Python for Beginners

S. Sharma

Assistant Professor,
Department of Theory of Nuclear system
sushil.sharma@uj.edu.pl
http://koza.if.uj.edu.pl/staff/ssharma

Programming languages

To instruct the computers to perform some sort of actions

Programming languages

To instruct the computers to perform some sort of actions

Low-level **high-level**

Programming languages

To instruct the computers to perform some sort of actions

Low-level **high-level**

Machine language(M.L):
written in binary numbers or bits, that computer can understood

Programming languages

To instruct the computers to perform some sort of actions

Low-level — **high-level**

Machine language(M.L):
written in binary numbers or bits, that computer can understood

Assembly language(A.L):
Based on the human-readable mnemonics codes for instructions, which can be converted into machine learning language (0,1), using the *assemblers.*
"like M.L, A.L also requires knowledge of internal computer architecture"

Programming languages

To instruct the computers to perform some sort of actions

Low-level

Machine language(M.L):
written in binary numbers or bits, that computer can understood

Assembly language(A.L):
Based on the human-readable mnemonics codes for instructions, which can be converted into machine learning language (0,1), using the *assemblers.*
"like M.L, A.L also requires knowledge of internal computer architecture"

high-level

In HLL, programs can be written independent to the details of the computer. They are called high level as they are close to human languages.

Programming languages

To instruct the computers to perform some sort of actions

Low-level

Machine language(M.L):
written in binary numbers or bits, that computer can understood

Assembly language(A.L):
Based on the human-readable mnemonics codes for instructions, which can be converted into machine learning language (0,1), using the *assemblers.*
"like M.L, A.L also requires knowledge of internal computer architecture"

high-level

In HLL, programs can be written independent to the details of the computer. They are called high level as they are close to human languages.

FORTRAN (FORmula TRANslation)

COBOL (COmmon Business Oriented Language)

SQL (**S**tructured **Q**uery **L**anguage)

……..

Programming languages

To instruct the computers to perform some sort of actions

Low-level

Machine language(M.L):
written in binary numbers or bits, that computer can understood

Assembly language(A.L):
Based on the human-readable mnemonics codes for instructions, which can be converted into machine learning language (0,1), using the *assemblers.*
"like M.L, A.L also requires knowledge of internal computer architecture"

high-level

In HLL, programs can be written independent to the details of the computer. They are called high level as they are close to human languages.

FORTRAN (FORmula TRANslation)

COBOL (COmmon Business Oriented Language)

SQL (**S**tructured **Q**uery **L**anguage)

……..

Object-**O**riented **L**anguage
C++, C#, Java, Python

Why Python(3) ?

Why Python(3) ?

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Why Python(3) ?

Characteristics and applications

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Why Python(3) ?

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Characteristics and applications

- Support structured, functional and OOP

Why Python(3) ?

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Characteristics and applications

- Support structured, functional and OOP
- Scripted language (eg., PERL, …)

Why Python(3) ?

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Characteristics and applications

- Support structured, functional and OOP
- Scripted language (eg., PERL, …)
- Easily integrated with other languages: C, C++, Java,..

Why Python(3) ?

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Characteristics and applications

- Support structured, functional and OOP
- Scripted language (eg., PERL, …)
- Easily integrated with other languages: C, C++, Java,..
- Easy to learn, read and maintain

Why Python(3) ?

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Characteristics and applications

- Support structured, functional and OOP
- Scripted language (eg., PERL, …)
- Easily integrated with other languages: C, C++, Java,..
- Easy to learn, read and maintain
- Code writing faster and shorter, supported by tons of dedicated Python libraries (data visualization, machine learning, data science, web development,..)

Why Python(3) ?

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Characteristics and applications

- Support structured, functional and OOP
- Scripted language (eg., PERL, …)
- Easily integrated with other languages: C, C++, Java,..
- Easy to learn, read and maintain
- Code writing faster and shorter, supported by tons of dedicated Python libraries (data visualization, machine learning, data science, web development,..)
- Portable (same interface for various platform)

Why Python(3) ?

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Characteristics and applications

- Support structured, functional and OOP
- Scripted language (eg., PERL, …)
- Easily integrated with other languages: C, C++, Java,..
- Easy to learn, read and maintain
- Code writing faster and shorter, supported by tons of dedicated Python libraries (data visualization, machine learning, data science, web development,..)
- Portable (same interface for various platform)
- GUI Programming ,

Why Python(3) ?

Key features:

Interpreted language

(no compilation required before execution)

Interactive

(direct interaction with python prompt)

Object Oriented Programming supported

(apart from structured and functional approach)

Characteristics and applications

- Support structured, functional and OOP
- Scripted language (eg., PERL, …)
- Easily integrated with other languages: C, C++, Java,..
- Easy to learn, read and maintain
- Code writing faster and shorter, supported by tons of dedicated Python libraries (data visualization, machine learning, data science, web development,..)
- Portable (same interface for various platform)
- GUI Programming ,

and many more.....

What we will cover in this course

What we will cover in this course

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

What we will cover in this course

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on If, elif and else
- *For* loop
- *While* loop
- Useful operators

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- global/local
- Functions calling functions

Reading/Writing Files

- Files and directories in Python
- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

- Standard Python library

  Datetime module
  Math and Random module
  Generators and decorators
  NumPy, Pandas, Matplotlib (basic uses)

What we will cover in this course

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- global/local
- Functions calling functions

Reading/Writing Files

- Files and directories in Python
- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

- Standard Python library

  Datetime module
  Math and Random module
  Generators and decorators
  NumPy, Pandas, Matplotlib (basic uses)

Object Oriented Programming

- Introduction to OOPs
- Attributes and Class keywords
- Inheritance and Polymorphism

Statistical analysis of data with Python

What we will cover in this course

Introduction (an overview to course)

- Getting started "Hello World"
- Keywords and identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* statement
- Elif statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

What we will cover in this course

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* statement
- Elif statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

Identifiers: is a name, to identify *variable*, *function*, *module*….

Keywords: predefined words, that has specific meaning and can't be use as constants, variables or other identifier names.

Variables:

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and Assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and Assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on If, elif and else
- *For* loop
- *While* loop
- Useful operators

DATA types & structure

Introduction (an overview to course)

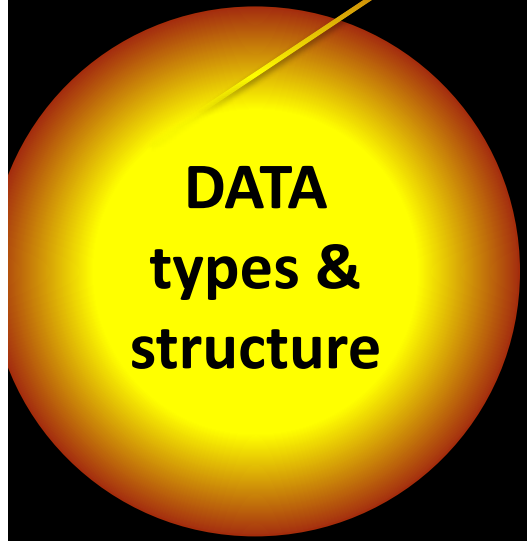
- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and Assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on If, elif and else
- *For* loop
- *While* loop
- Useful operators



NUMBERS

Introduction (an overview to course)

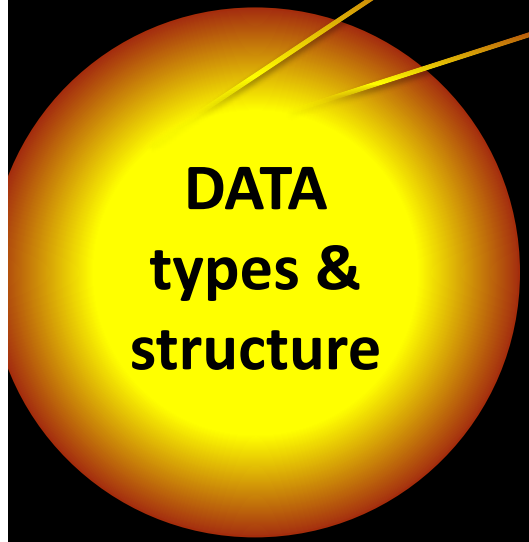
- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and import
- Object Oriented Programming

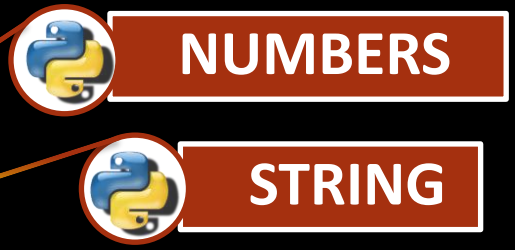
Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on If, elif and else
- *For* loop
- *While* loop
- Useful operators





Introduction (an overview to course)

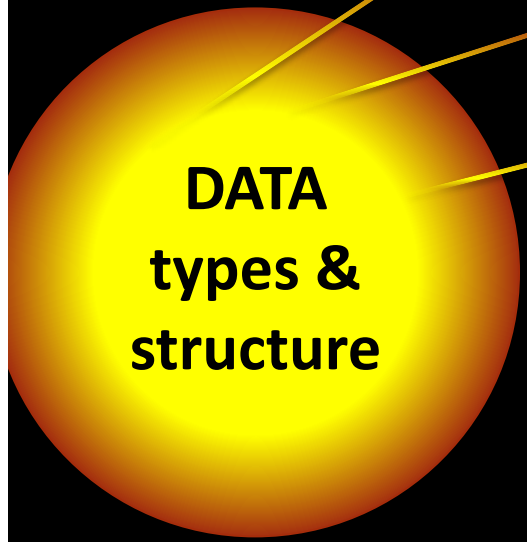
- Getting started "Hello World"
- Keywords and identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and import
- Object Oriented Programming

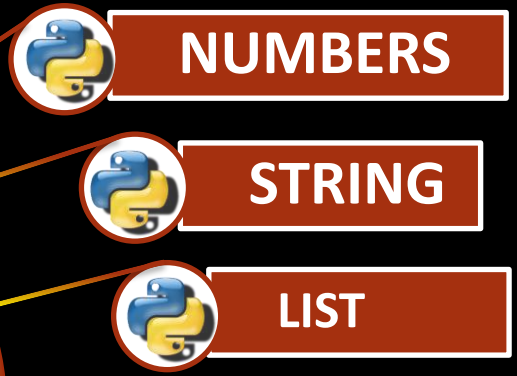
Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on If, elif and else
- *For* loop
- *While* loop
- Useful operators





Introduction (an overview to course)

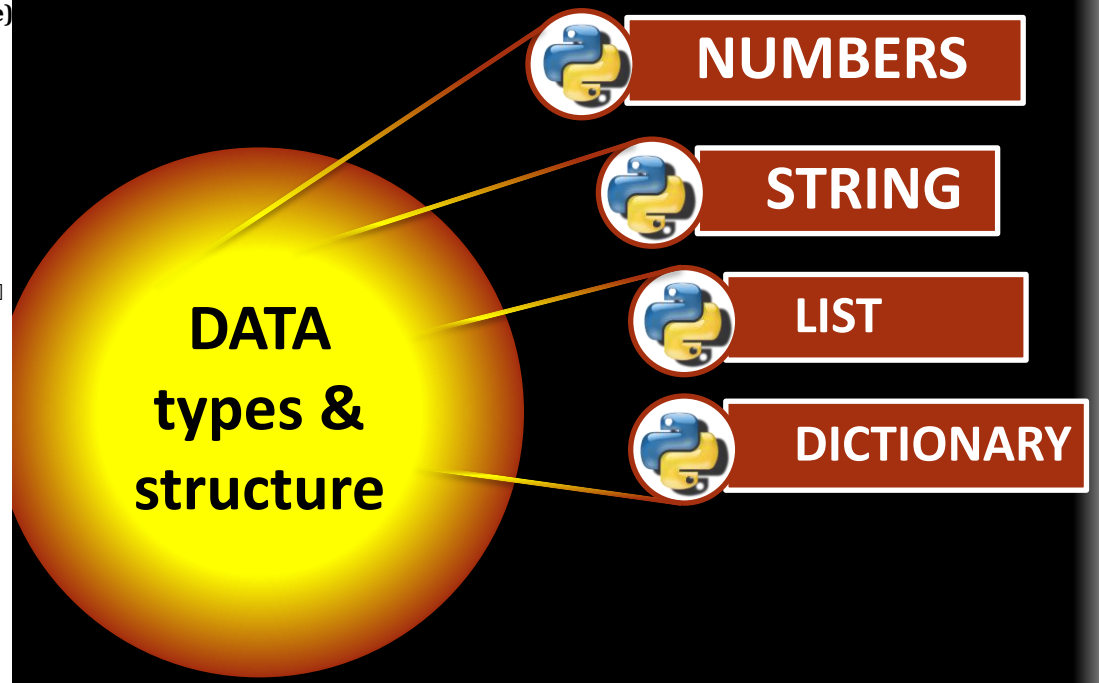
- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on If, elif and else
- *For* loop
- *While* loop
- Useful operators



Introduction (an overview to course)

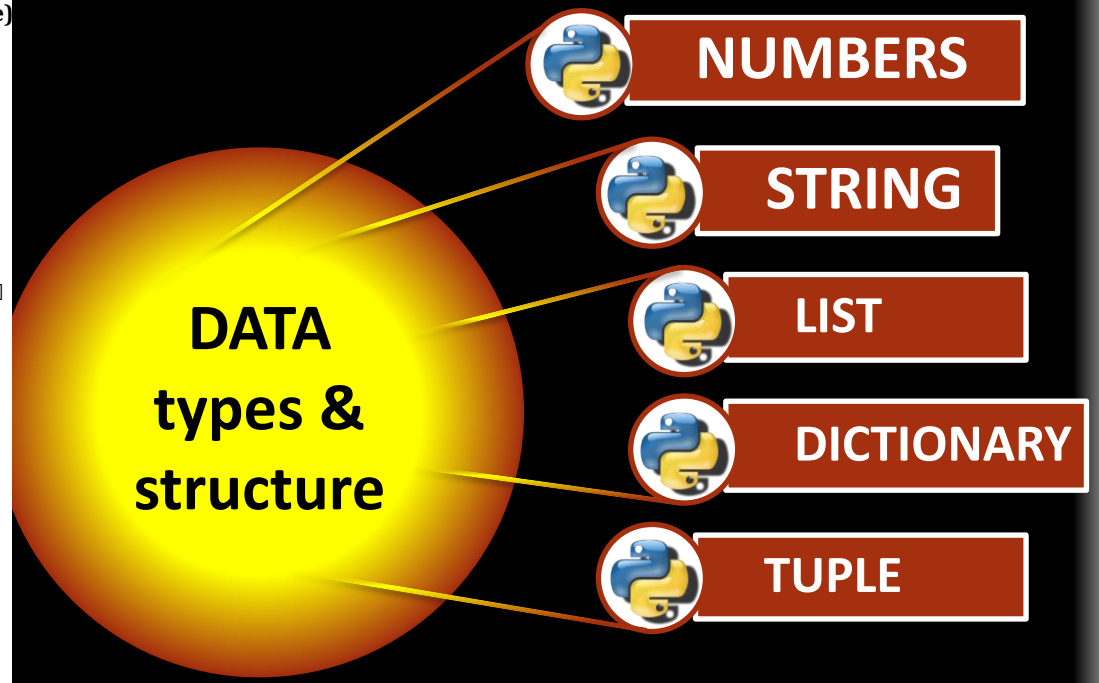
- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and Assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on If, elif and else
- *For* loop
- *While* loop
- Useful operators



Introduction (an overview to course)

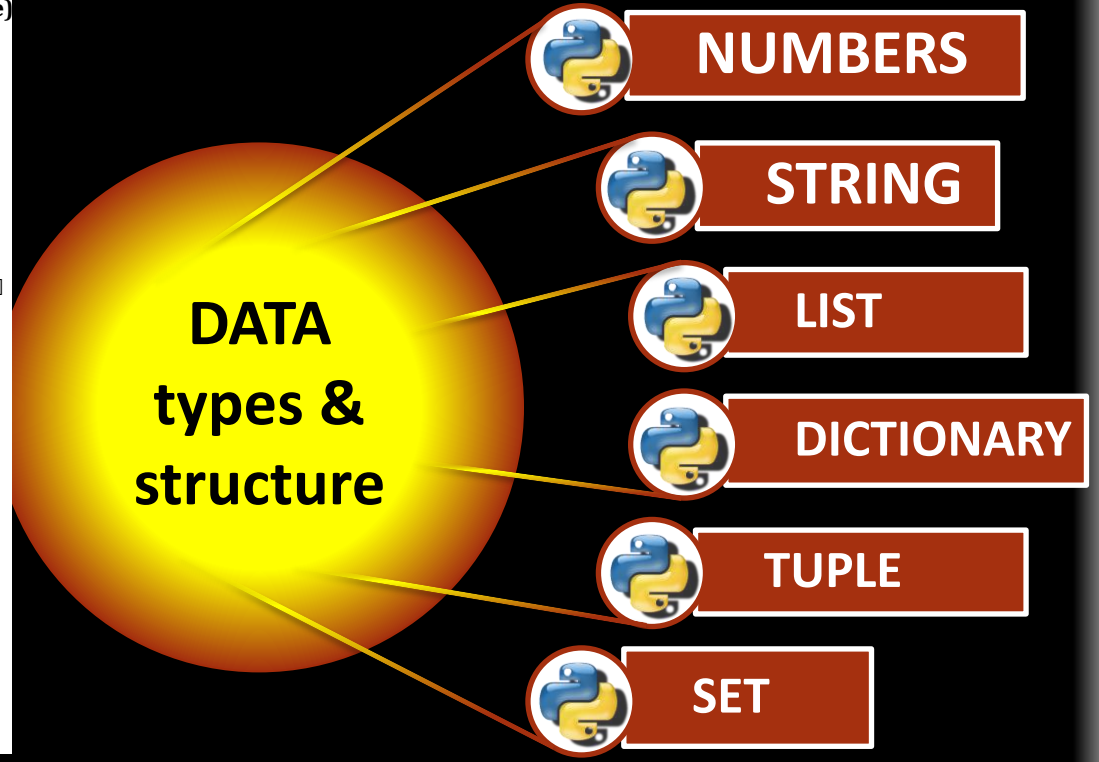
- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators



NUMBERS → **Numeric values** (e.g., Integers, real numbers, complex)

NUMBERS → **Numeric values** (e.g., Integers, real numbers, complex)

STRING → **Contiguous set** of *characters* in quotation marks (‘Hello’)

NUMBERS → **Numeric values** (e.g., Integers, real numbers, complex)

STRING → **Contiguous set** of *characters* in quotation marks (‘Hello’)

LIST → **Compound data type** : composed of *items* in square bracket[], separated by commas [‘abc’, 5, 10.5, ‘xyz’]

SEQUENCE Type

NUMBERS → **Numeric values** (e.g., Integers, real numbers, complex)

STRING → **Contiguous set** of *characters* in quotation marks (‘Hello’)

LIST → **Compound data type** : composed of *items* in square bracket[], separated by commas [‘abc’, 5, 10.5, ‘xyz’]

TUPLE → **Similar to the list**, however, tuples are **enclosed within parenthesis** (‘abc’, 786, 2.23, ‘john’)

SEQUENCE Type

- **NUMBERS** → **Numeric values** (e.g., Integers, real numbers, complex)
- **STRING** → **Contiguous set** of *characters* in quotation marks (‘Hello’)
- **LIST** → **Compound data type** : composed of *items* in square bracket[], separated by commas [‘abc’, 5, 10.5, ‘xyz’]
- **TUPLE** → **Similar to the list**, however, tuples are **enclosed within parenthesis** (‘abc’, 786, 2.23, ‘john’)
- **DICTIONARY** → **Dictionaries** consist of key-value pairs {1001: "John", 1002: "Jane”}

SEQUENCE Type

NUMBERS → **Numeric values** (e.g., Integers, real numbers, complex)

STRING → **Contiguous set** of *characters* in quotation marks (‘Hello’)

LIST → **Compound data type** : composed of *items* in square bracket[], separated by commas [‘abc’, 5, 10.5, ‘xyz’]

TUPLE → **Similar to the list**, however, tuples are **enclosed within parenthesis** (‘abc’, 786, 2.23, ‘john’)

DICTIONARY → **Dictionaries** consist of key-value pairs {1001: "John", 1002: "Jane”}

SET → Collection of unordered data type : iterable, mutable/changeable, without duplicacy

SEQUENCE Type

- **NUMBERS** → **Numeric values** (e.g., Integers, real numbers, complex)
- **STRING** → **Contiguous set** of *characters* in quotation marks (‘Hello’)
- **LIST** → **Compound data type** : composed of *items* in square bracket[], separated by commas [‘abc’, 5, 10.5, ‘xyz’]
- **TUPLE** → **Similar to the list**, however, tuples are **enclosed within parenthesis** (‘abc’, 786, 2.23, ‘john’)
- **DICTIONARY** → **Dictionaries** consist of key-value pairs {1001: "John", 1002: "Jane”}
- **SET** → Collection of unordered data type : iterable, mutable/changeable, without duplicacy
- **Boolean** → True (T) or False (F)

Program Flow Control in Python

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and Assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

Program Flow Control in Python

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and Assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

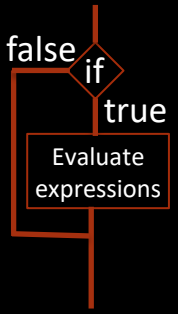
Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

Decision making



Program Flow Control in Python

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and Assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and Import
- Object Oriented Programming

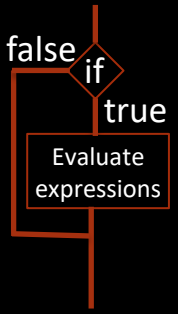
Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

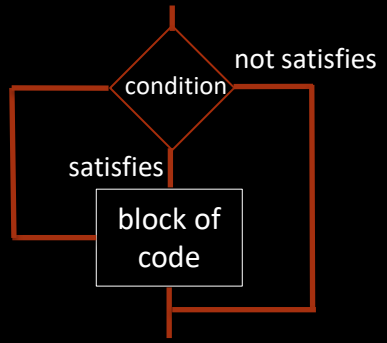
Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

Decision making



Loops: sequentially evaluation of **block of code** *multiple times*



Program Flow Control in Python

Introduction (an overview to course)

- Getting started "Hello World"
- Keywords and identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and import
- Object Oriented Programming

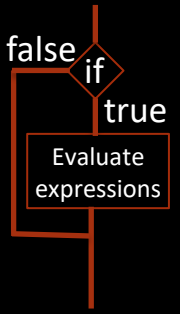
Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

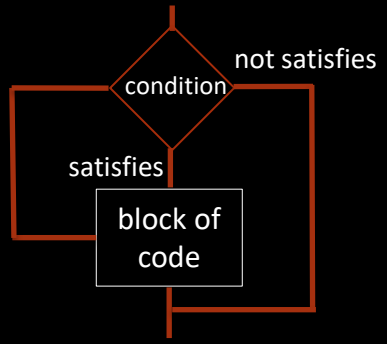
Program Flow Control in Python

- *If* statement
- Elif statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

Decision making



Loops: sequentially evaluation of **block of code** *multiple times*



Control statements: ***break***, **continue**, **pass**

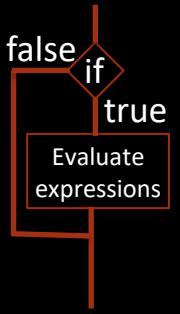
Program Flow Control in Python

Introduction (an overview to course)
- Getting started "Hello World"
- Keywords and identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and import
- Object Oriented Programming

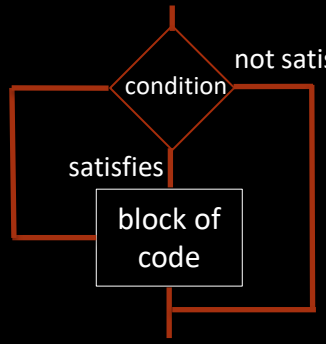
Data types
- Numbers
- List
- Tuple
- String
- Dictionary
- Set

Program Flow Control in Python
- *If* statement
- Elif statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

Decision making



Loops: sequentially evaluation of **block of code** *multiple times*



Control statements: ***break***, **continue**, **pass**

operators

Program Flow Control in Python

Introduction (an overview to course)

- Getting Started "Hello World"
- Keywords and Identifiers
- Comments and Statements
- Variables and assignments
- Data types
- Flow control
- Methods and Functions
- Reading and Writing files
- Modules and import
- Object Oriented Programming

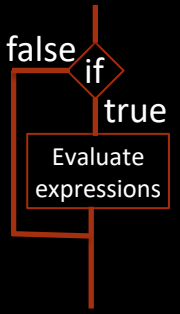
Data types

- Numbers
- List
- Tuple
- String
- Dictionary
- Set

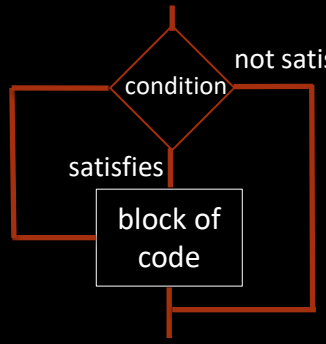
Program Flow Control in Python

- *If* Statement
- Elif Statement
- More on if, elif and else
- *For* loop
- *While* loop
- Useful operators

Decision making



Loops: sequentially evaluation of **block of code** *multiple times*



Control statements: ***break***, **continue**, **pass**

operators

Arithmetic operators (+, - , * , / , % , ...)
Assignment operators (=, +=, -=,*=,.....)
Comparison operators (==, !=, >, <,.....)
Logical operators (and, or, not),.....

Methods and Functions

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib
(basic uses)

Methods and Functions

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib
(basic uses)

A Function is a block of reusable code, which can be called to perform certain actions (*multiple times*).

Methods and Functions

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib
(basic uses)

A Function is a block of reusable code, which can be called to perform certain actions (*multiple times*).

Two types:

Built-in functions e.g., print ()

user-defined functions

Reading and Writing files

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

- Files and directories in Python
- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

- Standard Python library
  - Datetime module
  - Math and Random module
  - Generators and decorators
  - NumPy, Pandas, Matplotlib (basic uses)

Reading and Writing files

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

- Files and directories in Python
- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

- Standard Python library
  - Datetime module
  - Math and Random module
  - Generators and decorators
  - NumPy, Pandas, Matplotlib (basic uses)

While working with computers, users interact with computer by giving some inputs and obtaining results.

In this course, we will cover how to read data from files, and write to files

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib (basic uses)

Reading and Writing files

While working with computers, users interact with computer by giving some inputs and obtaining results.

In this course, we will cover how to read data from files, and write to files

Moreover, we will work with data formatted in CSV (comma-separated values) or JSON (JavaScript object notation),

two common formats that modules in Python's standard library handles.

Modules and Import

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib
(basic uses)

Modules and Import

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib
(basic uses)

- Python modules/libraries are similar to functions

Modules and Import

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib
(basic uses)

- Python modules/libraries are similar to functions
- Functions are useful if one needs to rereat set of actions within same program.

Modules and Import

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib
(basic uses)

- Python modules/libraries are similar to functions
- Functions are useful if one needs to rereat set of actions within same program.
- However, if one would like to *use same set of actions in multiple programs*, one can *create a dedicated library*, which can be accessed by all programs, and in python it is termed as Modules.

Modules and Import

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib (basic uses)

- Python modules/libraries are similar to functions
- Functions are useful if one needs to rereat set of actions within same program.
- However, if one would like to *use same set of actions in multiple programs*, one can *create a dedicated library*, which can be accessed by all programs, and in python it is termed as Modules.
- *Python comes with a large number of modules* and has several advantages, which will be discussed in dedicated lecture on modules

Modules and Import

Methods and Functions

- Defining a function
- Flow when calling a function
- Parameters and arguments
- Global/local
- Functions calling functions

Reading/Writing Files

Files and directories in Python

- Reading from a file
- Parsing data
- Printing data to external file

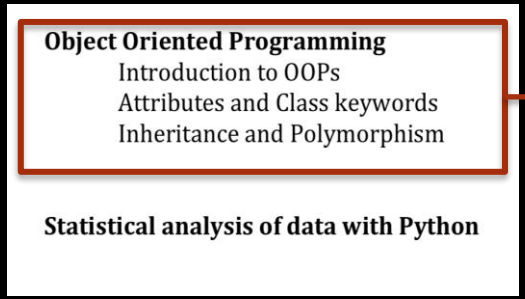
Modules and Import

Standard Python library

Datetime module
Math and Random module
Generators and decorators
NumPy, Pandas, Matplotlib (basic uses)

- Python modules/libraries are similar to functions
- Functions are useful if one needs to rereat set of actions within same program.
- However, if one would like to *use same set of actions in multiple programs*, one can *create a dedicated library*, which can be accessed by all programs, and in python it is termed as Modules.
- *Python comes with a large number of modules* and has several advantages, which will be discussed in dedicated lecture on modules
- Apart from Python standard library, a large number of modules available on Python package index (https://pypi.org), can be accessed using *import*

Object Oriented Programming



This is a bit advanced topic. In the framework of this course, We will learn the basic concepts following the simple implementation

Python scope and setup

There are several editors/IDS (Integrated Development Environment)-

Python scope and setup

There are several editors/IDS (Integrated Development Environment)-

>>>IDLE

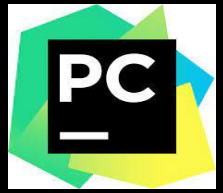
editor with light weight environment

Visual studio

Sublime Text

ATOM

PyCharm



Python IDE

Python scope and setup

There are several editors/IDS (Integrated Development Environment)-

>>>IDLE

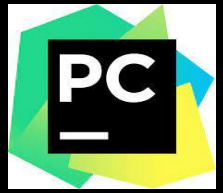
editor with light weight environment

Visual studio

Sublime Text

ATOM

PyCharm



Python IDE

Google Colab

Google colab /Jupyter is a web application that lets you run:

- ✓ Live codes,
- ✓ Embedded visualizations,
- ✓ Explanatory texts
- ✓ much more functionality

all in one place.

https://colab.research.google.com/

Python scope and setup

There are several editors/IDS (Integrated Development Environment)-

>>>IDLE

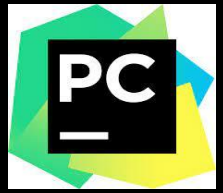
editor with light weight environment

Visual studio

Sublime Text

ATOM

PyCharm



Python IDE

Google Colab

Google colab /Jupyter is a web application that lets you run:

- ✓ Live codes,
- ✓ Embedded visualizations,
- ✓ Explanatory texts
- ✓ much more functionality

all in one place.

https://colab.research.google.com/

IP[y]: Ipython Interactive Computing

https://ipython.org

Google Colab

- Google Colab is free, cloud-based Jupyter notebook environment.
- Offers built-in support for GPUs, powerful tool for computation.
- Great for data science and machine learning
- Installation : not needed (Go to https://colab.research.google.com and work)

Google Colab

- Google Colab is free, cloud-based Jupyter notebook environment.
- Offers built-in support for GPUs, powerful tool for computation.
- Great for data science and machine learning
- Installation : not needed (Go to https://colab.research.google.com and work)

IPython

- IPython, an *interactive shell* for Python (for quick computation and testing)
- Provides rich toolkit for interactive computing.
- Allow shell syntax, tab completion, and *magic commands*
- To use IPython use ‘ pip install IPython ’ (in terminal or command prompt)

Comparison between Google Colab and IPython

Environment : Colab is cloud-based while IPython is local

Setup/install : Colab requires no installation, Ipython needs pip

GPU support : Colab has GPT/TPU (limited access), IPython doesn't.

File Handling : Colab integrates with Google drive; Ipython uses local files

Collaboration : Colab supports real-time collaboration; IPython doesn't

Media support: Colab supports inline media better than Ipython

Magic commands: IPython supports magic commands, Google Colab doesn't

Tutorials / Hands-On sessions

satyam.tiwari@doctoral.uj.edu.pl