Python Training Certification Course

Data Operations

Learning Objectives

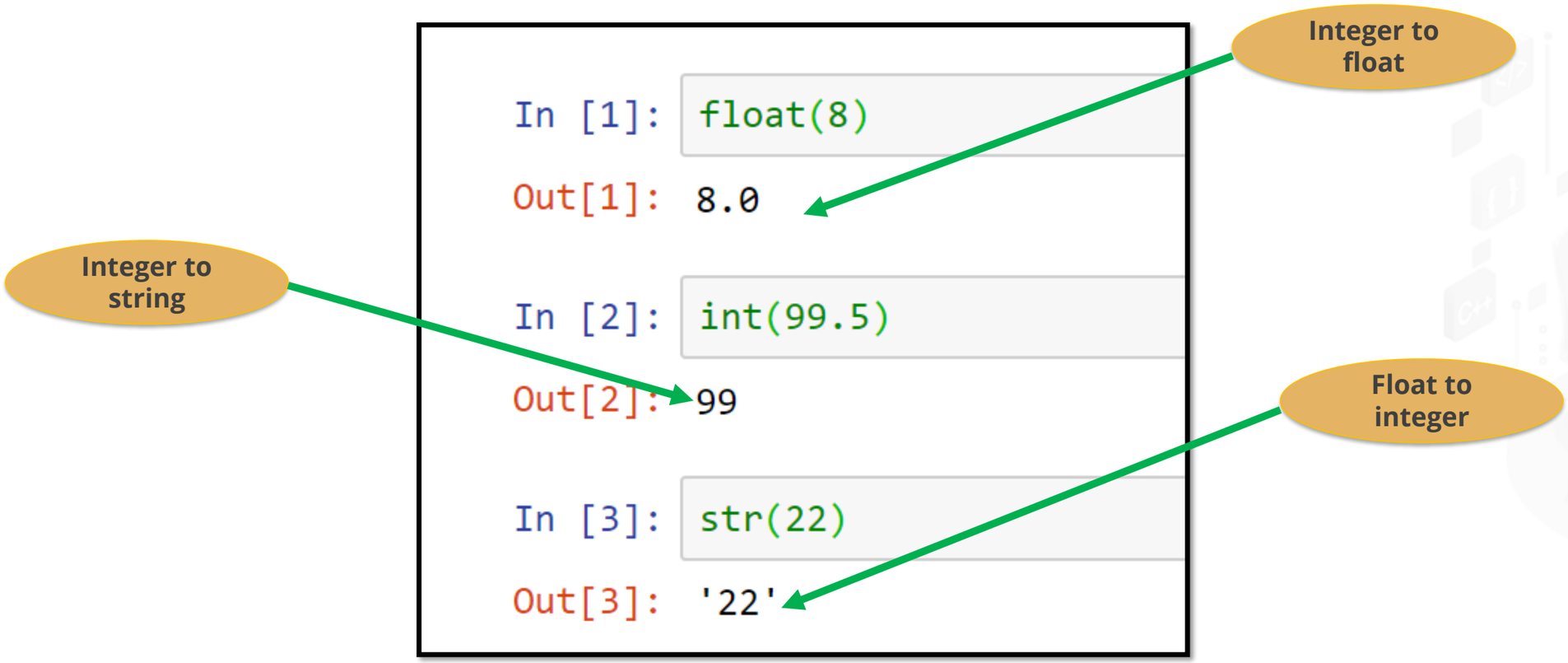
By the end of this lesson, you will be able to:

- Describe data type conversions and data operations
- Explain string methods, list methods, and tuple methods
- Perform operations on sets and dictionaries

Data Type Conversion

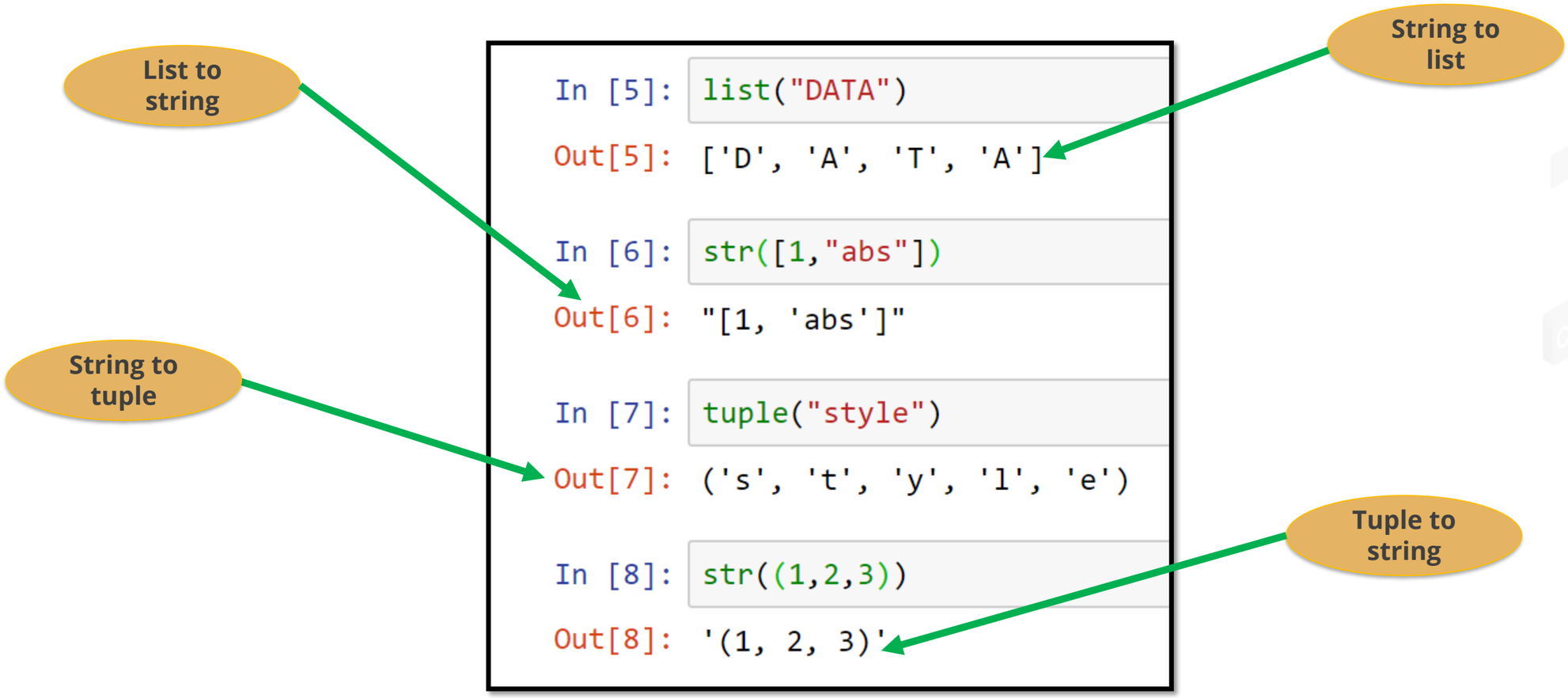
Data Type Conversion

In Python, we use datatype conversion to convert one data type into another. Data conversion is only possible if data is valid for the converted data type.



Data Type Conversion

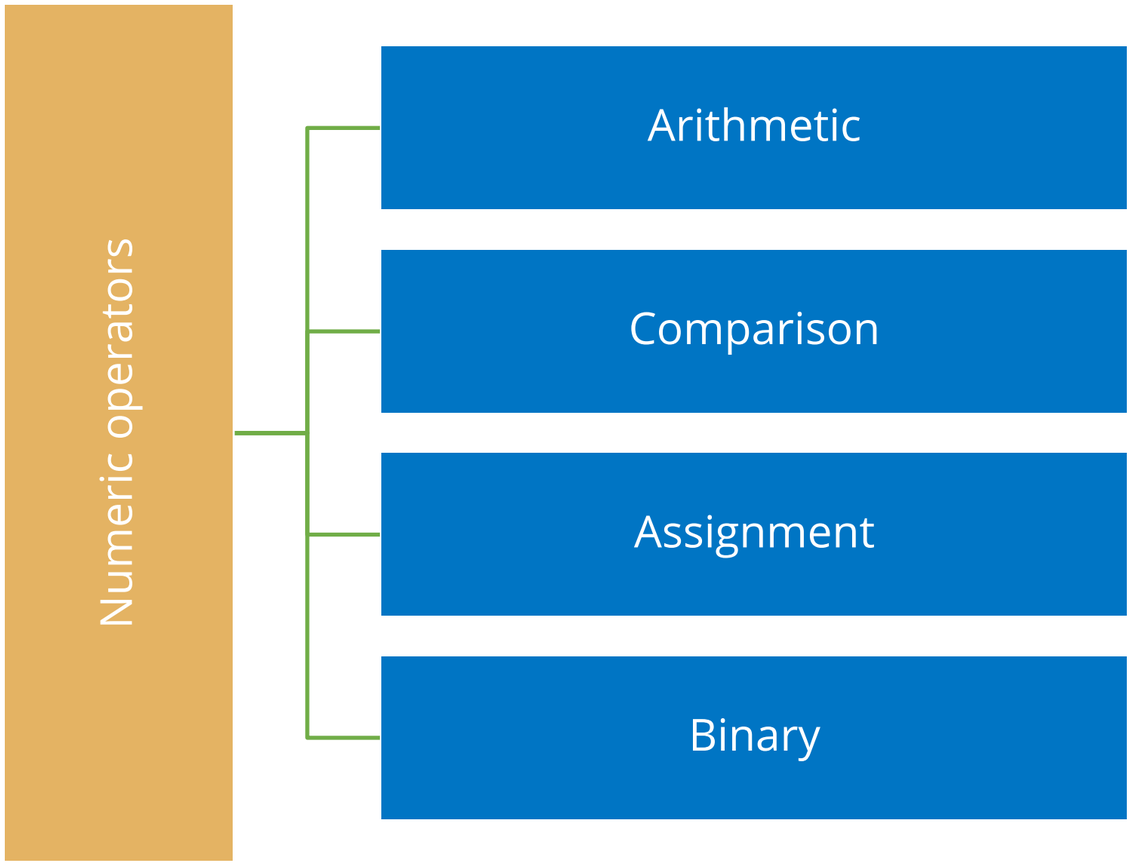
Unlike strings, lists and tuples can be converted only to sequence data types.



Data Operations

Data Operators for Numeric Type

Major data operators valid for numeric type are:



Arithmetic Operators

Basic arithmetic or mathematical operators in Python:

S.No	Type	Example	Output
1.	Add: +	6+7.8	14.8
2.	Subtract: -	7.9-5.2	2.7
3.	Multiply: *	7*6	42
4.	Divide to get float output: /	7/6	1.1666666666666667
5.	Divide to get integer output: //	7//6	1
6.	Modulus: %	7%6	1
7.	Power: **	2**3	8

Arithmetic Operators

Example of arithmetic operators:

```
In [5]: n=5
        n_1=8.4
        n2=3+4j
        print("sum= ",n_1+n2)
        print("diff=",n+n2)
        print("product=",n*n_1)
        print("div = ",n_1/n)
        print("div with integer output= ",n_1//n)
        print("modulus = ",n%3)
        print("power = ",n**n)

        sum=  (11.4+4j)
        diff= (8+4j)
        product= 42.0
        div =  1.6800000000000002
        div with integer output=  1.0
        modulus =  2
        power =  3125
```

Comparison Operators

As the name implies, comparison operators compare data members or variables. These operators return values as **True** or **False**.

S.No	Type	Example	Output
1.	equal: ==	6==5+4j	False
2.	not equal: !=	18!=12	True
3.	greater than: >	2.2>0	True
4.	less than: <	4.5<2	False
5.	greater than or equal to: >=	13>=9	True
6.	less than or equal to: <=	12<=88.9	False

Comparison Operators

Example of comparison operators:

```
In [8]: 2.5==4+3j

Out[8]: False

In [10]: 3>=1.4

Out[10]: True

In [11]: 13<18

Out[11]: True
```

Assignment Operators

Assignment operators allow a variable to store a value.

S.No	Type	Description
1.	a=1	Variable a will store value 1
2.	a+=b	Implies a+b
3.	x = 4+21.4	x will return 25.4

```
In [23]: a=1
         a

Out[23]: 1

In [17]: 4+21.4

Out[17]: 25.4
```

Binary Operators

Binary operators, also called bitwise operators, are used to perform bit operations. Numeric values convert to binary values during binary operations.

S.No	Type	Symbol	Description
1.	AND	&, and	Returns 1 only if both bits are 1
2.	OR	|, or	Returns 1 if either of the bits is 1
3.	XOR	^	Copies the bit if it is set in one operand but not both
4.	Complement	~, not	Is unary and has the effect of flipping bits

Binary Operators

Example of binary operators:

```
In [37]: a=101
         b=100
         a&b

Out[37]: 100

In [38]: a|b

Out[38]: 101

In [39]: ~a

Out[39]: -102

In [40]: a^b

Out[40]: 1
```

Data Operations

Duration: 20 min.

Objective: Write a program using Python to demonstrate arithmetic operators, comparison operators, and bitwise operators.

Steps to demonstrate arithmetic operators, comparison operators, and bitwise operators:

1. Open Jupyter Notebook
2. Click on File New Notebook
3. Select Python (version 3)
4. Write your program
5. Save your program
6. Click on Run to execute program

ASSISTED PRACTICE

String Methods

String Operators

Example of string operators:

S.No	Syntax	Output
1.	“Hello”+”CLASS” (Concatenate)	‘HelloClass’
2.	“Hello” * 3 (Multiply)	‘HelloHelloHello’
3.	a = “Python” a[0] ,a[-1]	(‘P’, ‘n’)
4.	a=“Python” a[2:5]	'tho'
5.	b=“Work hard”(membership) “W” in b	True
6.	b=“Work hard”(membership) “w” in b	False
7.	b=“Work hard”(membership) “W”not in b	False

Indexing

Characters in a sequence are numbered with indexes starting at 0:
Example: a = “A Girl"

index	0	1	2	3	4	5
member	A		G	i	r	l

Use this syntax to access an individual character of a string:

variableName [**index**]

```
In [1]: a="A Girl"
        a[0]

Out[1]: 'A'

In [2]: a[-1]

Out[2]: 'l'
```

String Methods

S.No	Methods	Output
1.	capitalize: Capitalizes first letter of a string Example: a="python“, a.capitalize()	Python
2.	upper: Converts whole string to uppercase Example: a.upper()	PYTHON
3.	lower: Converts whole string to lowercase Example: a.lower()	python
4.	len: Returns length of a string len(“python”)	6
5.	max & min: Returns maximum and minimum alphabets respectively Example: max(“python”), min(“python”)	(y, h)
6.	spilt: Splits string sentence into words Example: b=“This is class” b.split(" ")	[This, is, class]

String Methods

Example of string methods:

```
In [4]: a="This is Python Class"
        a.split()

Out[4]: ['This', 'is', 'Python', 'Class']
```

```
In [2]: s,x,y="1234","ASHA"," "
        print("returns true for digit members=",s.isdigit())
        print("returns true if space=",y.isspace())
        print("returns true if alphabets=",x.isalpha())
        print("returns true if uppercase members=",x.isupper())

        returns true for digit members= True
        returns true if space= True
        returns true if alphabets= True
        returns true if uppercase members= True
```

String Operations

Duration: 20 min.

Objective: Write a program using Python to perform different operations on a string.

Steps to perform different operations on a string:

1. Open Jupyter Notebook
2. Click on File New Notebook
3. Select Python (version 3)
4. Write your program
5. Save your program
6. Click on Run to execute program

ASSISTED PRACTICE

List Methods

List Operators

These are the list operators in Python:

S.No	Expression	Output	Description
1.	Len([4,5,6,6])	4	Length
2.	[8,7]+[1,2]	[8,7,1,2]	Concatenation
3.	x=["a",1] * 2	["a",1, "a",1]	Multiplication
4.	4 in [8,7,1,2]	False	Membership
5.	a=[4,5,6,6] ,a[0]	4	Indexing

List Methods

These are the list methods in Python:

S.No	Expression	Output	Description
1.	max([2,7,-1])	7	Returns max value
2.	a=[1,2,3] ,a.append(“6”)	[1,2,3,6]	Adds a single element to a list
3.	a.insert(0,3)	[3,1,2,3,6]	Adds members at specific positions
4.	a.remove(1)	[3,2,3,6]	Removes elements from a list
5.	a.count(3)	2	Returns occurrences of an element in a list
6.	a.sort()	[1,2,3,6]	Sorts elements in ascending order
7.	a.reverse()	[6,3,2,1]	Reverses a list

List Methods

An example:

```
In [4]: l=[1,2,3,"p"]
        l.append("a")
        l

Out[4]: [1, 2, 3, 'p', 'a']

In [6]: l.insert(2,-7)
        l

Out[6]: [1, 2, -7, -7, 3, 'p', 'a']

In [7]: del l[4]
        l

Out[7]: [1, 2, -7, -7, 'p', 'a']
```

List Operations

Duration: 20 min.

Objective: Write a program using Python to perform methods and operations on a list.

Steps to perform methods and operations on a list:

1. Open Jupyter Notebook
2. Click on File New Notebook
3. Select Python (version 3)
4. Write your program
5. Save your program
6. Click on Run to execute program

ASSISTED PRACTICE

Tuple Methods

Tuple Operations

These are the tuple operators in Python:

S.No	Expression	Output	Description
1.	Len((4,5,6,))	4	Length
3.	("a",1) * 2	("a",1, "a",1)	Multiplication
4.	8 in (8,7,1,2)	True	Membership
5.	a=(4,5,6,6) ,a[-1]	6	Indexing

Tuple Methods

These are the tuple methods in Python:

S.No	Expression	Output	Description
1.	min(1,3,-2,0)	-2	Returns minimum value
2.	x=(1,1,2,9) x.count(1)	2	Counts member occurrence
3.	x.index(9)	3	Returns the smallest index

```
b=(1,2,3,5,6)
del b[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-44-5512826b0c9a> in <module>()
      1 b=(1,2,3,5,6)
----> 2 del b[0]

TypeError: 'tuple' object doesn't support item deletion
```

Tuple Operations

Duration: 20 min.

Objective: Write a program using Python to perform operations on tuple.

Steps to perform operations on tuple:

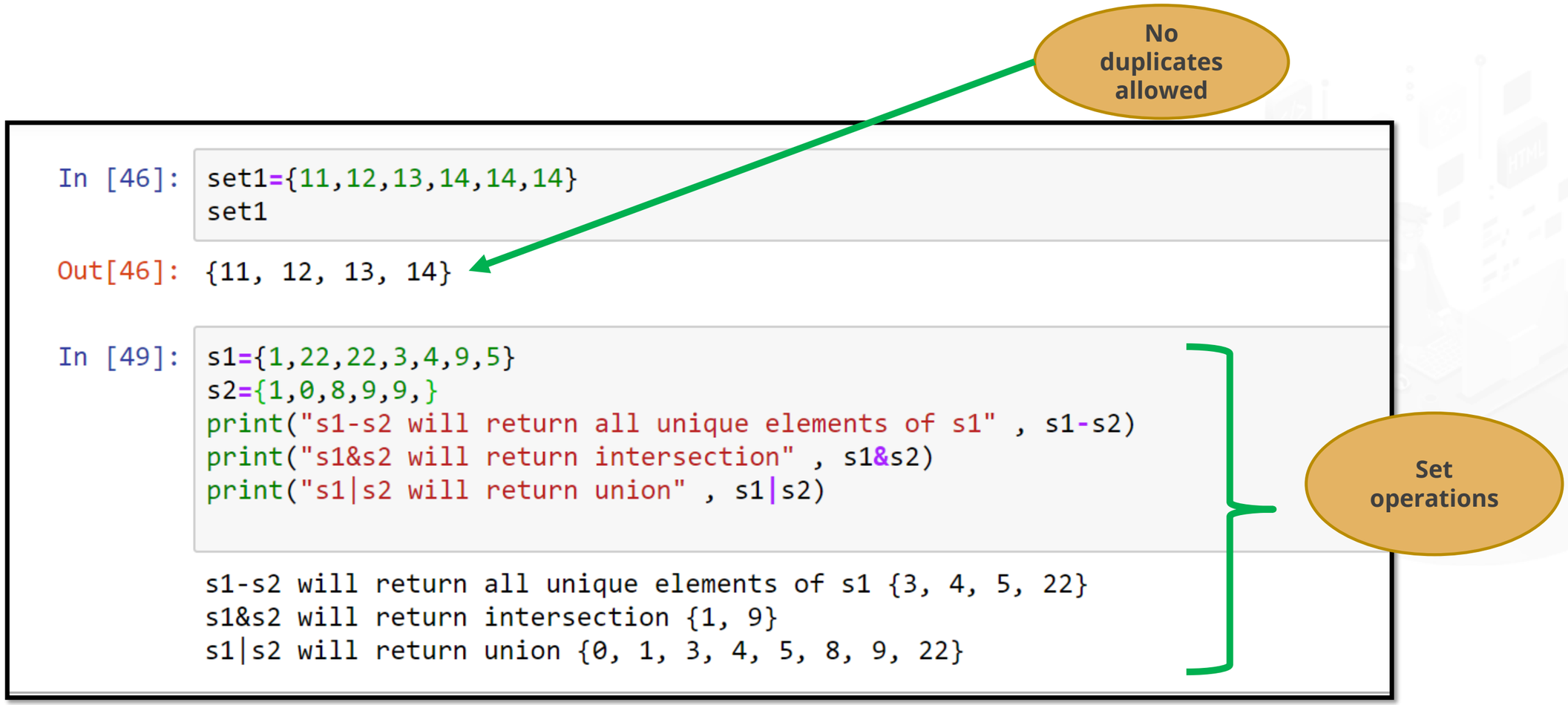
1. Open Jupyter Notebook
2. Click on File New Notebook
3. Select Python (version 3)
4. Write your program
5. Save your program
6. Click on Run to execute program

ASSISTED PRACTICE

Sets

Sets

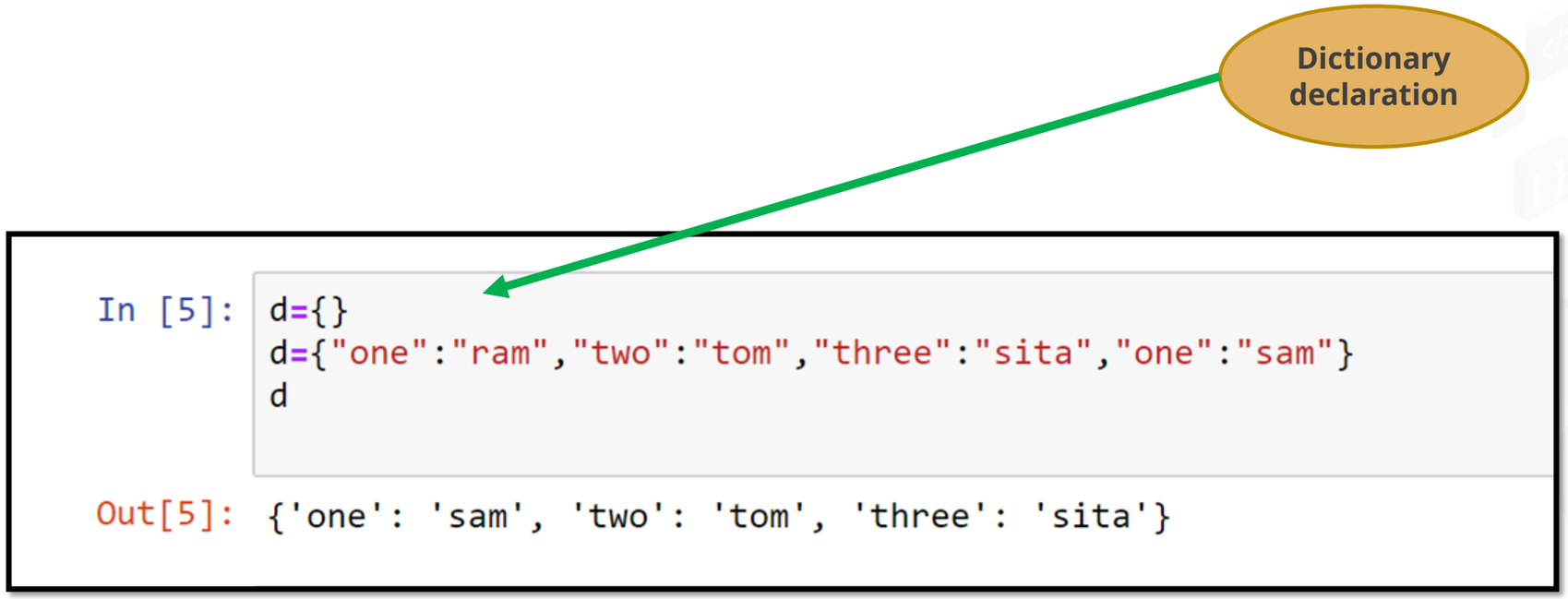
A **set** is an unordered collection of data types. It is mutable and has no duplicate elements.



Dictionaries

Dictionaries

A dictionary is an unordered collection of items. It maps a set of objects(keys) to another set of objects(values). **Dictionaries** are mutable, which means they can be changed.



Dictionaries: Methods and Operations

Dictionary members are represented in **key-value** forms; duplicate keys are not allowed while duplicate values are allowed. Each key in a dictionary points to a respective value.

```
In [9]: c={"a":2,"b":5,"c":3,"d":4,"e":-1}
        print("Class=",type(c))
        print("Call members throgh keys",c["a"])
        print("Keys in above dictionary=",c.keys())
        print("Values in above dictionary=",c.values())
        print("Max value =",max(c.values()))

        Class= <class 'dict'>
        Call members throgh keys 2
        Keys in above dictionary= dict_keys(['a', 'b', 'c', 'd', 'e'])
        Values in above dictionary= dict_values([2, 5, 3, 4, -1])
        Max value = 5
```

Dictionaries: Methods and Operations

sort() function present in an operator module is used to arrange dictionary members in an ascending and descending order.

```
In [10]: #sort (ascending and descending) a dictionary by value.
         import operator
         d = {3: -2, 6: 4, 4: 3, 2: 1, -3: 0}
         print('Original  ',d)
         sorted_d = sorted(d.items(), key=operator.itemgetter(0))
         print('Dictionary in ascending order by value : ',sorted_d)
         sorted_d = sorted(d.items(), key=operator.itemgetter(1),reverse=True)
         print('Dictionary in descending order by value : ',sorted_d)
         print("*"*20)

         Original   {3: -2, 6: 4, 4: 3, 2: 1, -3: 0}
         Dictionary in ascending order by value :  [(-3, 0), (2, 1), (3, -2), (4, 3), (6, 4)]
         Dictionary in descending order by value :  [(6, 4), (4, 3), (2, 1), (-3, 0), (3, -2)]
         ********************
```

Dictionaries: Methods and Operations

We use update() function to merge two dictionaries.

In [12]:

```
d1 = {'a': 1, 'b': 2}
d2 = {'a': 3, 'd': 4}
print("merge two dictionaries into d1:")
d1.update(d2)
print(d1)
```

```
merge two dictionaries into d1:
{'a': 3, 'b': 2, 'd': 4}
```

Dictionaries: Methods and Operations

We use zip() function to create a dictionary from two lists.

```
In [13]: #Map two list into dictionaries
         a = ['red', 'green', 'blue']
         b = [0,1,2]
         d1 = dict(zip(b, a))
         print(d1)
         print("*"*20)
```

```
{0: 'red', 1: 'green', 2: 'blue'}
********************
```

Dictionary Operations

Duration: 20 min.

Objective: Write a program using Python to sort, merge, and concatenate operations on dictionaries.

Steps to sort, merge, and concatenate operations on dictionaries:

1. Open Jupyter Notebook
2. Click on File New Notebook
3. Select Python (version 3)
4. Write your program
5. Save your program
6. Click on Run to execute program

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to sort, merge, and concatenate operations on dictionaries:

1. Open Jupyter Notebook
2. Click on File New Notebook
3. Select Python (version 3)
4. Write your program
5. Save your program
6. Click on Run to execute program

Knowledge Check

Knowledge Check

1

Which of the following is true for data type conversions?

a. Numeric data types can be converted to both tuples and lists

b. String 1234 can be converted to a numeric data type

c. Numeric data types can be converted to tuples

d. Numeric data types can be converted to lists

Knowledge Check

1

Which of the following is true for data type conversions?

a. Numeric data types can be converted to both tuples and lists

b. String 1234 can be converted to a numeric data type

c. Numeric data types can be converted to tuples

d. Numeric data types can be converted to lists

The correct answer is **b**

Only strings with digits can be converted to numeric data types.

Knowledge Check 2	If a= [1,2,3,4], b= [4,6,7,8], then operation a+b will give the following output:

a. [1,2,3,4,4,6,7,8]

b. [1,2,3,4,6,7,8]

c. [5,8,10,12]

d. [1,2,3,4]+[4,6,7,8]

Knowledge Check

2

If a= [1,2,3,4], b= [4,6,7,8], then operation a+b will give the following output:

a. [1,2,3,4,4,6,7,8]

b. [1,2,3,4,6,7,8]

c. [5,8,10,12]

d. [1,2,3,4]+[4,6,7,8]

The correct answer is **a**

a+b concatenates lists, allowing duplicates.

Knowledge Check

3

Which of the following is not possible for tuples?

a. Deleting members

b. Appending new members

c. Changing member values

d. All of the above

Knowledge Check

3

Which of the following is not possible for tuples?

a. Deleting members

b. Appending new members

c. Changing member values

d. All of the above

The correct answer is **d**

Tuples are immutable sequence types.

Knowledge Check 4	Let x= {1: "apple", 2: “grapes"}, then x[0] will return:

a. apple

b. 1

c. An error

d. 1: "apple"

Knowledge Check 4	Let x= {1: "apple", 2: “grapes"}, then x[0] will return:

a. apple

b. 1

c. An error

d. 1: "apple"

The correct answer is **c**

x[0] points to nothing in the dictionary above. Keys are natural indexes for dictionary values.

Knowledge Check 5	Let Set1= {1,2,3,4,5,5}, Set2= {3,6,5}, then output for (Set1&Set2) will be:

a. {3,5,5}

b. {1,2,3,4,5,5,5,6}

c. {3,5}

d. {6}

Knowledge Check 5	Let Set1= {1,2,3,4,5,5} ,Set2= {3,6,5}, then output for (Set1&Set2) will be:

a. {3,5,5}

b. {1,2,3,4,5,5,5,6}

c. {3,5}

d. {6}

The correct answer is **c**

& represents the intersection of two sets.

Key Takeaways

- Strings and tuples are immutable data types.
- Duplicate values are not allowed in sets.
- A dictionary is an unordered collection of items.