Python

Unit 1

By
Gaurav Siddharth

Introduction, Data Types and Operators

- Installation and working
- Variables and data types
- Perform computations and create logical statements using Python’s operators:

  Arithmetic, Assignment, Comparison, Logical, Membership, Identity, Bitwise operators, list, tuple and string operations.

Introduction

- Low-level versus high-level
- General versus targeted to an application domain
- Interpreted versus compiled

- Python is a high-level, general purpose and interpreted programming language.
- The language and many supporting tools are free, and Python programs can run on any operating system.
- Integrated development environment (IDE)
- All of the Python IDEs provide
  - A text editor with syntax highlighting, auto completion, and smart indentation
  - A shell with syntax highlighting, and
  - An integrated debugger.

Introduction

Features:

- Free
- Less to type, debug and maintain
- Programs are simpler, shorter, more flexible
- No compiling
- No type declarations
- Automatic memory management
- High-level data types and operations
- Fewer restrictions and rules
- Strong library support
- Component Integration: Can invoke C, C++ libraries and Java components

Introduction

Use of Python for various applications:

(a) System programming

(b) Building GUI applications

(c) Internet scripting

(d) Component integration

(e) Database programming

(f) Rapid prototyping

(g) Numeric and Scientific programming

(h) Game programming

(i) Robotics programming

Python is used by:

Intel, HP, Seagate, IBM, Qualcomm, Google, Youtube, etc

Introduction

Programming Paradigms: Paradigm means organization principle. It is also known as model. Python supports

(a) Functional Programming Model

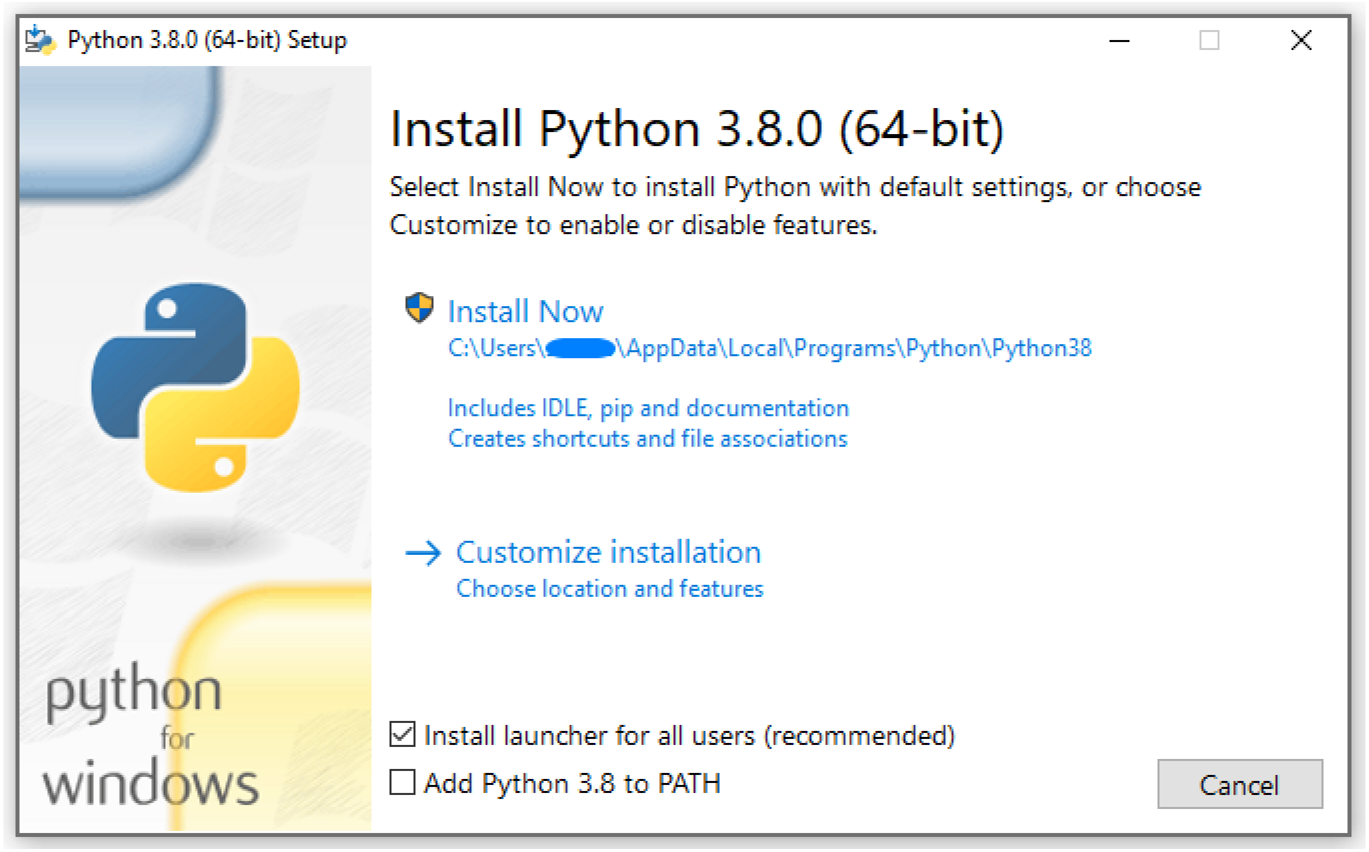
(b) Procedural Programming Model

(c) Object-oriented Programming Model

(d) Event-driven Programming Model

Installation and working

- http://www.python.org
- http://www.python.org/doc/



Python 3.8.0 (64-bit) Setup

Install Python 3.8.0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\[redacted]\AppData\Local\Programs\Python\Python38

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.8 to PATH

Python Resources

- Python source code, binaries and documentation is available at:
  - Python official website: www.python.org (http://www.python.org)
  - Documentation website: www.python.org/doc (http://www.python.org/doc
- Program development in Python can be done in 3 ways:
  - Using built-in IDLE.
  - Using third-party IDEs.
  - Using online Python shells.
- Third-party development tools and the links from where they can be downloaded are given below:
  - NetBeans IDE for Python:
    https://download.netbeans.org/netbeans/6.5/python/ea/ (https://download
  - PyCharm IDE for Python:
    https://www.jetbrains.com/pycharm (https://www.jetbrains.com/pycharm)
  - Visual Studio Code IDE:
    https://code.visualstudio.com/download (https://code.visualstudio.com/dow
- If you do not wish to install any Python development tool on your machine, then you can use any of the following online Python shells:
  - https://www.python.org/shell/ (https://www.python.org/shell/)
  - https://ideone.com/ (https://ideone.com/)
  - https://repl.it/languages/python3 (https://repl.it/languages/python3)

Working

- Python Integrated Development and Learning Environment (IDLE)-

  It offers handy features like syntax highlighting, context-sensitive help and debugging.

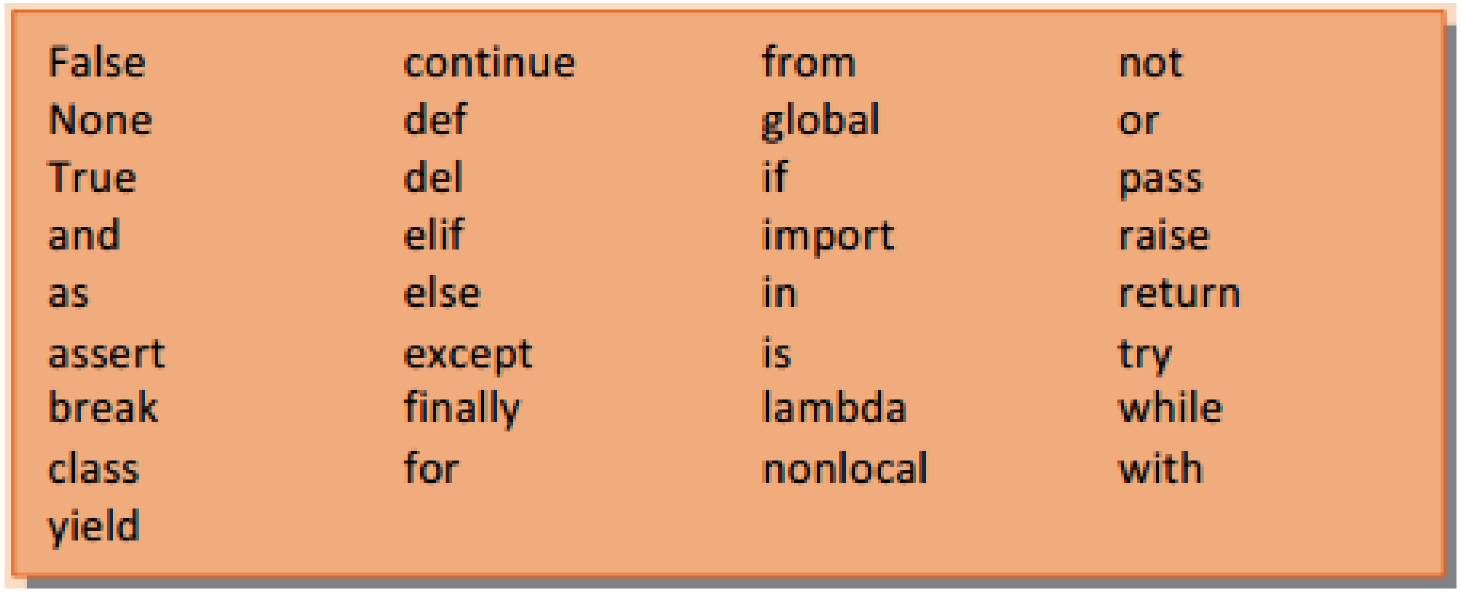
- Python can be used in two modes:

  Interactive mode

  Script mode

- Identifiers and Keywords

  Python is a case sensitive language.



False	continue	from	not
None	def	global	or
True	del	if	pass
and	elif	import	raise
as	else	in	return
assert	except	is	try
break	finally	lambda	while
class	for	nonlocal	with
yield			

Data Types and Variable

- Python supports 3 categories of data types:

  Basic types - int, float, complex, bool, string, bytes

  Container types - list, tuple, set, dict

  User-defined types - class

- Variable Type and Assignment

  ```
  a  = 25      #  type  of a  is inferred  as  int
  b  = 31.4    #  type  of a  is inferred as  float
  c  =  'Hi'   #  type  of a  is inferred  as str
  ```

  By using type() function we can determine the data type of a variable- print(type(a))

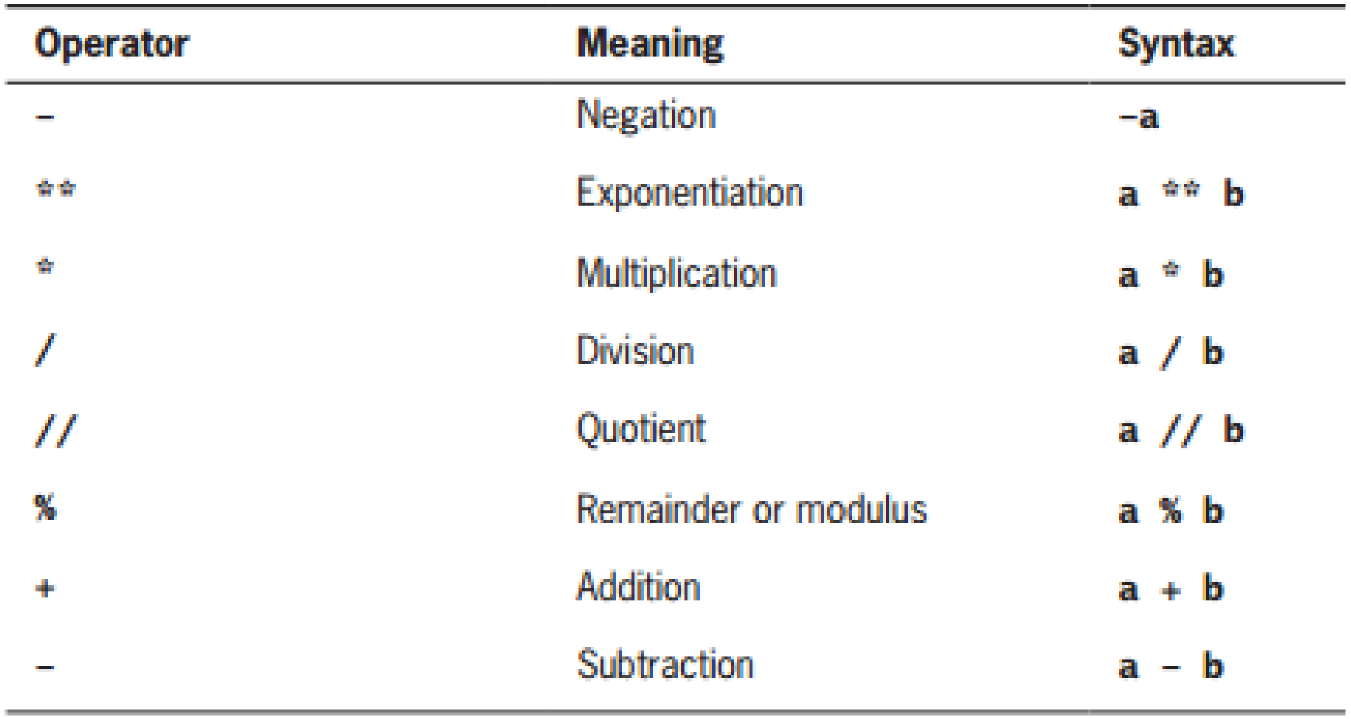
- Multiple variable assignment:

  ```
  a  = 5 ;  x  = 3.4 ;  name =  ‘Ranchi’        #  use ;  as  statement separator
  a,  x, name = 5, 3.4,  'Ranchi'
  a  = b = c = d =  5
  ```

Operators

Description	Operator	Associativity
Grouping	( )	Left to Right
Function call	function( )	Left to Right
Slicing	[start:end:step]	Left to Right
Exponentiation	**	Right to Left
Bitwise NOT	~	Right to Left
Unary plus / minus	+ -	Left to Right
Multiplication	*	Left to Right
Division	/	Left to Right
Modular Divsion	%	Left to Right
Addition	+	Left to Right
Subtraction	-	Left to Right
Bitwise left shift	<<	Left to Right
Bitwise right shift	>>	Left to Right
Bitwise AND	&	Left to Right
Bitwise XOR	^	Left to Right
Bitwise OR	|	Left to Right
Membership	In not in	Left to Right
Identity	is is not	Left to Right
Relational	< > <= >=	Left to Right
Equality	==	Left to Right
Inequality	!=<>	Left to Right
Logical NOT	not	Left to Right
Logical AND	and	Left to Right
Logical OR	or	Left to Right
Assignment	= += -= *= /= %= //= **= &= |= ^= >>= <<=	Right to Left

Arithmetic Operators



Operator	Meaning	Syntax
–	Negation	–a
**	Exponentiation	a ** b
*	Multiplication	a * b
/	Division	a / b
//	Quotient	a // b
%	Remainder or modulus	a % b
+	Addition	a + b
–	Subtraction	a – b

- a // b, result is the largest integer which is less than or equal to the quotient. // is called floor division operator.

Example

• print(10 // 3)	3
• print(-10 // 3)	-4
• print(10 // -3)	-4
• print(-10 // -3)	3
• print(3 // 10)	0
• print(3 // -10)	-1
• print(-3 // 10)	-1
• print(-3 // -10)	0

Answer

- print(10 // 3)   # yields 3
- print(-10 // 3)   # yields -4
- print(10 // -3)   # yields -4
- print(-10 // -3)   # yields 3
- print(3 // 10)   # yields 0
- print(3 // -10)   # yields -1
- print(-3 // 10)   # yields -1
- print(-3 // -10)   # yields 0

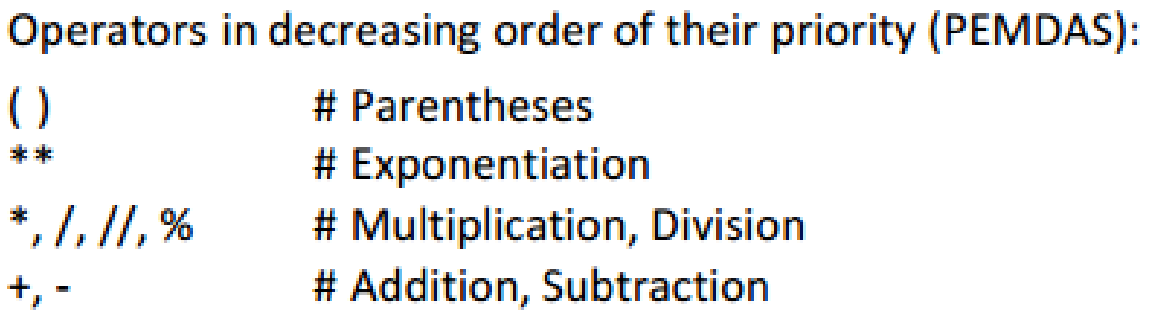
Operation a % b is evaluated as a - (b * (a // b))

- print(10 % 3)
- print(-10 % 3)
- print(10 % -3)
- print(-10 % -3)
- print(3 % 10)
- print(3 % -10)
- print(-3 % 10)
- print(-3 % -10)

Answer

- print(10 % 3)   # yields 1
- print(-10 % 3)   # yields 2
- print(10 % -3)   # yields -2
- print(-10 % -3)   # yields -1
- print(3 % 10)   # yields 3
- print(3 % -10)   # yields -7
- print(-3 % 10)   # yields 7
- print(-3 % -10)   # yields -3

Precedence and Associativity



Operators in decreasing order of their priority (PEMDAS):

```
( )              # Parentheses
**               # Exponentiation
*, /, //, %      # Multiplication, Division
+, -             # Addition, Subtraction
```

- If there is a tie between operators of same precedence, it is settled using associativity of operators.

Assignment Operators

Operator	Function
=	
+=	operator.iadd(a,b)
-=	operator.isub(a,b)
*=	operator.imul(a,b)
/=	operator.itruediv(a,b)
//=	operator.ifloordiv(a,b)
%=	operator.imod(a, b)
&=	operator.iand(a, b)
|=	operator.ior(a, b)
^=	operator.ixor(a, b)
>>=	operator.irshift(a, b)
<<=	operator.ilshift(a, b)

Comparison Operators

- The comparison operators compare two operands and return a boolean either True or False.

Operator	Function	Description
>	operator.gt(a,b)	True if the left operand is higher than the right one
<	operator.lt(a,b)	True if the left operand is lower than right one
==	operator.eq(a,b)	True if the operands are equal
!=	operator.ne(a,b)	True if the operands are not equal
>=	operator.ge(a,b)	True if the left operand is higher than or equal to the right one
<=	operator.le(a,b)	True if the left operand is lower than or equal to the right one

Logical Operators

- The logical operators are used to combine two boolean expressions. The logical operations are generally applicable to all objects, and support truth tests, identity tests, and boolean operations

Operator	Description
and	True if both are true
or	True if at least one is true
not	Returns True if an expression evaluates to false and vice-versa

Membership Test Operators

- The membership test operators in and not in test whether the sequence has a given item or not.

Operator	Function	Description
in	operator.contains(a,b)	Returns True if the sequence contains the specified item else returns False.
not in	not operator.contains(a,b)	Returns True if the sequence does not contains the specified item, else returns False.

Identity Operators

- The identity operators check whether the two objects have the same id value i.e. both the objects point to the same memory location.

Operator	Function	Description
is	operator.is_(a,b)	True if both are true
is not	operator.is_not(a,b)	True if at least one is true

Bitwise Operators

- Bitwise operators perform operations on binary operands.

Operator	Function	Description
&	operator.and_(a,b)	Sets each bit to 1 if both bits are 1.
|	operator.or_(a,b)	Sets each bit to 1 if one of two bits is 1.
^	operator.xor(a,b)	Sets each bit to 1 if only one of two bits is 1.
~	operator.invert(a)	Inverts all the bits. And then find the 2's complement of the bits and then convert it into decimal.
<<	operator.lshift(a,b)	Shift left by pushing zeros in from the right and let the leftmost bits fall off.
>>	operator.rshift(a,b)	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off.

in shift => add 1 and then change sign

Data Types

String

- In Python, string is an immutable sequence data type. It is the sequence of Unicode characters wrapped inside single ‘ ’, double “ ”, or triple ‘‘‘ ’’’quotes.
- Multi-line strings must be embed in triple quotes.
- If a string literal required to embed double quotes as part of a string then, it should be put in single quotes. Likewise, if a string includes a single quote as a part of a string then, it should be written in double quotes.
- Use the len() function to retrieve the length of a string.
- Python support both positive and negative indexing.
- Use the str() function to convert a number to a string.
- Use a backslash character followed by the character you want to insert in a string e.g. \' to include a quote, or \" to include a double quotes in a string.
- Use r or R to ignore escape sequences in a string.

Escape sequence

Escape sequence	Description
\	Backslash
\n	Newline
\t	Tab

String Methods

Method	Description
str.capitalize()	Returns the copy of the string with its first character capitalized and the rest of the letters are in lowercased.
string.casefold()	Returns a lowered case string. It is similar to the lower() method, but the casefold() method converts more characters into lower case.
string.center()	Returns a new centered string of the specified length, which is padded with the specified character. The default character is space.
string.count()	Searches (case-sensitive) the specified substring in the given string and returns an integer indicating occurrences of the substring.
string.endswith()	Returns True if a string ends with the specified suffix (case-sensitive), otherwise returns False.
string.expandtabs()	Returns a string with all tab characters \t replaced with one or more space, depending on the number of characters before \t and the specified tab size.
string.find()	Returns the index of the first occurence of a substring in the given string (case-sensitive). If the substring is not found it returns -1.
string.index()	Returns the index of the first occurence of a substring in the given string.
string.isalnum()	Returns True if all characters in the string are alphanumeric (either alphabets or numbers). If not, it returns False.

String Methods Cont.

Method	Description
string.isalpha()	Returns True if all characters in a string are alphabetic (both lowercase and uppercase) and returns False if at least one character is not an alphabet.
string.isascii()	Returns True if the string is empty or all characters in the string are ASCII.
string.isdecimal()	Returns True if all characters in a string are decimal characters. If not, it returns False.
string.isdigit()	Returns True if all characters in a string are digits or Unicode char of a digit. If not, it returns False.
string.isidentifier()	Checks whether a string is valid identifier string or not. It returns True if the string is a valid identifier otherwise returns False.
string.islower()	Checks whether all the characters of a given string are lowercased or not. It returns True if all characters are lowercased and False even if one character is uppercase.
string.isnumeric()	Checks whether all the characters of the string are numeric characters or not. It will return True if all characters are numeric and will return False even if one character is non-numeric.
string.isprintable()	Returns True if all the characters of the given string are Printable. It returns False even if one character is Non-Printable.
string.isspace()	Returns True if all the characters of the given string are whitespaces. It returns False even if one character is not whitespace.

String Methods Cont.

Method	Description
string.istitle()	Checks whether each word's first character is upper case and the rest are in lower case or not. It returns True if a string is titlecased; otherwise, it returns False. The symbols and numbers are ignored.
string.isupper()	Returns True if all characters are uppercase and False even if one character is not in uppercase.
string.join()	Returns a string, which is the concatenation of the string (on which it is called) with the string elements of the specified iterable as an argument.
stingr.ljust()	Returns the left justified string with the specified width. If the specified width is more than the string length, then the string's remaining part is filled with the specified fillchar.
string.lower()	Returns the copy of the original string wherein all the characters are converted to lowercase.
string.lstrip()	Returns a copy of the string by removing leading characters specified as an argument.
string.maketrans()	Returns a mapping table that maps each character in the given string to the character in the second string at the same position. This mapping table is used with the translate() method, which will replace characters as per the mapping table.
string.partition()	Splits the string at the first occurrence of the specified string separator sep argument and returns a tuple containing three elements, the part before the separator, the separator itself, and the part after the separator.

String Methods Cont.

Method	Description
string.replace()	Returns a copy of the string where all occurrences of a substring are replaced with another substring.
string.rfind()	Returns the highest index of the specified substring (the last occurrence of the substring) in the given string.
string.rindex()	Returns the index of the last occurence of a substring in the given string.
string.rjust()	Returns the right justified string with the specified width. If the specified width is more than the string length, then the string's remaining part is filled with the specified fill char.
string.rpartition()	Splits the string at the last occurrence of the specified string separator sep argument and returns a tuple containing three elements, the part before the separator, the separator itself, and the part after the separator.
string.rsplit()	Splits a string from the specified separator and returns a list object with string elements.
string.rstrip()	Returns a copy of the string by removing the trailing characters specified as argument.
string.split()	Splits the string from the specified separator and returns a list object with string elements.
string.splitlines()	Splits the string at line boundaries and returns a list of lines in the string.
string.startswith()	Returns True if a string starts with the specified prefix. If not, it returns False.

String Methods Cont.

Method	Description
string.strip()	Returns a copy of the string by removing both the leading and the trailing characters.
string.swapcase()	Returns a copy of the string with uppercase characters converted to lowercase and vice versa. Symbols and letters are ignored.
string.title()	Returns a string where each word starts with an uppercase character, and the remaining characters are lowercase.
string.translate()	Returns a string where each character is mapped to its corresponding character in the translation table.
string.upper()	Returns a string in the upper case. Symbols and numbers remain unaffected.
string.zfill()	Returns a copy of the string with '0' characters padded to the left. It adds zeros (0) at the beginning of the string until the length of a string equals the specified width parameter.

List

- In Python, the list is a mutable sequence type. A list object contains one or more items of different data types in the square brackets [] separated by a comma.
- List items can be accessed using a zero-based index in the square brackets [].
- A list can contain multiple inner lists as items that can be accessed using indexes.
- Use the list() constructor to convert from other sequence types such as tuple, set, dictionary, string to list.
- You can add new items in the list using the append() or insert() methods, and update items using indexes.
- Use the remove(), pop() methods, or del keyword to delete the list item or the whole list.

List Methods

List Method	Description
list.append()	Adds a new item at the end of the list.
list.clear()	Removes all the items from the list and make it empty.
list.copy()	Returns a shallow copy of a list.
list.count()	Returns the number of times an element occurs in the list.
list.extend()	Adds all the items of the specified iterable (list, tuple, set, dictionary, string) to the end of the list.
list.index()	Returns the index position of the first occurrence of the specified item. Raises a Value Error if there is no item found.
list.insert()	Inserts an item at a given position.
list.pop()	Returns an item from the specified index position and also removes it from the list. If no index is specified, the list.pop() method removes and returns the last item in the list.
list.remove()	Removes the first occurrence of the specified item from the list. It the specified item not found then throws a Value Error.
list.reverse()	Reverses the index positions of the elements in the list. The first element will be at the last index, the second element will be at second last index and so on.
list.sort()	Sorts the list items in ascending, descending, or in custom order.

Tuple

- Tuple is an immutable (unchangeable) collection of elements of different data types. It is an ordered collection, so it preserves the order of elements in which they were defined.
- Tuples are defined by enclosing elements in parentheses (), separated by a comma. However, it is not necessary to enclose the tuple elements in parentheses.
- Tuples cannot be declared with a single element unless followed by a comma.
- Tuple elements can be unpacked and assigned to variables. However, the number of variables must match with the number of elements in a tuple; otherwise, an error will be thrown.
- Tuple is unchangeable. So, once a tuple is created, any operation that seeks to change its contents is not allowed. However, you can delete an entire tuple using the del keyword.
- The tuple() constructor is used to convert any iterable to tuple type.

String, List and Tuple operators

Operator	Description
+	Returns a list containing all the elements of the first and the second list.
*****	Concatenates multiple copies of the same list.
[]	The slice operator **[]** returns the item at the given index. A negative index counts the position from the right side.
[FromIndex : Untill Index - 1]	The range slice operator **[FromIndex : Untill Index - 1]** fetches items in the range specified by the two index operands separated by : symbol. If the first operand is omitted, the range starts from the index 0. If the second operand is omitted, the range goes up to the end of the list.
in	Returns true if an item exists in the given list.
not in	Returns true if an item does not exist in the given list.

Set

- A set is a mutable collection of distinct objects, same as the list and tuple. It is an unordered collection of objects.
- The set is a Python implementation of the set in Mathematics.
- A set object contains one or more items, not necessarily of the same type, which are separated by a comma and enclosed in curly brackets {}.
- A set doesn't store duplicate objects. Hence, indexing and slicing operations cannot be done on a set object.
- Only immutable objects can be a part of a set object. Numbers (integer, float, as well as complex), strings, and tuple objects are accepted, but set, list, and dictionary objects are not.
- Use the set() function to create an empty set. Empty curly braces will create an empty dictionary instead of an empty set.
- The set() function is also used to convert string, tuple, or dictionary object to a set object.
- Use built-in set functions add(), remove() or update() methods to modify set collection.

Set Operations

Operators	Description
| **Method:** set.union()	**Union:** Returns a new set with elements from both the sets.
& **Method:** set.intersection()	**Intersection:** Returns a new set containing elements common to both sets.
- **Method:** set.difference()	**Difference:** Returns a set containing elements only in the first set, but not in the second set.
^ **Method:** set.symmetric_difference()	**Symmetric Difference:** Returns a set consisting of elements in both sets, excluding the common elements.

Set Methods

Method	Description
set.add()	Adds an element to the set. If an element is already exist in the set, then it does not add that element.
set.clear()	Removes all the elements from the set.
set.copy()	Returns a shallow copy of the set.
set.difference()	Returns the new set with the unique elements that are not in the another set passed as a parameter.
set.difference_update()	Updates the set on which the method is called with the elements that are common in another set passed as an argument.
set.discard()	Removes a specific element from the set.
set.intersection()	Returns a new set with the elements that are common in the given sets.
set.intersection_update()	Updates the set on which the instersection_update() method is called, with common elements among the specified sets.

Set Methods

Method	Description
set.isdisjoint()	Returns true if the given sets have no common elements. Sets are disjoint if and only if their intersection is the empty set.
set.issubset()	Returns true if the set (on which the issubset() is called) contains every element of the other set passed as an argument.
set.pop()	Removes and returns a random element from the set.
set.remove()	Removes the specified element from the set. If the specified element not found, raise an error.
set.symmetric_difference()	Returns a new set with the distinct elements found in both the sets.
set.symmetric_difference_update()	Updates the set on which the instersection_update() method called, with the elements that are common among the specified sets.
set.union()	Returns a new set with distinct elements from all the given sets.
set.update()	Updates the set by adding distinct elements from the passed one or more iterables.

Dictionary

- The dictionary is an unordered collection that contains key:value pairs separated by commas inside curly brackets {}. Dictionaries are optimized to retrieve values when the key is known.
- The key should be unique and an immutable object. A number, string or tuple can be used as key.
- However, a dictionary with a list as a key is not valid, as the list is mutable. But, a list can be used as a value.
- The same key cannot appear more than once in a collection. If the key appears more than once, only the last will be retained.
- One value can be assigned to more than one key.
- A dictionary can also be created using the dict() constructor method.
- Dictionary is an unordered collection, so a value cannot be accessed using an index; instead, a key must be specified in the square brackets.
- Use the same key and assign a new value to it to update the dictionary object.
- Use a new key and assign a value to it. The dictionary will show an additional key-value pair in it.

Dictionary Methods

Method	Description
dict.clear()	Removes all the key-value pairs from the dictionary.
dict.copy()	Returns a shallow copy of the dictionary.
dict.fromkeys()	Creates a new dictionary from the given iterable (string, list, set, tuple) as keys and with the specified value.
dict.get()	Returns the value of the specified key.
dict.items()	Returns a dictionary view object that provides a dynamic view of dictionary elements as a list of key-value pairs. This view object changes when the dictionary changes.
dict.keys()	Returns a dictionary view object that contains the list of keys of the dictionary.

Dictionary Methods

Method	Description
dict.pop()	Removes the key and return its value. If a key does not exist in the dictionary, then returns the default value if specified, else throws a KeyError.
dict.popitem()	Removes and return a tuple of (key, value) pair from the dictionary. Pairs are returned in Last In First Out (LIFO) order.
dict.setdefault()	Returns the value of the specified key in the dictionary. If the key not found, then it adds the key with the specified defaultvalue. If the defaultvalue is not specified then it set None value.
dict.update()	Updates the dictionary with the key-value pairs from another dictionary or another iterable such as tuple having key-value pairs.
dict.values()	Returns the dictionary view object that provides a dynamic view of all the values in the dictionary. This view object changes when the dictionary changes.

Summary of String, List, Tuple, Set and Dictionary

- String - ordered collection, immutable, iterable.
- List - ordered collection, mutable, iterable.
- Tuple - ordered collection, immutable, iterable.
- Set - unordered collection, mutable, iterable.
- Dictionary - unordered collection, mutable, iterable.

THANK YOU