Module - 1

Python Fundamentals

Data Types and Operators

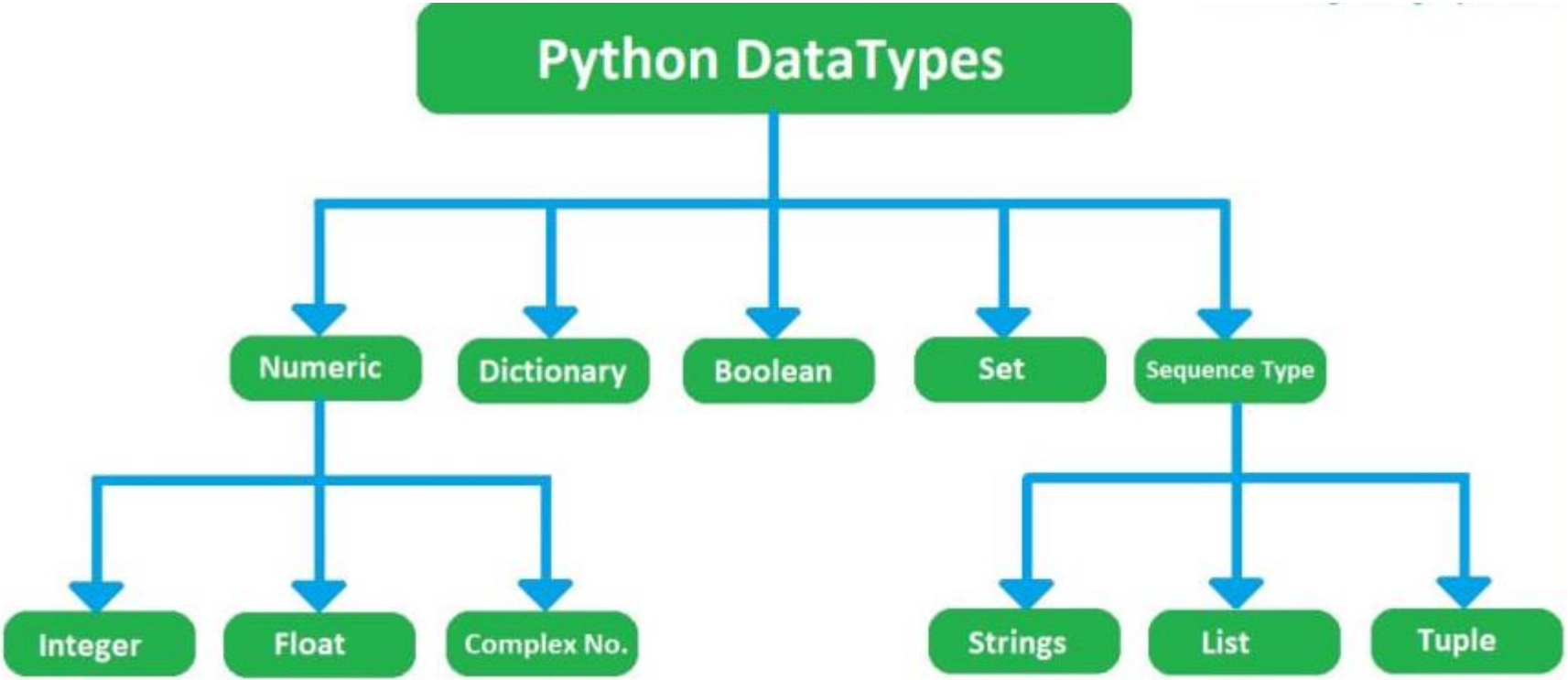
Presentation By

Mohammed Tahir Mirji

MTech CS

CONTENTS

Introduction to Datatypes in python



What is data type?

In computer science and computer programming, a data type or simply type is an attribute of data which tells the compiler or interpreter how the programmer intends to use the data.

What are Integers (sub type of Numeric)

This value is represented by int class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.

```
# Python program to
# demonstrate numeric value

a = 5
print("Type of a: ", type(a))
```

Note – type() function is used to determine the type of data type.

What are Floats

This value is represented by float class. It is a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.

```
# Python program to
# demonstrate numeric value
b = 5.0
print("\nType of b: ", type(b))
```

Note – type() function is used to determine the type of data type.

What are Complex Numbers?

Complex number is represented by complex class.
It is specified as ***(real part) + (imaginary part)j***.

For example – 2+3j

```
# Python program to
# demonstrate complex type value

c = 2 + 4j
print("\nType of c: ", type(c))
```

Note – type() function is used to determine the type of data type.

What are Boolean

Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.

Note – True and False with capital ‘T’ and ‘F’ are valid booleans otherwise python will throw an error.

```
# Python program to
# demonstrate boolean type

print(type(True))
print(type(False))

print(type(true))
```

What are Strings

In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one.

Creating String
Strings in Python can be created using single quotes or double quotes or even triple quotes.

```
# Python Program for
# Creation of String

# Creating a String
# with single Quotes

String1 = 'Welcome to the Python World'
print("String with the use of Single Quotes: ")
print(String1)

# Creating a String
# with double Quotes

String1 = "I'm a programmer"
print("\nString with the use of Double Quotes: ")
print(String1)
print(type(String1))
```

What are Lists

Lists are just like the arrays, declared in other languages which is a ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

Creating List

Lists in Python can be created by just placing the sequence inside the square brackets[].

```
# Python program to demonstrate
# Creation of List
# Creating a List

List = []
print("Initial blank List: ")
print(List)

# Creating a List with
# the use of a String

List = [“Python World”]
print("\nList with the use of String: ")
print(List)

# Creating a List with
# the use of multiple values

List = [“Python", "For", “World"]
print("\nList containing multiple values: ")
print(List[0])
print(List[2])
```

What are Tuples

Just like list, tuple is also an ordered collection of Python objects. The only difference between tuple and list is that tuples are immutable i.e. tuples cannot be modified after it is created. It is represented by tuple class.

Creating Tuple

In Python, tuples are created by placing a sequence of values separated by ‘comma’ with or without the use of parentheses for grouping of the data sequence. Tuples can contain any number of elements and of any datatype (like strings, integers, list, etc.).

Note: Tuples can also be created with a single element, but it is a bit tricky. Having one element in the parentheses is not sufficient, there must be a trailing ‘comma’ to make it a tuple.

Accessing elements of Tuple

In order to access the tuple items refer to the index number. Use the index operator [ ] to access an item in a tuple. The index must be an integer. Nested tuples are accessed using nested indexing.

Ex: print(Tuple1[0])

```
# Creating a Tuple with
# the use of Strings

Tuple1 = (‘Python', 'For')
print("\n Tuple with the use of String: ")
print(Tuple1)

# Creating a Tuple
# with nested tuples

Tuple1 = (0, 1, 2, 3)
Tuple2 = ('python', ‘world')
Tuple3 = (Tuple1, Tuple2)
print("\nTuple with nested tuples: ")
print(Tuple3)
```

What are Sets

In Python, Set is an unordered collection of data type that is iterable, **mutable** and has **no duplicate elements**. The order of elements in a set is undefined though it may consist of various elements.

Creating Sets

Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, separated by 'comma'. Type of elements in a set need not be the same, various mixed-up data type values can also be passed to the set.

Accessing elements of Sets

Set items cannot be accessed by referring to an index, since sets are unordered the items has no index. But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

```
# Python program to demonstrate
# Creation of Set in Python

# Creating a Set

set1 = set()
print("Initial blank Set: ")
print(set1)

# Creating a Set with
# the use of a String

set1 = set("Python world")
print("\n Set with the use of String: ")
print(set1)

# Accessing element using
# for loop

print("\nElements of set: ")
for i in set1:
    print(i, end =" ")
```

What are Dictionaries

Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds **key:value pair.** Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon **:,** whereas each key is separated by a ‘comma’.

Creating Dictionary

In Python, a Dictionary can be created by placing a sequence of elements within curly {} braces, separated by ‘comma’. Values in a dictionary can be of any datatype and can be **duplicated**, whereas keys can’t be repeated and must be **immutable**. Dictionary can also be created by the built-in function **dict**(). An empty dictionary can be created by just placing it to curly braces{}.

Note – Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.

```
# Creating an empty Dictionary
Dict = {}
print("Empty Dictionary: ")
print(Dict)


# Creating a Dictionary

# with Integer Keys

Dict = {1: ‘My', 2: ‘Loving', 3: ‘Python'}

print("\nDictionary with the use of Integer Keys: ")

print(Dict)


# Creating a Dictionary

# with Mixed keys

Dict = {'Name': ‘Python', 1: [1, 2, 3, 4]}

print("\nDictionary with the use of Mixed Keys: ")

print(Dict)
```

What are Operators?

Operators are **special symbols in Python that carry out arithmetic or logical computation**. The value that the operator operates on is called the operand. Here, + is the operator that performs addition. 2 and 3 are the operands and 5 is the output of the operation.

- Arithmetic Operators.
- Relational Operators.
- Assignment Operators.
- Logical Operators.
- Membership Operators.
- Identity Operators.

What are Operators and operands?

OPERATORS: Are the special symbols. Eg- + , * , /, etc.

OPERAND: It is the value on which the operator is applied.

Ex: c = a + b

What are Arithmetic Operators?

Arithmetic operators are used to performing mathematical operations like addition, subtraction, multiplication, and division.

Operator	Description	Syntax
+	Addition: adds two operands	x + y
-	Subtraction: subtracts two operands	x - y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power: Returns first raised to power second	x ** y

Let’s work on Arithmetic Operators

```
# Examples of Arithmetic Operator
a = 9
b = 4

# Addition of numbers
add = a + b

# Subtraction of numbers
sub = a - b

# Multiplication of number
mul = a * b

# Division(float) of number
div1 = a / b

# Division(floor) of number
div2 = a // b

# Modulo of both number
mod = a % b

# Power
p = a ** b

# print results
print(add)
print(sub)
print(mul)
print(div1)
print(div2)
print(mod)
print(p)
```

What is Comparison Operators?

Comparison of Relational operators compares the values. It either returns True or False according to the condition.

Operator	Description	Syntax
>	Greater than: True if the left operand is greater than the right	x > y
<	Less than: True if the left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to – True if operands are not equal	x != y
>=	Greater than or equal to True if the left operand is greater than or equal to the right	x >= y
<=	Less than or equal to True if the left operand is less than or equal to the right	x <= y
is	x is the same as y	x is y
is not	x is not the same as y	x is not y

What is Comparison Operators?

```
# Examples of Relational Operators
a = 13
b = 33

# a > b is False
print(a > b)

# a < b is True
print(a < b)

# a == b is False
print(a == b)

# a != b is True
print(a != b)

# a >= b is False
print(a >= b)

# a <= b is True
print(a <= b)
```

What is Logical Operators?

Logical operators perform **Logical AND**, **Logical OR**, and **Logical NOT** operations. It is used to combine conditional statements.

Operator	Description	Syntax
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if the operand is false	not x

```
# Examples of Logical Operator
a = True
b = False

# Print a and b is False
print(a and b)

# Print a or b is True
print(a or b)

# Print not a is False
print(not a)
```

What is Bitwise Operators?

Bitwise operators act on bits and perform the bit-by-bit operations. These are used to operate on binary numbers.

Operator	Description	Syntax
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

What is Bitwise Operators?

```
# Examples of Bitwise operators
a = 10
b = 4
# Print bitwise AND operation
print(a & b)

# Print bitwise OR operation
print(a | b)

# Print bitwise NOT operation
print(~a)

# print bitwise XOR operation
print(a ^ b)

# print bitwise right shift operation
print(a >> 2)

# print bitwise left shift operation
print(a << 2)
```

What are Assignment Operators ?

Assignment operators are used to assign values to the variables.

Operator	Description	Syntax
=	Assign value of right side of expression to left side operand	x = y + z
+=	Add AND: Add right-side operand with left side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign the result to left operand	a%=b a=a%b

What are Assignment Operators ?

Assignment operators are used to assign values to the variables.

Operator	Description	Syntax
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b a=a//b
**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a**=b a=a**b
&=	Performs Bitwise AND on operands and assign value to left operand	a&=b a=a&b
|=	Performs Bitwise OR on operands and assign value to left operand	a|=b a=a|b
^=	Performs Bitwise xOR on operands and assign value to left operand	a^=b a=a^b
>>=	Performs Bitwise right shift on operands and assign value to left operand	a>>=b a=a>>b

What are Assignment Operators ?

Assignment operators are used to assign values to the variables.

```
# Examples of Assignment Operators
a = 10

# Assign value
b = a
print(b)

# Add and assign value
b += a
print(b)
```

```
# Subtract and assign value
b -= a
print(b)

# multiply and assign
b *= a
print(b)

# bitwise lishift operator
b <<= a
print(b)
```

What are Identity Operators?

is and is not are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal do not imply that they are identical.

is	True if the operands are identical
is not	True if the operands are not identical

#Example: Identity Operator

```
a = 10
b = 20
c = a

print(a is not b)
print(a is c)
```

What are Membership Operators?

in and not in are the membership operators; used to test whether a value or variable is in a sequence.

in	True if value is found in the sequence
not in	True if value is not found in the sequence

```
# Python program to illustrate
# not 'in' operator
x = 24
y = 20
list = [10, 20, 30, 40, 50]

if (x not in list):
        print("x is NOT present in given list")
else:
        print("x is present in given list")

if (y in list):
        print("y is present in given list")
else:
        print("y is NOT present in given list")
```

What is Operators precedence and associativity?

Operator precedence and associativity determine the priorities of the operator.

This is used in an expression with more than one operator with different precedence to determine which operation to perform first.

PRECEDENCE:

- P - Parentheses
- E - Exponentiation
- M - Multiplication (Multiplication and division have the same precedence)
- D - Division
- A - Addition (Addition and subtraction have the same precedence)
- S - Subtraction

What is Operators precedence and associativity?

```
# Examples of Operator Precedence

# Precedence of '+' & '*'
expr = 10 + 20 * 30
print(expr)
```

```
# Precedence of 'or' & 'and'
name = "Alex"
age = 0

if name == "Alex" or name == "John" and age >= 2:
    print("Hello! Welcome.")
else:
    print("Good Bye!!")
```

What is Operators associativity?

If an expression contains two or more operators with the same precedence then Operator Associativity is used to determine. It can either be Left to Right or from Right to Left.

Examples of Operator Associativity

```
# Left-right associativity
# 100 / 10 * 10 is calculated as
# (100 / 10) * 10 and not
# as 100 / (10 * 10)
print(100 / 10 * 10)

# Left-right associativity
# 5 - 2 + 3 is calculated as
# (5 - 2) + 3 and not
# as 5 - (2 + 3)
print(5 - 2 + 3)

# left-right associativity
print(5 - (2 + 3))

# right-left associativity
# 2 ** 3 ** 2 is calculated as
# 2 ** (3 ** 2) and not
# as (2 ** 3) ** 2
print(2 ** 3 ** 2)
```

Assignments of the Data Types and Operators

Refer Readme.md of github repository for list of assignments:

https://github.com/tahirmirji/ai_with_python_keonics

Thank You