Python3 Tutorial

Debajyoti Halder

CSE 357 – Statistical Methods for Data Science
Fall 2024
Instructor – Dr. Anshul Gandhi

Outline

- Get Python3
- Hello world !
- Basics
  - Data Structures – strings, lists, etc.
  - Built-in functions – **`print, len, sum,`** etc.
  - Control flow – loops, conditional statements
- Standard libraries
  - **`random`**
  - **`math`**
  - **`statistics`**
- **`NumPy`** – faster arrays, matrices
- **`pandas`** – data analysis
- **`Matplotlib`** – plotting

Get Python3

Get started right away using Google Colab → `colab.new`

Jupyter Notebook → `jupyter.org/install`

Download and install Python → `python.org/downloads`

Use **VS Code** or any IDE

Usually built-in – GNU/Linux

`python.org/shell → to get started rn !`

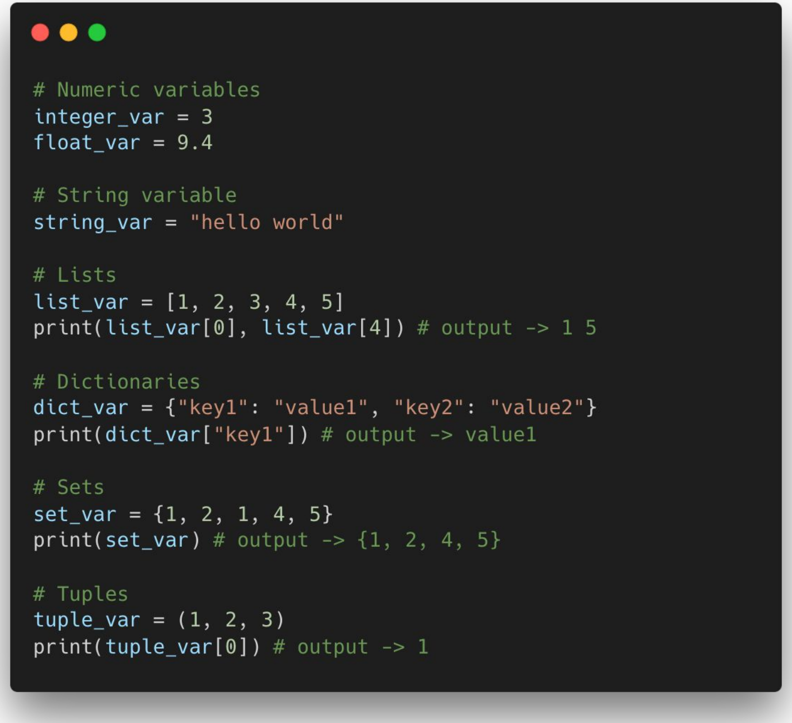
Hello world !

```
print("Hello world !")
```

Basics

Data Structures

- Variables
  - Numeric – integer, float – `3, 56, -4, 9.4, -4.56`
  - Strings – “**hello world**”
- Lists
  - Arrays – `[2, “four”, 6, [‘e’, ‘i’, ‘g’, ‘h’, ‘t’]]`
  - Index – `0` → `n-1`, where n is length of list
- Dictionaries
  - key-value pair, hashmap
  - key is the index
- Sets, Tuples



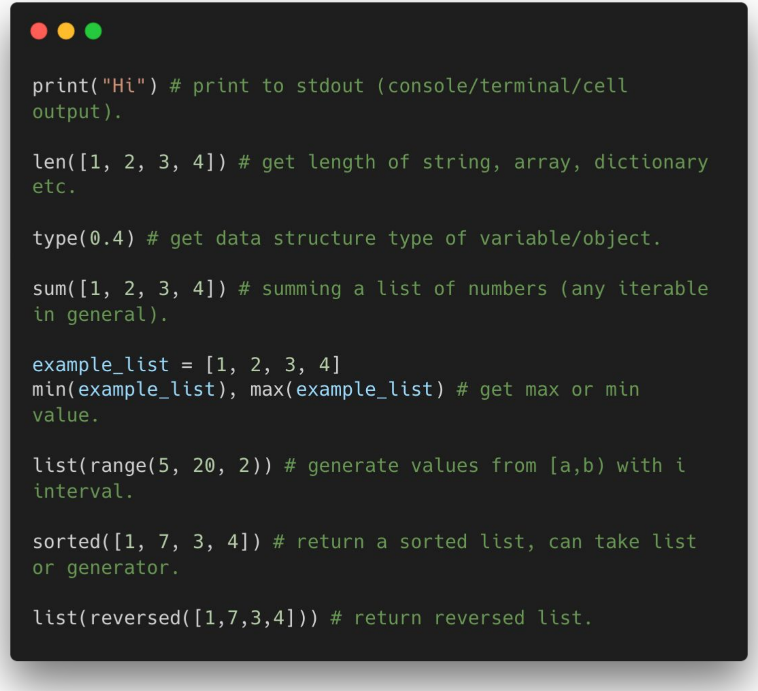
```
# Numeric variables
integer_var = 3
float_var = 9.4

# String variable
string_var = "hello world"

# Lists
list_var = [1, 2, 3, 4, 5]
print(list_var[0], list_var[4]) # output -> 1 5

# Dictionaries
dict_var = {"key1": "value1", "key2": "value2"}
print(dict_var["key1"]) # output -> value1

# Sets
set_var = {1, 2, 1, 4, 5}
print(set_var) # output -> {1, 2, 4, 5}

# Tuples
tuple_var = (1, 2, 3)
print(tuple_var[0]) # output -> 1
```

Keywords and conventions

- Keywords (Don’t use as variable names)
  - **`and, in, for, with, or, is, import, global, class, break, return`** etc.
- Identifiers, Variable names
  - Uppercase, lowercase, underscore, digits
  - Case-sensitive
- Indentation is important in python

Useful built-in functions

- `print()` – Print to stdout (console/terminal/cell output).
- `len()` – Get length of string, array, dictionary etc.
- `type()` – Get data structure type of variable/object.
- `sum()` – Summing a list of numbers (any ***iterable*** in general).
- `min()`, `max()` – Get max or min value.
- `range(a,b,i)` – Generate values from `[a,b)` with `i` interval.
- `list()` – Create a list, can take a generator like `range()`.
- `sorted()` – Return a sorted list, can take list or generator.
- `reversed()` – Return reversed list.

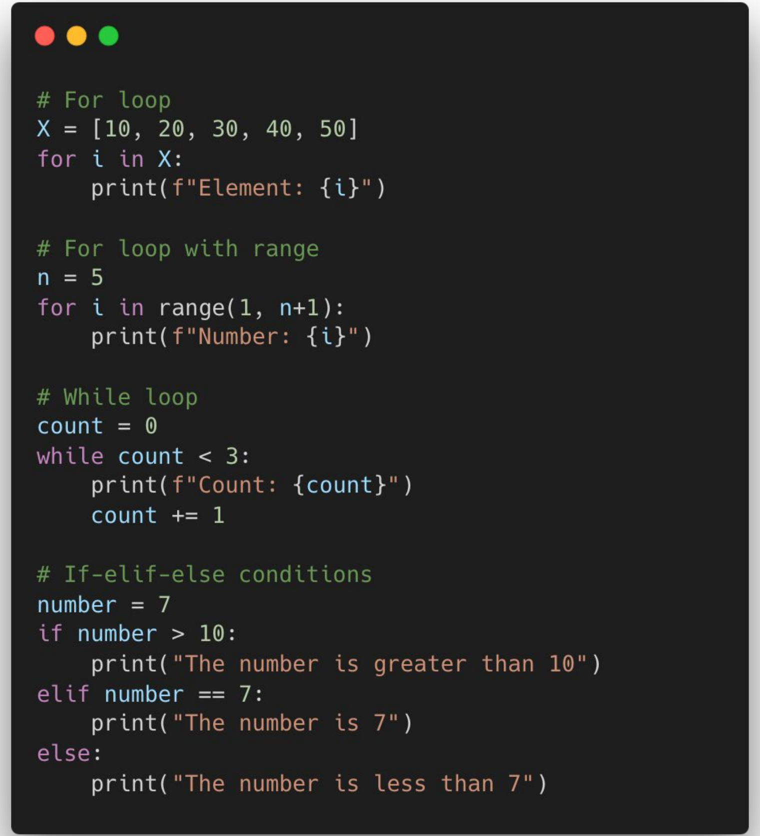


```
print("Hi") # print to stdout (console/terminal/cell
output).

len([1, 2, 3, 4]) # get length of string, array, dictionary
etc.

type(0.4) # get data structure type of variable/object.

sum([1, 2, 3, 4]) # summing a list of numbers (any iterable
in general).

example_list = [1, 2, 3, 4]
min(example_list), max(example_list) # get max or min
value.

list(range(5, 20, 2)) # generate values from [a,b) with i
interval.

sorted([1, 7, 3, 4]) # return a sorted list, can take list
or generator.

list(reversed([1,7,3,4])) # return reversed list.
```

Control flow statements

- Loops → `for, while`
- Conditionals → `if...else`

Don’t forget the **colon**, and **indentation**

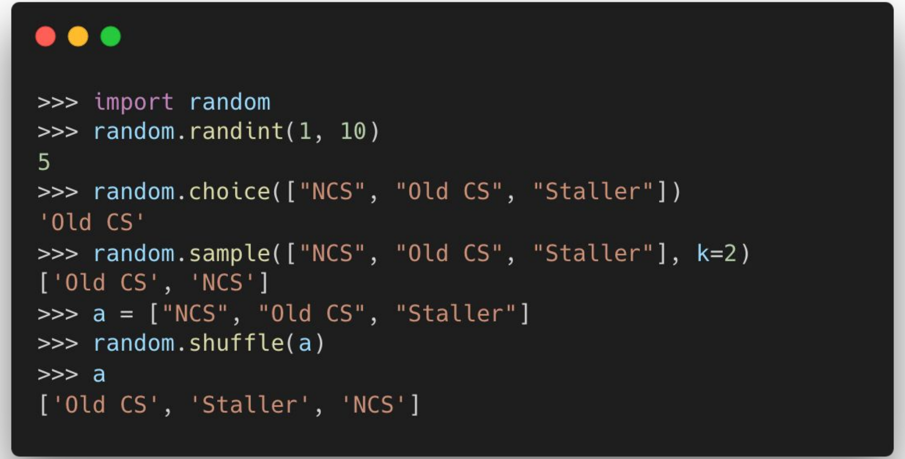


```python
# For loop
X = [10, 20, 30, 40, 50]
for i in X:
    print(f"Element: {i}")

# For loop with range
n = 5
for i in range(1, n+1):
    print(f"Number: {i}")

# While loop
count = 0
while count < 3:
    print(f"Count: {count}")
    count += 1

# If-elif-else conditions
number = 7
if number > 10:
    print("The number is greater than 10")
elif number == 7:
    print("The number is 7")
else:
    print("The number is less than 7")
```

Standard libraries

random

- **`random.randint(a, b)`** – Return a random integer N such that **`a <= N <= b`**
- **`random.choice(seq)`** – Return a random element from the non-empty sequence **`seq`**
- **`random.choices(population, k)`** – List of length **`k`**, containing unique elements chosen from **`population`**
- **`random.sample(population, k=1)`** – List of length **`k`**, elements chosen from **`population`** with replacement
- **`random.shuffle(seq)`** – Shuffle the list **`seq`** in place

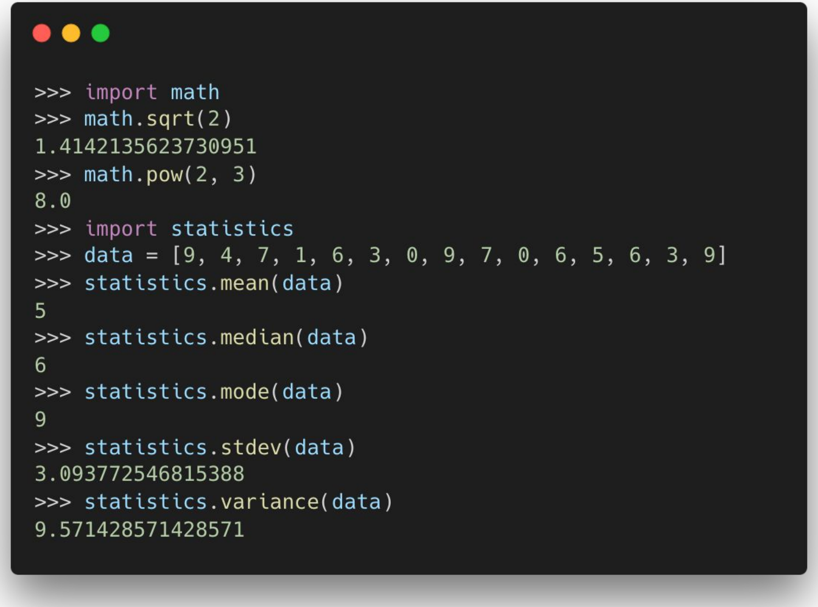


```
>>> import random
>>> random.randint(1, 10)
5
>>> random.choice(["NCS", "Old CS", "Staller"])
'Old CS'
>>> random.sample(["NCS", "Old CS", "Staller"], k=2)
['Old CS', 'NCS']
>>> a = ["NCS", "Old CS", "Staller"]
>>> random.shuffle(a)
>>> a
['Old CS', 'Staller', 'NCS']
```

math and statistics

- `math.sqrt(x)` – Return square root of `x`
- `math.pow(a,b)` – Return a^b - can also use `a**b`

- `statistics.mean(data)` – Average, equivalent to `sum(data)/len(data)`
- `statistics.median(data)`
- `statistics.mode(data)`
- `statistics.stdev(data)` – Standard deviation
- `statistics.variance(data)`



```
>>> import math
>>> math.sqrt(2)
1.4142135623730951
>>> math.pow(2, 3)
8.0
>>> import statistics
>>> data = [9, 4, 7, 1, 6, 3, 0, 9, 7, 0, 6, 5, 6, 3, 9]
>>> statistics.mean(data)
5
>>> statistics.median(data)
6
>>> statistics.mode(data)
9
>>> statistics.stdev(data)
3.093772546815388
>>> statistics.variance(data)
9.571428571428571
```

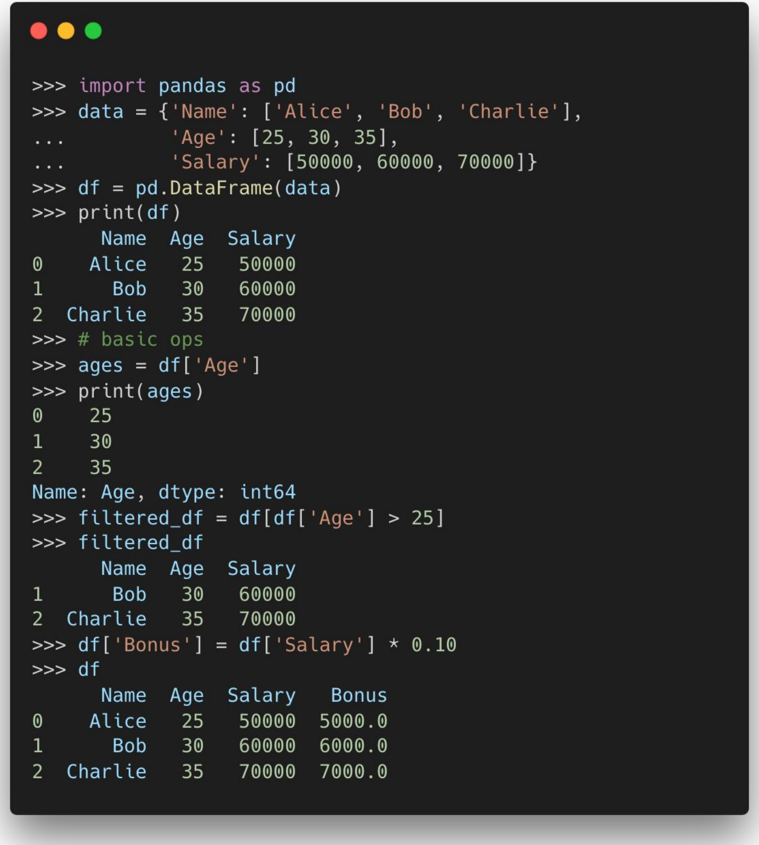
NumPy

- Library for **faster** numerical and n-dimensional array (eg. **matrices**) operations.
- Supports broadcasting – Element-wise operations on arrays of different shapes.

- `import numpy as np`
- `np.array()`
- `np.asmatrix()`
- `np.random.randn(d0...dn)`
- `np.random.permutation(x)`
- `np.matmul()` – matrix multiplication, `np.dot()` – scalar multiplication
- `np.mean()`, `np.median()`, `np.percentile()`

pandas

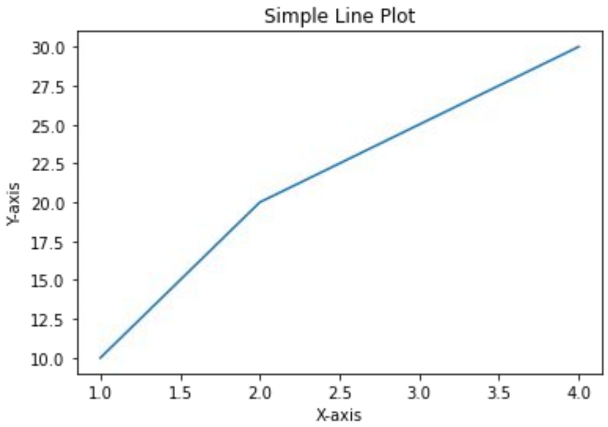
- Library for **data analysis**
- pandas data structures –
  - Series: 1-D labeled array
  - DataFrame: 2-D labeled table
- Some key pandas functions – **head()**, **tail()**, **describe()**, **groupby()**, **merge()**, **sort_values()**
- 10 mins to pandas – pandas.pydata.org/docs/user_guide/10min.html

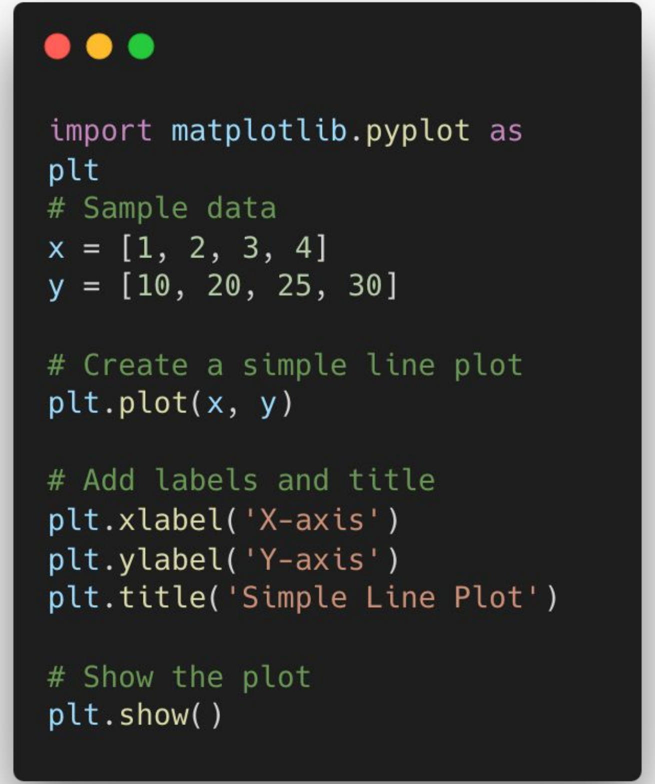


```
>>> import pandas as pd
>>> data = {'Name': ['Alice', 'Bob', 'Charlie'],
...         'Age': [25, 30, 35],
...         'Salary': [50000, 60000, 70000]}
>>> df = pd.DataFrame(data)
>>> print(df)
      Name  Age  Salary
0    Alice   25   50000
1      Bob   30   60000
2  Charlie   35   70000
>>> # basic ops
>>> ages = df['Age']
>>> print(ages)
0    25
1    30
2    35
Name: Age, dtype: int64
>>> filtered_df = df[df['Age'] > 25]
>>> filtered_df
      Name  Age  Salary
1      Bob   30   60000
2  Charlie   35   70000
>>> df['Bonus'] = df['Salary'] * 0.10
>>> df
      Name  Age  Salary   Bonus
0    Alice   25   50000  5000.0
1      Bob   30   60000  6000.0
2  Charlie   35   70000  7000.0
```

Matplotlib

- Library for **data visualization**
- Static, animated, and interactive plots
- Plots – line, bar, scatter, pie, histogram, etc.
- Works well with **NumPy** and **pandas**





```
import matplotlib.pyplot as
plt
# Sample data
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]

# Create a simple line plot
plt.plot(x, y)

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Simple Line Plot')

# Show the plot
plt.show()
```

Resources

- Python: The Ultimate Beginner's Guide!
- Python Library Reference
  - https://docs.python.org/3/library/index.html
- Plotting in Python: matplotlib
  - https://matplotlib.org/stable/tutorials/pyplot.html
- PyCharm: Python IDE
  - https://www.jetbrains.com/pycharm/
- Transforming code into Beautiful, Idiomatic Python
  - https://www.youtube.com/watch?v=OSGv2VnC0go&t=1861s