

TITLE **Edition**

Lecture PPT

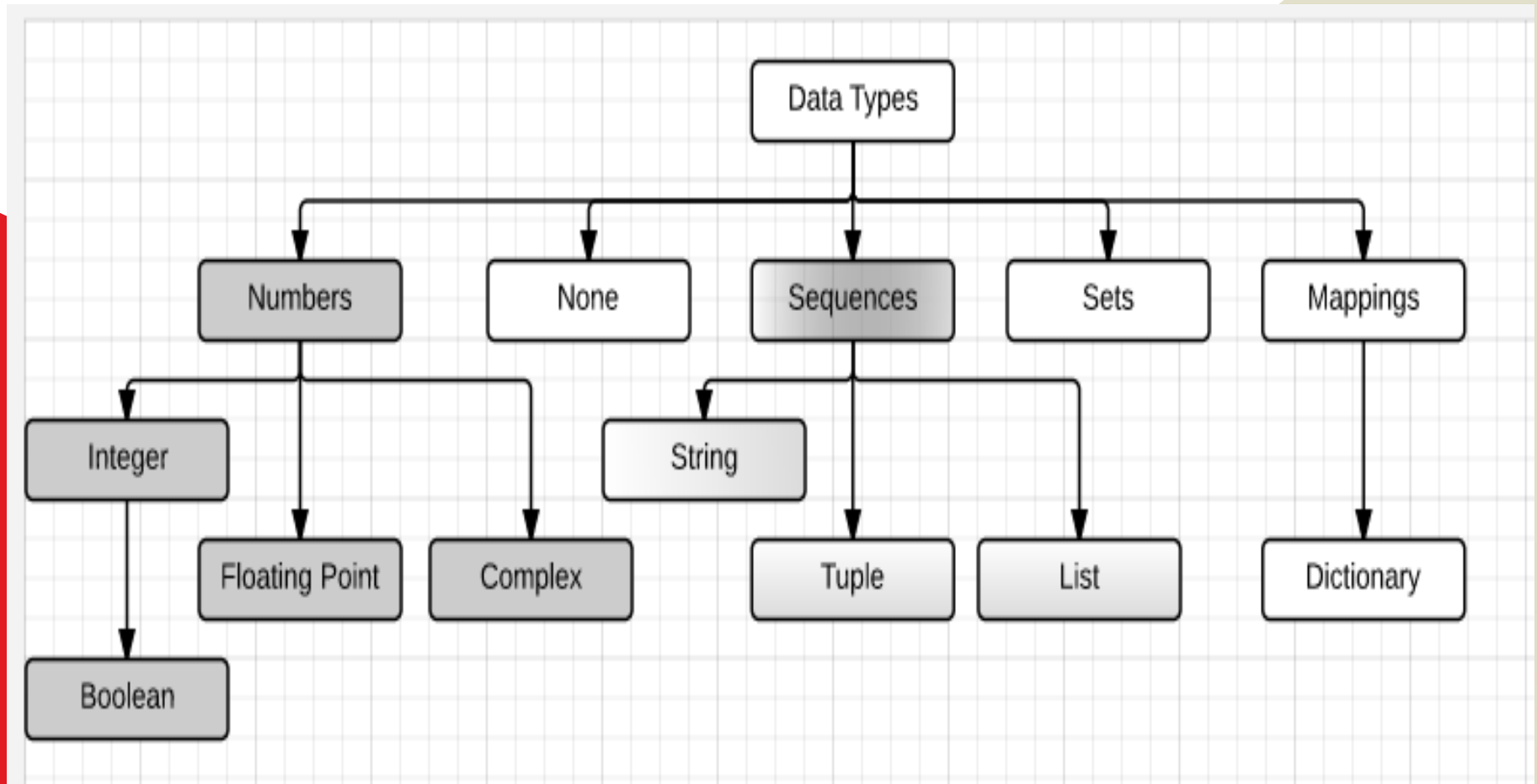
Learning objectives

- Understand the various “built-in” data types and concepts of “sequence” and “containers”
- Comprehend the concept of “mutability” and related concept of object ID.
- Describe “type casting” and the various “implicit” and “explicit” ways of casting.
- Provide “input” to a python script.
- Explain what are modules and packages.
- Understand python strings, “binary literals” and “boolean” types.

Built-in Data Types

- Python has what are known as “built in” data types.
- These built in data are of following 2 types:
 - - Simple: They includes int, float, complex, bool, str etc
 - - Compound: These data types act as “containers” of other data types. They includes lists, tuples, dictionaries, strings, sets and frozensets.

Tree structure of built in data types



Numbers (including integers, floating points, complex and bool)

- In Python, numbers are represented by data types integers (int), floating points (float), complex and boolean (bool). Some important characteristics of numbers in Python are as follows:
- They store numeric values.
- They are immutable data types.
- Numeric data types can be 'literals' or 'variables'. As with all others, numerics (whether literals or variables) are objects in Python.
- Number data types are created when you assign a value to them.

int (signed integers):

Some noteworthy points regarding integers in Python are as follows:-

- **Python 2.x had two separate types for integers, that is, int and long.**
- **Python 3.x has just one integer type, that is, int, which behaves mostly like the old long type from Python 2.x.**
- **The maximum value of the int data type in Python 2.x was limited by sys.maxint. This may be different for different platforms. (But usually it is 232).**
- **But the sys.maxint constant is no more there in Python 3.x. So, there is no longer a limit to the value of integers.**

float (floating point real values):

- In Python, a floating point number is written with a decimal point.
- The part to the left represents the integer part of the number and the part to the right represents the fraction part of the number.
- Floating point numbers in Python can also be represented in scientific notation using E or e.
- Note that, here 'e' stands for base 10 (Not Euler number).
- So you can represent a floating point number in scientific notation as say $3.33 \text{ e } -2$ which is the same as 3.33×10^{-2} .

complex (complex numbers)

In Python, a complex number has the following characteristics:-

- **A complex number consists of an “ordered pair” of numbers of type $x + yj$, where x and y may be int or float.**
- **Here, x represents the “real” part of the number while “ y ” represents the “imaginary” part of the number.**
- **The term “ordered pair” means that the order of the number x and y matter. So $x + yj$ is different from $y + xj$.**

bool

- **Boolean values in Python are True and False (Not true and false, that is, first letter must be capital).**
- **The Boolean type in Python is called bool (Not Bool nor boolean nor Boolean).**

Strings

Some important characteristics of string data types are as follows:

- **Strings in Python are a “sequence of characters enclosed by quotation marks”. You can use pairs of either single or double quotes to mark the beginning and end of a string.**
- **Since a string is a sequence, you can use an “index” to access its individual characters.**
- **All index in Python start from 0. This means, the first character in a string will have index 0. If there are “n” characters in a string, then the index of the last item will be n-1.**
- **You can use the slice operator “[m : n]” with indexes m and n.**
- **You can use the plus (+) operator for “concatenation” of two strings.**
- **You can use the asterisk (*) operator to “multiply” a string with a positive integer. Here, “multiply” means “repetition”.**

Lists

- **A list in Python is a sequenced container.**
- **What does 'container' mean? A container in Python is an object which can contain other objects. Hence, a list can contain any valid python object, such as strings, numbers or even other lists.**
- **What does sequence mean? In a sequence in Python, each item is identified by an index, which starts from 0. Hence, if a list in Python has n items, then the sequence of first item is 0 and the sequence of the nth item is n-1.**
- **A list contains items separated by commas and enclosed within square brackets ([]).**

Tuples

Some important aspects of the tuple data type are as follows:

- **Tuples are also “sequences” and are very similar to lists.**
- **A tuple has a number items/ objects/ values separated by commas.**
- **However, while lists are enclosed within “square brackets”, tuples are enclosed within “parentheses”.**
- **The elements of a list can be changed. So, lists are “mutable”. However, the elements of a tuple cannot be changed. So, tuples are “immutable”.**
- **You can think of a tuple as a “read-only” list.**

Dictionary - 1

Some important points regarding dictionary are as follows:

- It has “unordered” key-value pairs.
- A dictionary is a container, which contains other objects. So, you cannot “access” the items of a dictionary through an “index” but you can access the items of a dictionary through its “keys”.
- So, a dictionary has keys and for each key there is a value, which can be an object of any type.
- Just like in maths you have a function definition as $y = f(x)$, dictionaries are something similar. You can think of x as keys and y as their corresponding values.
- Just like in a mathematical function for one key there can be only one value (Though the reverse need not be true, that is, for the same value there can be different keys or to put it differently, different keys may have the same values).

Dictionarys -2

- A dictionary is a ‘mapping’ between x and y or between the keys and its values.
- Dictionarys are enclosed by curly braces { } and values can be assigned and accessed using square braces [].
- The keys in a dictionary must be unique. The key value pairs are separated by a colon, that is, :.
- The keys of a dictionary must be of an “immutable data type” such as strings, numbers, or tuples.

Sets -1

- A set contains an unordered collection of immutable and unique objects.
- Sets, unlike lists or tuples, cannot have multiple occurrences of the same element. There are three important words in this definition:
- A set is a collection.
- It is unordered.
- The elements must be unique.

Sets -2

- You can think of a set in Python as a dictionary with no value, that is, a dictionary which only has keys.
- Remember, a dictionary uses curly braces and has a pair comprising a key and a value.
- Think of a set as a data type, which has no values but has only keys.
- Just like a dictionary, the items of a set can be enclosed in curly brackets, but they must all be unique.
- In case there are duplicate items, the Python interpreter will simply keep only one instance of the duplicate items.

Some other aspects about Python Sets

- **With sets you cannot do “sequence-like behavior”, such as indexing and slicing.**
- **In Python, sets cannot contain mutable elements, such as lists or dictionaries.**
- **Please note that sets are themselves mutable (This means, elements can be added or removed from a set).**
- **However, the items which comprise a set must be immutable, that is, these items cannot be changed.**
- **Since lists and dictionaries in Python can be changed, they cannot be items of a set. But you can “convert” a list or a dictionary to a Python set using some operators.**

None in Python

The following points regarding the data type “None” are relevant:

- **“None” is used to define a “null” value or “no value”.**
- **None does not mean either “0” or “an empty container”, such as an empty string.**
- **None also does not mean False.**
- **None does not mean undefined.**
- **None data type is a real data type, which occupies space in memory and can be assigned to a variable just as any other data type.**
- **In many programming languages, such as Java/ C++, the keyword “null” is used rather than None.**

Mutable vs Immutable

Mutable—Internal state of the object is changeable.

Immutable— Internal state of the object cannot be modified.



Explicit versus implicit type casting

The type casting may be explicit or implicit.

- In explicit type casting, the programmer explicitly converts or casts one type of data into another.
- In implicit type casting, on the other hand, the interpreter implicitly (that is, without explicit instructions from the programmer) converts or casts one type of data into another.

Mixed arithmetic, “narrow” and “wide” types

- Python supports mixed type of arithmetic as long as the data types, which are linked together by operators are capable of being converted to other data types.
- For instance, Python does not allow something like `'Hello' + 1` whereas it is permitted in C++. C++ would convert , that is, cast the 1 into a string and then concatenate it to `'Hello'`.
- Why does Python not allow it? Because as per the Zen of Python, “Explicit is better than implicit” meaning that the programmer should do these things knowingly, that is, explicitly rather than unknowingly done by the interpreter leading to bugs.
- Further, a type casting may be narrow or wide. If casting leads to loss of data, it is ‘narrow’. For instance, casting a float to an int leads to loss of fraction portion so it is ‘narrow’ casting. But casting an integer to a float is ‘wide’.

Some important Boolean type conversions

- **Empty string is mapped to False.**
- **Non-empty string is mapped to True.**
- **Integer 0 is mapped to False.**
- **Every non-zero integer (Including negatives) is mapped to True.**

The following are considered False in Python:-

- **None**
- **False**
- **Zero of any numeric type, for instance,**
 - - Numeric integer $\rightarrow 0$,
 - - Numeric float $\rightarrow 0.0$,
 - - Numeric complex $\rightarrow 0j$.
- **Any empty sequence, for instance,**
 - - Empty string $\rightarrow ''$,
 - - Empty tuple $\rightarrow ()$,
 - - Empty list $\rightarrow []$.
- **Any empty mapping, for instance,**
 - - Empty dictionary $\rightarrow \{\}$.

What is runnable code?

- A runnable code is that piece of code, which will be executed upon loading of the module.
- For instance, if the module has a `print()` function, which is executed when the module is loaded, then this `print()` function is an example of runnable code.
- On the other hand, if the module has a class definition, then such code is not executed till such time such an instance of a class, that is, an object is created. So, a class definition in a module is not runnable code.
- Similarly, a method definition inside a module is not executed when the module is loaded, because functions are executed when they are called and not when the module is loaded.
- Therefore, function definitions in modules are also examples of non-runnable codes.

Modules, packages and libraries

- **Module**— A module is simply a file with `.py` extension, in which you may put your script/ code.
- **Package**— A python package is a “directory of python modules”. It is a “name space”, which contains “modules” or even other packages”. In addition, a package contains an additional file named `__init__.py`. This file exists just below the directory. This means, it is not in one of the modules but rather directly under the directory, which is the package. (Concepts of “name space” and `__init__.py` are explained later)
- **Library**— In Python the term “library” does not have a “conceptually different” meaning than “package”. Loosely, you may say that a Python package is a “directory” of Python modules. (Which, in a way, is same as a package).

Y

X




Because learning changes everything.®

Thank You!

For any queries or feedback contact us at:

 support.india@mheducation.com

 1800-103-5875

 www.mheducation.co.in

in

