Python Data Types and Operators

Learning Objectives

By the end of this lesson, you will be able to:

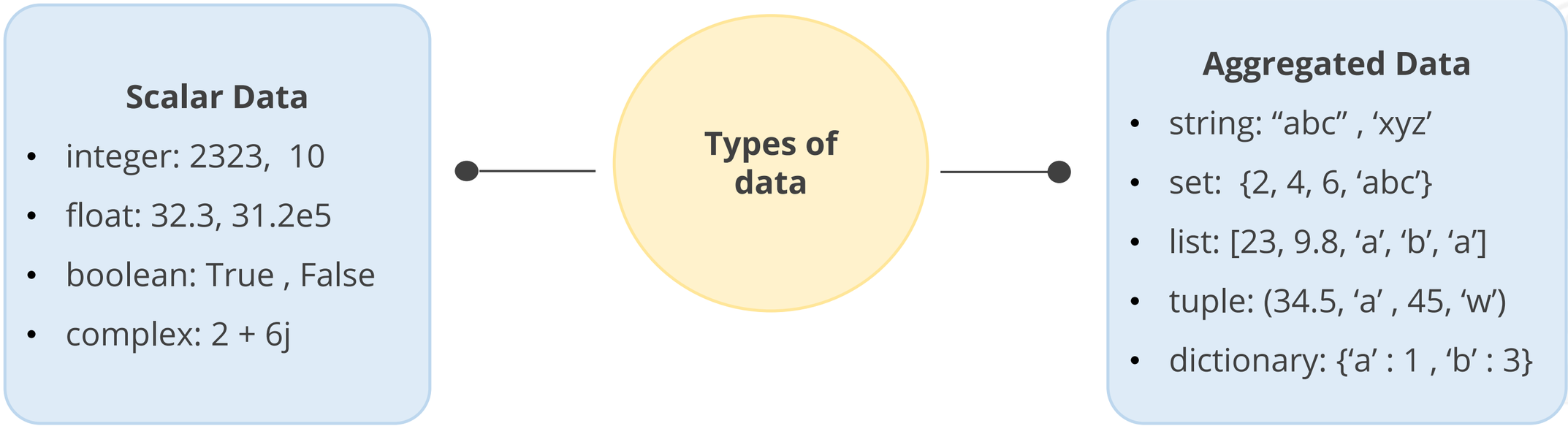
- Identify Python data types
- Learn data assignment in Python
- List the different operators in Python
- Work with string functions

DATA AND
ARTIFICIAL INTELLIGENCE

Data Types and Data Assignment

Data Types

Variables can store various kinds of data, each of which has a specific function.
Objects are used to represent data.



Note

The data type of any object can be checked by using the built-in function: *type()*.

Data Assignment

Python variables are references to objects, but actual data is contained in the objects.

```
x = 34
y = x
print('x = ', x, ' ; id of x: ', id(x))
print('y = ', y, ' ; id of x: ', id(y))
```

```
x =  34  ; id of x:  140210482584912
y =  34  ; id of x:  140210482584912
```

Here, *x* and *y* are pointing to the same memory location where 34 (an integer object) is stored.

```
y = 78
print('x = ', x, ' ; id of x: ', id(x))
print('y = ', y, ' ; id of x: ', id(y))
```

```
x =  34  ; id of x:  140210482584912
y =  78  ; id of x:  140210482774800
```

Here, *y* is pointing to a new integer object with 78 as a value, and *x* is pointing to the previous object with 34.

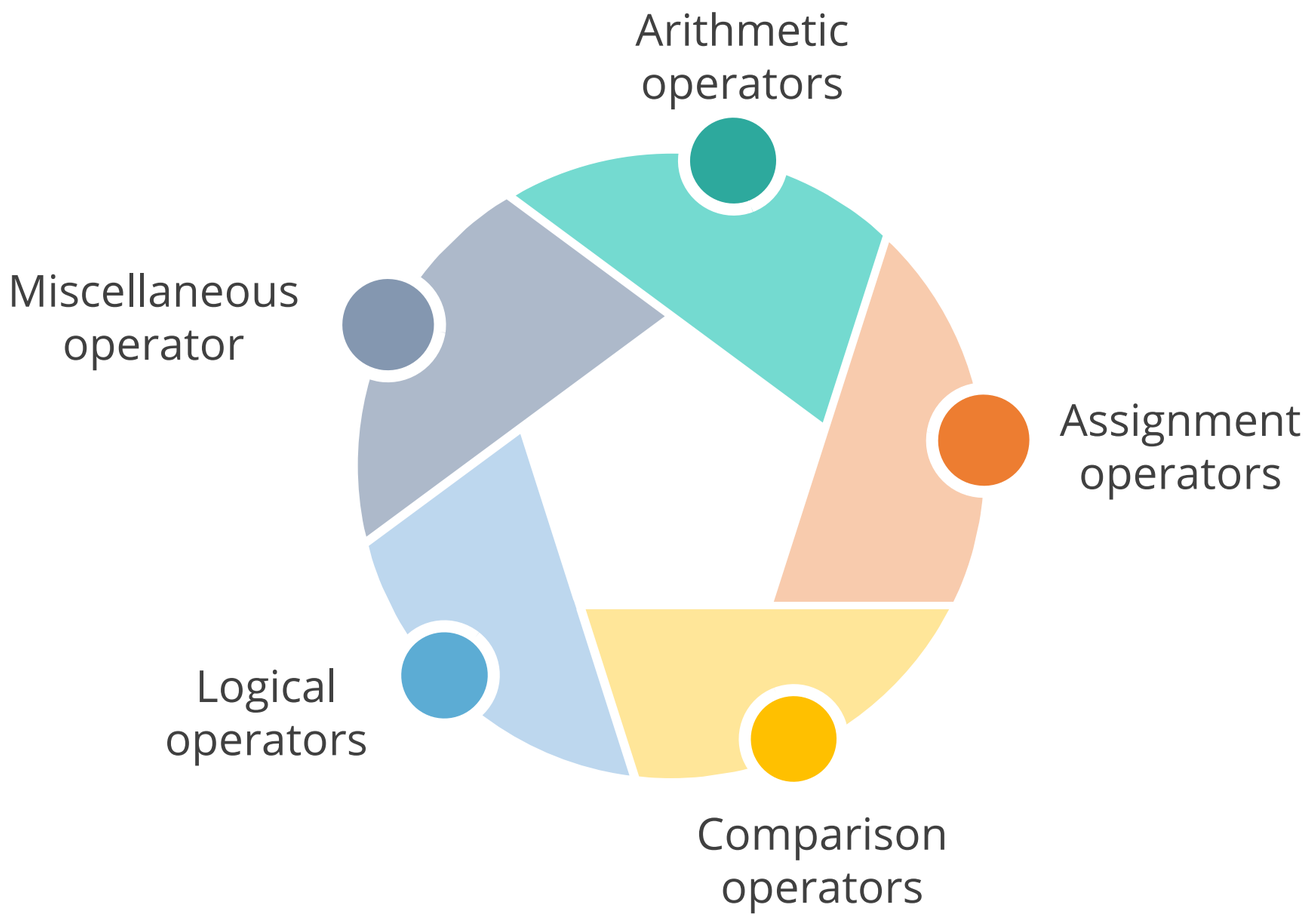
These references can be verified by using id() function

DATA AND
ARTIFICIAL INTELLIGENCE

Operators in Python

Operators

The different types of operators are given below:



Arithmetic Operators

Different arithmetic operators are given below with an example:

Operator	Description	Example
+	Addition	>> x = 12 ; y = 50 >> x + y 62
-	Subtraction	>> x = 45 ; y = 24 >> x - y 21
*	Multiplication	>> x = 50 ; y = 4 >> x * y 200
/	Division	>> x = 50 ; y = 4 >> x / y 12.5

Arithmetic Operators

Different arithmetic operators are given below with an example:

Operator	Description	Example
%	Modulo	>> x = 50 ; y = 4 >> x % y 2
//	Integer divide	>> x = 50 ; y = 4 >> x // y 12
**	Exponent	>> x = 5 ; y = 4 >> x ** y 625

Assignment Operators

The *equal to* "=" operator is used for data assignment in Python.

Example:

```
a = 10
print(a)
a += 5
print(a)
```

```
10
15
```

- Assignment operators can be combined with arithmetic operators.
- Here, a += 5 is same as a = a + 5.
- Similar operators are: -= , *=, /=, %=, //=, and **= .

Comparison Operators

Comparison operators are used to compare two values

Operator	Description	Example
==	Returns True when the two values are equal.	>> x = 20 ; y = 20 >> x == y True
!=	Returns True when the two values are not equal.	>> x = 45 ; y = 24 >> x != y True

Note

These operators return true or false based on the comparison.

Comparison Operators

Comparison operators are used to compare two values

Operator	Description	Example
<	Returns True when first value is less than the second.	>> x = 20 ; y = 30 >> x < y True
>	Returns True when first value is greater than the second.	>> x = 20 ; y = 30 >> x > y False

Note

These operators return true or false based on the comparison.

Comparison Operators

Comparison operators are used to compare two values

Operator	Description	Example
<=	Returns True when first value is less than or equals to the second.	>> x = 40 ; y = 30 >> x <= y False
>=	Returns True when first value is greater than or equals to the second value.	>> x = 30 ; y = 30 >> x >= y True

Note

These operators return true or false based on the comparison.

Logical Operators

Logical operators are used for combining conditional statements.

```
a=1
b=2
a==1 and b==2
```

```
True
```

```
a=1
b=4
a==1 or b==2
```

```
True
```

```
b=0
not b
```

```
True
```

Operator	Description
and	Returns True if both statements are true
or	Returns True if one of the statements is true
not	Reverses the results, returns False if the result is true

Miscellaneous Operators

There are two kinds of miscellaneous operators, such as identity and membership operators.

```
a = ['a', 'b', 'c']
b = ['a', 'b', 'c']
a is b
```

```
False
```

```
a = ['a','b','c']
b = ['a','b','c']
a is not b
```

```
True
```

1. Identity Operators

Identity operators compare variables to see whether they are the same object at the same memory address.

- **is:** Returns True if both variables are the same object
- **is not:** Returns True if both variables are not the same object

Miscellaneous Operators

There are two kinds of miscellaneous operators, such as identity and membership operators.

```
a = [20, 45, 10]
10 in a

True

a = [20, 45, 10]
10 not in a

False
```

2. Membership Operators

Membership operators are used to testing if a sequence is present in an object.

in: Returns True if a value is present in the object

not in: Returns True if a value is not present in the object

DATA AND
ARTIFICIAL INTELLIGENCE

Strings in Python

Strings

Strings are a sequence of characters that can be either letters or alphanumeric.

Strings in python are enclosed in either single quotation marks or double quotation marks.

```
message_1 = "Hi! Welcome to Python Programming!!"
message_2 = 'Hi! Welcome to Python Programming!!'

print(message_1)
print(message_2)
```

```
Hi! Welcome to Python Programming!!
Hi! Welcome to Python Programming!!
```

In the above example, the values of message_1 and message_2 are same:

Strings

Triple single quotes (‘’’... ’’’) or triple double quotes (""" ... """) can be used to create multiline strings in Python.

```
message_1 = 'Hello Class!! \nHi! Welcome to Python Programming!!'
print(message_1)
```

```
Hello Class!!
Hi! Welcome to Python Programming!!
```

```
message_2 = '''Hello Class!!
Hi! Welcome to Python Programming!!'''
print(message_2)
```

```
Hello Class!!
Hi! Welcome to Python Programming!!
```


In the above example, the values of message_1 and message_2 are similar.

Accessing Characters in Strings

Strings can be accessed by using subscripts or indexes. Indexing in Python starts with 0.

The characters of the string can be accessed as:

```
string = "Hello World!"
print(string[0])
```

```
H
```

```
print(string[4])
```

```
o
```

The last character or reverse of the string can be accessed as:

```
print(string[-1])
```

```
!
```

```
print(string[-2])
```

```
d
```

Explanation

0	1	2	3	4	5	6	7	8	9	10	11
H	e	l	l	o		W	o	r	l	d	!
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

String Functions

String functions include:

Methods	Description
capitalize	Returns a string in sentence case, which means that the first letter is upper case, and the rest is lower case
lower	Returns a copy of the string converted to lowercase
upper	Returns a copy of the string converted to uppercase
strip	Returns a copy of the string with leading and trailing whitespace removed
join	Concatenate any number of strings

Strings are immutable Python objects.
All string methods return a new value and do not change the original string.

Key Takeaways

- In Python, data is represented as objects.
- Python supports different types of operators, such as arithmetic, comparison, logical, and miscellaneous operators.
- Strings are immutable objects in Python.
- Strings can be accessed by using subscripts or indexes.
- All string methods return a new value and do not change the original string.

DATA AND
ARTIFICIAL INTELLIGENCE

Thank You