being a local platform, it requires download and installation on your machine, which offers the advantage of offline access and customization. One of the flagship tools bundled with Anaconda is the Jupyter Notebook, an open-source web application.

Google Colab: is a cloud-based platform that offers an interactive environment similar to Jupyter Notebooks. It allows users to write and execute Python code directly from their browsers without any setup. Being integrated with Google Drive, it ensures easy sharing and real-time collaboration, although it does require an internet connection to access.

Comparing

Anaconda and Google Colab

Features/Attributes	Google Colab	Anaconda
Platform	Cloud-based web application	Local installation (cross-platform: Windows, macOS, Linux)
Setup	Zero setup; access via a web browser	Requires downloading and installing packages
Access	Internet connection required	Offline access after initial setup
Cost	Free	Free (Commercial versions available for enterprise needs)
Computational Power	Offers free limited access to GPUs and TPUs	Depends on local machine's resources
Environment	Pre-configured Python environment	Customizable Python environments via conda
Package Management	Pip (with some pre-installed packages)	Conda (comprehensive package & environment manager)
Libraries & Extensions	Some popular libraries pre-installed; others can be added	Wide range of packages available through Conda repositories
Integration	Integrated with Google Drive	Local file system or any cloud storage if configured
Collaboration	Real-time collaboration & easy sharing via Google Drive	No built-in real-time collaboration

Accessing Google Colab:

- Open a browser and go to Google Colab or Welcome To Colaboratory

- You can access it directly via the browser without any installations.

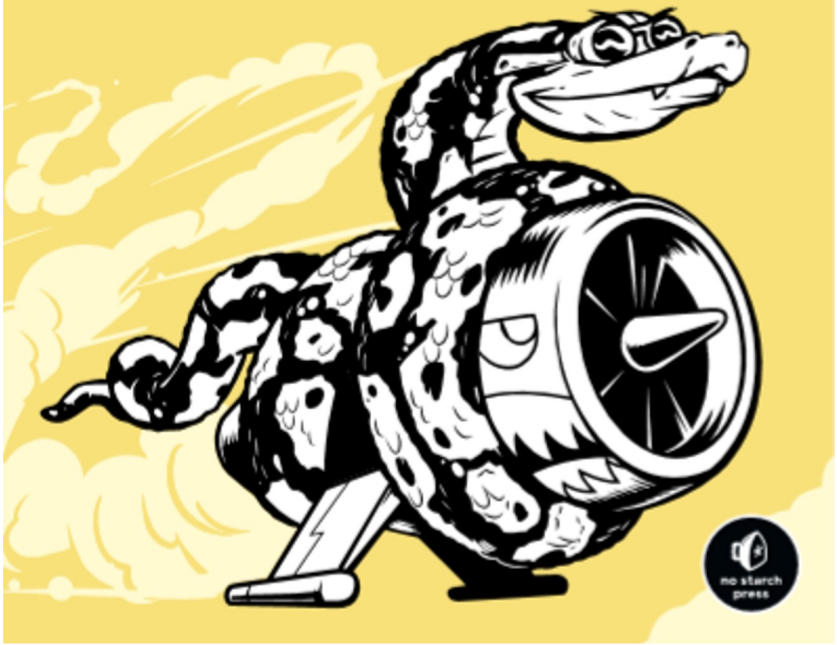
Anaconda

https://www.anaconda.com/products/individual

PYTHON CRASH COURSE

A HANDS-ON, PROJECT-BASED INTRODUCTION TO PROGRAMMING

ERIC MATTHES



no starch press

Variables:

Variables are used to store data in Python. They are created using an assignment statement and can hold various types of values.

```
message = "Hello, World!"

count = 10
```

Rules for creating variables in Python

1.Naming Rules:

- Variable names must start with a letter (a-z, A-Z) or an underscore (_).
- The rest of the variable name can contain letters, numbers, and underscores.
- Variable names are case-sensitive, so "myVariable" and "myvariable" are different variables.

2. Descriptive and Meaningful Names:

- Choose variable names that are descriptive and meaningful, representing the purpose or content of the variable.
- Use lowercase letters and underscores to separate words in variable names (e.g., my_variable, user_age).

3. Avoid Reserved Words:

Reserved words, also known as keywords, are predefined words in Python that have special meanings and **cannot be used as variable names**.

Here are some examples of reserved words in Python:

for, while, if, else, import, from, break, continue, return, def, class, try, except, finally, raise, with, yield, lambda, nonlocal, global, assert, async, await, pass, del, is, in, not, and, or.

4. Avoid Built-in Function Names:

- Avoid using names of built-in functions or commonly used modules as variable names, as it can lead to confusion and potential conflicts.
- Examples of built-in function names to avoid as variable names include print, input, list, str, dict, type, etc.

5. Use Snake Case for Multiple Words:

- In Python, it is common to use snake case (all lowercase with words separated by underscores) for variable names with multiple words.
- For example: student_name, total_score, is_authenticated.

6. Be Consistent:

1. Maintain consistency in variable naming throughout your codebase.
2. Stick to a naming convention that makes sense to you and your team and follow it consistently.

Here are some examples of valid variable names:

myVariable

variable1

_variable

variable_name

And here are some examples of invalid variable names:

1variable (starts with a number)

variable-name (contains a hyphen, which is not an alphanumeric character or underscore)

Example of Correct Variable Names:

```
name = "John Doe"
age = 25
is_student = True
total_marks = 95.5
```

Example of Incorrect Variable Names:

```
3marks = 90  # Starts with a number
my-variable = 10  # Contains a hyphen
for = 5  # Reserved word
```

Operators:

Python provides various operators to perform operations on variables and values.

These include

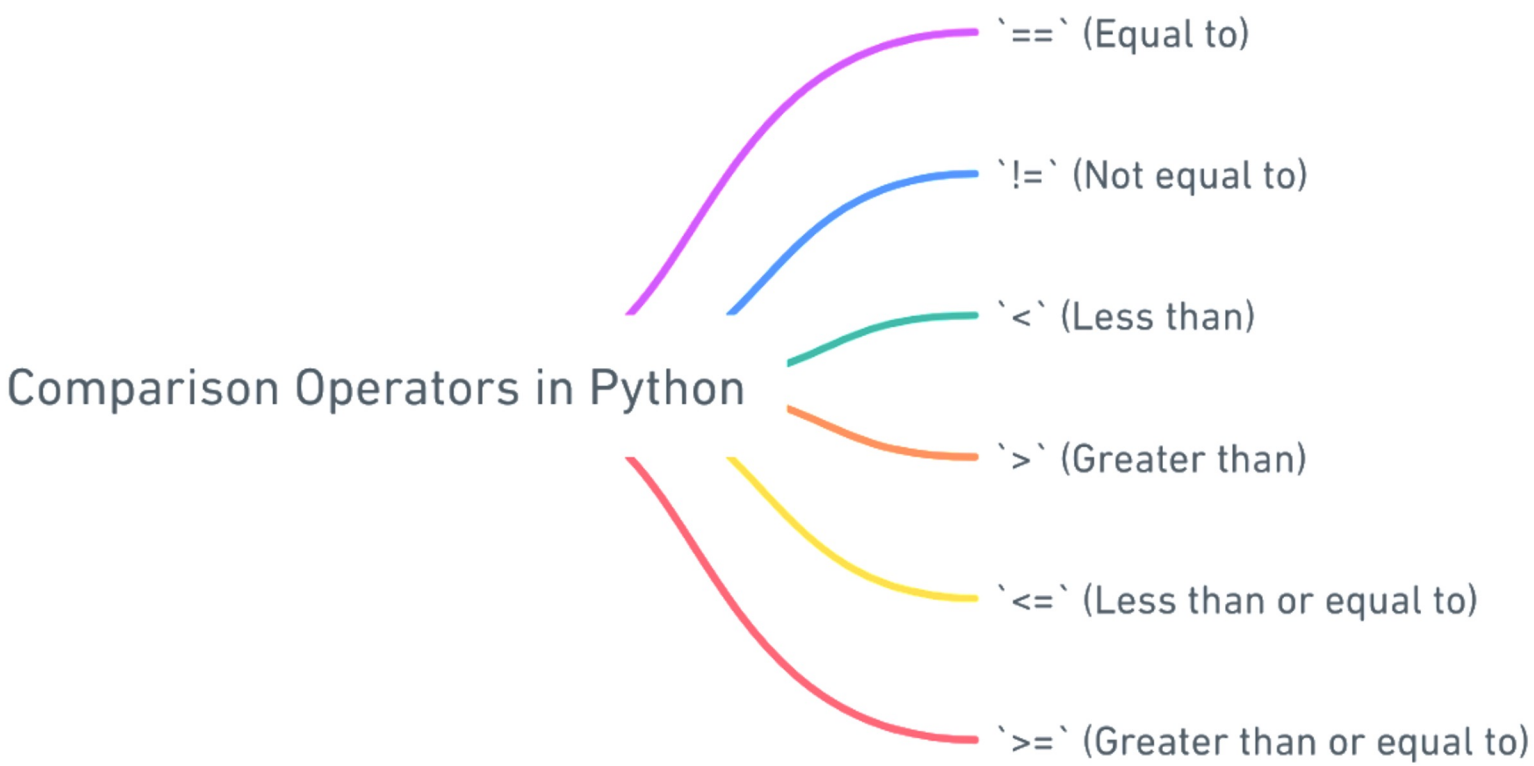
1) Arithmetic operators (+, -, *, /, %, etc.),

2) Assignment operators (=, +=, -=, etc.),

3) Comparison operators (==, !=, >, <, etc.),

4) Logical operators (and, or, not), and more.

Comparison Operators in Python



Expressions

Expressions are combinations of values, variables, operators, and function calls that evaluate to a single value. They can be simple or complex, and they are used extensively in Python to perform calculations and produce results.

```
x = 5

y = 3

z = x + y  # z is now 8
```

Data Types:

Python supports several built-in data types, including:

2. **Numeric types**: int (integers), float (floating-point numbers), complex (complex numbers).
3. **Sequence types**: list (mutable lists), tuple (immutable sequences), str (strings).
4. **Mapping type:** dict (dictionaries).
5. **Set types**: set (unordered collection of unique elements), frozenset (immutable set).
6. **Boolean type**: bool (represents True or False).