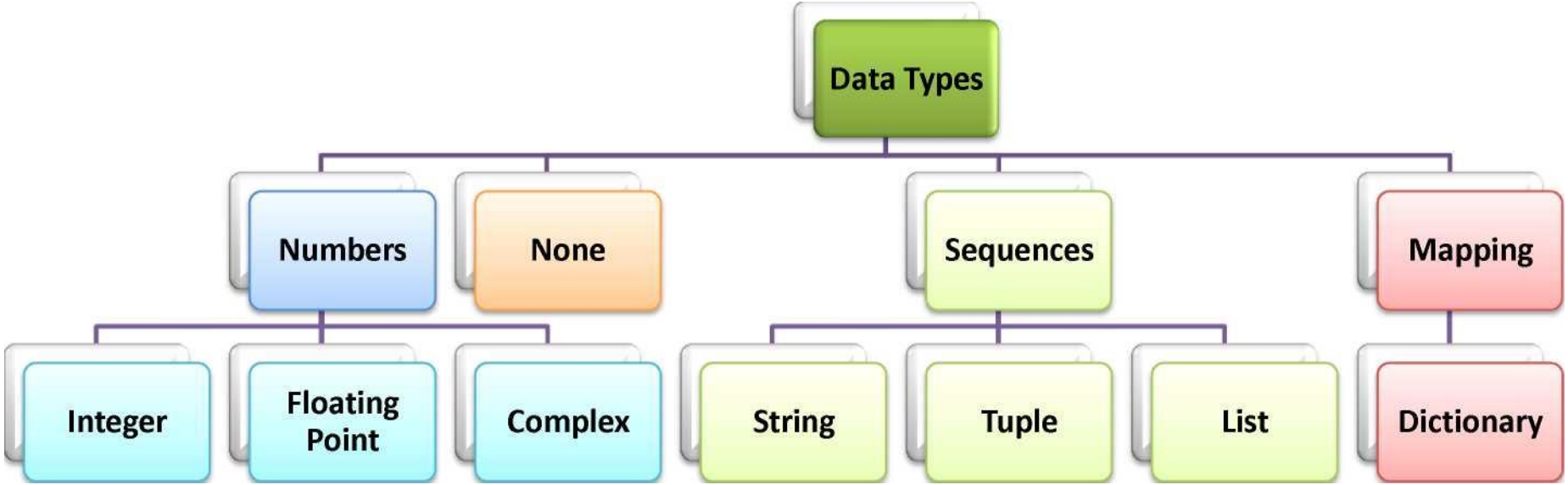
Data Types in Python

- **Data types are used to identify the type of data and set of valid operations which can be performed on it.**
- ***Python has following data types:***
  - **Numbers(integer(whole no), floating(number with decimal)**
  - **String**
  - **List**
  - **Tuple**
  - **Dictionary**


Data Types
Numbers
None
Sequences
Mapping
Integer
Floating Point
Complex
String
Tuple
List
Dictionary

Numbers

Data types offered by python to store and process different type of numeric data

Integer

- Signed Integer
- Booleans

Floating-point Numbers

Complex Numbers

Integers

Integers

- int
- Stores any integer(signed or unsigned) (big or small)
- e.g. -123, 1234

Booleans

- True or False
- 1 or 0

Example of integer datatype in Python

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x=10
>>> print(x)
10
>>> type(x)
<class 'int'>
>>> |
```

Ln: 8 Col: 4

Examples of Boolean Expression in Python

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a=(5>10)
>>> print(a)
False
>>> b=(5<10)
>>> print(b)
True
>>>
```

Ln: 9 Col: 4

Floating Point Numbers

- float
- Represents double precision floating point numbers (15 digit precision)
- e.g. 12.345

Examples of Floating Point Numbers in Python

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x=12.34
>>> print(x)
12.34
>>> type(x)
<class 'float'>
>>> |
```

Ln: 8 Col: 4

Complex Numbers

❖Python represents Complex Numbers in the form A + Bj

❖Where, A & B are Real Numbers and j = V-1 (imaginary)

❖*Real* and *imaginary* part are internally represented as a pair of floating point numbers (float)

A is real part of the complex number Z = A + Bj

python™

imaginary

- Bj is Imaginary part of the complex number Z = A + Bj

Examples of Complex Numbers in Python

```
»> x=5+4j
»> print(x.real, x.imag)
5 . 0 4 . 0
»> y=2-2j
»> print (y. real, y.imag)
2 . 0 2 . 0
»> z=x+y
»> print (z)
( 7 + 2 j )
»> type(z)
<class 'complex'>
```

Range of Python Numbers

Integers	Unlimited range, subject to available (virtual) memory only
Booleans	• True (1) , False (0)
Floating Point Numbers	Unlimited range, subject to available (virtual) memory on machine architecture
Complex Numbers	Same as floating point numbers (Real and Imaginary Parts are represented as floats)

Strings

- In Python 3.x, all strings are a sequence of pure Unicode* Characters and each character can be individually accessed using its index
- A string can hold any known character like letters, numbers, special characters etc., of any known scripted language
- E.g. "abcd", "$@&%", '???', "1234", "apy”,

*Unicode is a system designed to represent every character from every language

Example of Strings in Python

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> Language="Python"
>>> print(Language)
Python
>>> type(Language)
<class 'str'>
>>> |
```

Ln: 8 Col: 4

Lists

- A list in Python is represented as a group of comma-separated values of any datatype enclosed in *square brackets*.
- E.g. [1,2,3,4,5], ['Ankit', 101, 90.5], ['a', 'e', 'i', 'o', 'u']
- Elements in a List can be individually accessed using its index

List	a	e	i	o	u
Index Value	0	1	2	3	4

Example of Lists in Python

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> A=['a','e','i','o','u']
>>> print(A)
['a', 'e', 'i', 'o', 'u']
>>> type(A)
<class 'list'>
>>> A[2]
'i'
>>> A[0]
'a'
>>> |
```

Ln: 12 Col: 4

Tuples

- A tuple in Python is represented as a group of comma-separated values of any datatype enclosed within *parentheses*.
- E.g. A=(1,2,3,4,5), R=('Ankit', 101, 90.5), V=('a', 'e', 'i', 'o', 'u')
- Elements in a Tuple can be individually accessed using its index (Negative or Positive)

Positive Index	0	1	2	3	4	5
Tuple	1	2	3	a	b	c
Negative Index	-6	-5	-4	-3	-2	-1

Example of Tuples in Python

76 Python Shell

File Edit Shell Debug Options Windows Help

```
Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit (Intel)] on Win32
Type "copyright", "credits" or "license()" for more information.
»> tupl=(1,2,3,4)
»> print (tupl)
(1, 2, 3, 4)
»> type(tupl)
<class 'tuple'>
»> tup2=()
»> print (tup2)
»> tup3=(1,2,3,'a','b','c')
»> print (tup3)
(1, 2, 3, 'a', 'b',    'c')
»> tup3[4]
'b'
»> tup3[-2]
'b'  "

»>
```

Dictionaries

The dictionary in Python is an unordered set of comma-separated key:value pair enclosed within *curly braces*.

E.g. vowels ={'a':1, 'e':2, 'i':3, 'o':4, 'u':5}

Where, 'a', 'e', 'i', 'o', 'u' are the keys of dictionary vowels & 1,2,3,4,5 are the values for these keys respectively

Key: Value Pair => 'a' : 1

Example of Dictionaries in Python

Python Shell

File Edit Shell Debug Options Windows Help

```
Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit (I
ntel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> d={}
>>> print(d)
{}
>>> d1={1:'One',2:'Two',3:'Three'}
>>> print(d1)
{1: 'One', 2: 'Two', 3: 'Three'}
>>> type(d1)
<class 'dict'>
>>> d1[2]
'Two'
>>> |
```

Ln: 13 Col: 4

Mutable and Immutable Data type

A **mutable data type** can change its state or contents

Immutable data type cannot change its state or contents .

Mutable data type:
list, dict, byte array

Immutable data type:
int, float, complex, string, tuple, bytes , set

Mutability can be checked with id() method.

```
x=10
print(id(x)) #id() is use to find the address of any variable

x=20

print(id(x))

#id of both print statement is different as integer is immutable
```

Boolean In Python

It is used to store two possible values either true or false

e.g.

```
str="comp sc"
b=str.isupper() # test if string contains upper case
print(b)
```

Output :-

False

Set In Python

It is an unordered collection of unique and immutable (which cannot be modified)items.

e.g.

```
set1={11,22,33,22}
print(set1)
```

Output :-

{33, 11, 22}

Type conversion

The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion.

Python has two types of type conversion.

- Implicit Type Conversion
- Explicit Type Conversion

Implicit Type Conversion:

In Implicit type conversion, Python automatically converts one data type to another data type. This process doesn't need any user involvement.

e.g.

```
n = 12
f = 10.23
num_new = n + f
print("datatype of n:",type(n))
print("datatype of f:",type(f))
print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))
```

Explicit Type Conversion:

In Explicit Type Conversion, users convert the data type of an object to required data type. We use the predefined functions like int(),float(),str() etc.

Type Conversion of Integer

int() function converts any data type to integer.

e.g.

```
a = "101" # string
b=int(a) # converts string data type to integer.
c=int(122.4) # converts float data type to integer.
print(b)
print(c)
```

Output :-

```
101
122
```

Type Conversion of Floating point numbers

Floating point numbers is a positive or negative real numbers with a decimal point

float() function converts any data type to floating point number.

e.g.

```
a='301.4' #string
b=float(a) #converts string data type to floating point number.
c=float(121) #converts integer data type to floating point number.
print(b)
print(c)
```

Output :-

```
301.4
121.0
```

Explicit Type Conversion

e.g.

```
n = 12
s = "45"
print("Data type of n:",type(n))
print("Data type of s before Type Casting:",type(s))
s = int(s)
print("Data type of s after Type Casting:",type(s))
num_sum = n + s
print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```