Programming on Python

Content

5. **LECTURE 5:** Input-Output. Printing on screen. Reading data from keyboard. Opening and closing file. Reading and writing files Functions
   .Python input/output
   .Reading input from the keyboard
   .Python Files (Files as types)
   .File Handling (“r”, “a”, “w”, “x”). Additionally (“t”, “b”)
   .Python File Open
   .Python File Read (read only parts, read lines, close files)
   .Python File Write (“a”, “w”)
   .Python File Creation (“x”, “a”,”w”)
   .Python Delete File (os.remove())
   .Check File Exist (os.path.exists())
   .Delete Folder (os.rmdir)

6. **LECTURE 6:** Functions. Defining a function. Calling a function. Types of functions. Function Arguments
   .Python Functions (What is a function in Python?, SYNTAX(optional)**)**
   .Docstring
   .Parameters
   .Arguments
   .Types of Arguments (Required Arguments, Keyword Arguments, Default Arguments, Variable-length Arguments)
   .Return Values (Fruitful and Nonfruitful functions)
   .Function Types (User Defined functions, Built-In functions)
   .Type Conversions

7. **LECTURE 7:** Regular Expressions (YOU CAN USE SITE **regex101** чтобы проверить выражение)
   .What is a Regular Expression?
   .Specify Pattern using Regex
   .Metacharacters ([], . , ^ , $, * , + , ? , {} , | , () , \ )
   .Special Sequences (\A, \b, \B, \d, \D, \s, \S, \w, \W, \Z)
   .Python regex (import re, re.findall(), re.split(), re.sub(), re.subn(), re.search(), group(), match.start(), match.end(), match.span(), re.findall())

8. **LECTURE 8:** OOPs concept. Class and object. Inheritance, Overloading, Overriding. Data hiding
   .Object Oriented Programming
   .Principles of OOP (Encapsulation , Abstraction , Inheritance, Polymorphism)
   .OOP Languages
   .Python Classes (methods, properties)
   .Python Object
   .Object Methods
   .”Implicit” parameter (self)
   .Calling Methods (object.method(parameters), Class.method(object,parameters))
   .Constructor
   .Self parameter
   .Modify object properties, Delete Object Properties, Delete Objects
   .Pass statement

.Python Inheritance (parent class , child class , super function, add properties)
.Encapsulation (_ , __)
.Polymorphism

9. **LECTURE 9:** Databases
.Data Modelling
.History of Database
.Type of Models (Conceptual data models, logical data models, physical data models)
.Database Model (Hierarchical Model, Network Model, Relational Model)
.Hierarchical model
.Network model
.Relational model
.Relation (Name, Attributes, Tuples)
.DBMS Components (Hardware, Software, Data, Users, Procedures)
.DBMS Environment (Hardware, Software , Data , People , Procedure)
.DBMS Facility (Data Definition language(DDL), Data Manipulation Language(DML), Structured Query Language (SQL) , Security system, Integrity system , Concurrency control system , Backup & recovery system , view mechanism)
.Advantages of DBMS
.Limitations of DBMS
.Database architecture (Internal level, Conceptual level , External Level)
.Operations on Relations (Insert , Delete, Update , Select, Project, Join ,Union, Intersection , Defference)
.Structured Query Language

10. **LECTURE 10:** List Comprehensions. NumPy
.List comprehensions (Sytax, Conditions, Iterable, expression, len() function, sum() functions)
.Sum to **COUNT**
.NumPy
.2D NumPy Array
.What is NumPy?
.Why use NumPy?
.Why is NumPy Faster than Lists
.Splitting NumPy Arrays
.Split into Arrays
.Splitting 2-D Arrays
.NumPy Searching Arrays
.NumPy Filter Array
.Creating Filter Directly

11. **LECTURE 11:** Multithreading and Client/Server Programming; introduction to HTML, interacting with remote HTML server, running html-based queries, downloading pages
.Multithreading
.Thread (Thread Identifier, Stack pointer, Program counter, Thread state, Thread's register set, Parent process Pointer)
.Clien/Server Programming (Client-Side Programming , Server-Side Programming)
.HTML (What is HTML)
.Web technologies
.Anatomy of HTML tag (EVERYTHING THAT IS RELATED TO TAGS)

.Structure of HTML page
.Page Structure Elements (<!DOCTYPE> , <html> , <head> , <tittle> , <body> , <h1>)
.Key Structural Elements (<h1>, <h2> , <p> , <div>)
.HTML links
.Absolute and Relative References
.An Image as Link and Link to Email Address
.Text Formatting (<b> , <strong> , <i> , <em>, <mark> , <small> , <del>, <ins>, <sub> , <sup>)
.HTML Lists (Ordered lists , Unordered lists , Description Lists)
.HTML Tables (<table> , <tr> , <th>, <td>)
.HTML Forms (<form> , <input>)
.Creating HTML with Python .Using Python to Control Browser

12. **LECTURE 12:** CSS for styling. Basic CSS. CSS properties. More CSS syntax.
.What is CSS?
.Power of CSS
.Problems HTML Formatting
.How CSS Fixes Formatting Problems
.Advantages of CSS
.CSS – Syntax (Selector , Property , Value **) → PAY ATTENTION**
.Types of Selectors (The Universal , Type Selector, ID selector , Class Selector)
.Properties (background-color (image) , border , color, display , float, font-(family| size| style| weight), margin ,padding , visibility)
.Padding
.Margin
.Three Methods of using CSS (In-line , Internal , External)
.In-Line (style attribute)
.Internal (style tag)
.External (separated file)

13. **LECTURE 13: Exceptions**
.Defensive Programming (Testing Validation, Debugging**)**
.When are you ready to test?
.Classes of Tests (Unit Testing , Regression Testing , Integration testing)
.Testing Approaches
.Black box testing (Without looking, biases, reused, paths)
.Glass Box Testing (Use code, path- complete, drawbacks)
.Debugging Steps (study code, scientific method)
.Kinds of Errors (**Syntax errors, Exceptions**)
.Syntax Errors
.Exceptions (IndexError , TypeError, NameError, SyntaxError)
.Sytax Error
.NameError
.AttributeError
.TypeError
.**ValuesError**
.IOError
.Try and Except Block
.Else
.Finally

14. **LECTURE 14:** Searching, Sorting, and Complexity Analysis
.Searching Algorithms (Membership Operators, Linear Search, Binary Search, Jump Search, Fibonacci Search, Exponential Search, Interpolation Search)
.Using Search Algorithms ОБЯЗАТЕЛЬНО ТАБЛИЦУ
.Sorting Algorithms (Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort)
.Sorting Algorithms CONCLUSION
.Complexity Analysis (Time Complexity, Space Complexity) (ТАБЛИЦА СО СЛОЖНОСТЬЮ)

METHODS

STRING METHODS:

capitalize() -- Converts the first character to upper case
casefold() -- Converts string into lower case. **Similar to lower() but Casefold() stronger**
center() -- Returns a centered string
count() -- Returns the number of times a specified value occurs in a string
encode() -- Returns an encoded version of the string
endswith() -- Returns true if the string ends with the specified value
expandtabs() -- Sets the tab size of the string
find() -- Searches the string for a specified value and returns the position of where it was found. Returns -1 if not found. **Almost the same as index(),** but **index() raises Exception if not Foun**
format() -- Formats specified values in a string
format_map() -- Formats specified values in a string
index() -- Searches the string for a specified value and returns the position of where it was found Raises an exception if the value is not found.**Almost the same as find()** but **find()** returns -1 if NO
Isalnum() -- Returns True if all characters in the string are alphanumeric (a-z) and (0-9)
Isalpha() -- Returns True if all characters in the string are in the alphabet (a-z)
Isdecimal() -- Returns True if all characters in the string are decimals (0-9)
Isdigit() -- Returns True if all characters in the string are digits
Isidentifier() -- Returns True if the string is an identifier if it only contains alphanumeric letters (a-z) and (0-9), or underscores (_).
islower() -- Returns True if all characters in the string are lower case
isnumeric() -- Returns True if all characters in the string are numeric
isprintable() -- Returns True if all characters in the string are printable
isspace() -- Returns True if all characters in the string are whitespaces
istitle() -- Returns True if the string follows the rules of a title. Each word start with an upper case letter
isupper() -- Returns True if all characters in the string are upper case
join() -- Joins the elements of an iterable to the end of the string
ljust() -- Returns a left justified version of the string. will left align the string
lower() -- Converts a string into lower case
lstrip() -- Returns a left trim version of the string
maketrans() -- Returns a translation table to be used in translations
partition() -- Returns a tuple where the string is parted into three parts
replace() -- Returns a string where a specified value is replaced with a specified value
rfind() -- Searches the string for a specified value and returns the last position of where it was found. Returns -1 if not found. Almost the same as **rindex(). Rindex() will return exception if not found**

rindex() -- Searches the string for a specified value and returns the last position of where it was found. Will raise an exception if not found. Almost the same as **rfind(). Rfind() will return -1 if not found**

rjust() -- Returns a right justified version of the string will right align the string

rpartition() -- Returns a tuple where the string is parted into three parts

rsplit() -- Splits the string at the specified separator, and returns a list

rstrip() -- Returns a right trim version of the string

split() -- Splits the string at the specified separator, and returns a list

splitlines() -- Splits the string at line breaks and returns a list

startswith() -- Returns true if the string starts with the specified value

strip() -- Returns a trimmed version of the string. Remove spaces at the beginning and end

swapcase() -- Swaps cases, lower case becomes upper case and vice versa

title() -- Converts the first character of each WORD to upper case

translate() -- Returns a translated string

upper() -- Converts a string into upper case

zfill() -- Fills the string with a specified number of 0 values at the beginning

LIST METHODS:

append() -- Adds an element at the end of the list

clear() -- Removes all the elements from the list

copy() -- Returns a copy of the list

count() -- Returns the number of elements with the specified value

extend() -- Add the elements of a list (or any iterable LIST), to the end of the current list

index() -- Returns the index of the first element with the specified value

insert() -- Adds an element at the specified position

pop() -- Removes the element at the specified position

remove() -- Removes the the first occurrence of the element with the specified value.

reverse() -- Reverses the order of the list

sort() -- Sorts the list

TUPLE METHODS:

count() -- Returns the number of times a specified value occurs in a tuple

index() -- Searches the tuple for a specified value and returns the position of where it was found. Raises an exception if the value is not found.

SET METHODS:

add() -- Adds an element to the set. If there → do not add the element

clear() -- Removes all the elements from the set

copy() -- Returns a copy of the set

difference() -- Returns a set containing the difference between two or more sets

difference_update() -- Removes the items in this set that are also included in another, specified set

discard() -- Remove the specified item. Similar to remove(), but **remove() raises an error if no** while **discard no.**

intersection() -- Returns a set, that is the intersection of two other sets

intersection_update() -- Removes the items in this set that are not present in other, specified set(s)

isdisjoint() -- Returns whether two sets have a intersection or not

issubset() -- Returns whether another set contains this set or not

issuperset() -- Returns whether this set contains another set or not

pop() -- Removes an element from the set (random item) (method returns the removed item.)

remove() -- Removes the specified element. Similar to **discard(),**but **remove() raises error if not item** while **discard() doesn't do it**

symmetric_difference() -- Returns a set with the symmetric differences of two sets

symmetric_difference_update() -- inserts the symmetric differences from this set and another

union() -- Return a set containing the union of sets

update() -- Update the set with the union of this set and others by adding items from another set (or any other iterable).

DICTIONARY METHODS:

clear() -- Removes all the elements from the dictionary

copy() -- Returns a copy of the dictionary

fromkeys() -- Returns a dictionary with the specified keys and value

get() -- Returns the value of the specified key

items() -- Returns a list containing a tuple for each key value pair

keys() -- Returns a list containing the dictionary's keys

pop() -- Removes the element with the specified key. removed item is the return value

popitem() -- Removes the last inserted key-value pair removed item is the return as a tuple

setdefault() -- Returns the value of the specified key. If the key does not exist: insert the key, with the specified value

update() -- Updates the dictionary with the specified key-value pairs.Specified items can be a dictionary

values() -- Returns a list of all the values in the dictionary