

**DEPARTMENT OF COMPUTER SCIENCE &IT SCHOOL OF ENGINEERING &IT**

This is to certify that Ms./Mr.

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Fatima Khan\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Reg. No. ..210101060.. has satisfactorily completed the lab exercises prescribed for INTERNET TECHNOLOGY LAB [CSE 3262] of Third Year B. Tech. (Computer Science and Engineering) Degree at MIT, Manipal, in the academic year 2023-2024.

# Date: 31-May-2022



# Faculty: Zaheeruddin Ahmed

**CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **LAB NO.** | **TITLE** | **PAGE NO.** | **REMARKS** |
|  | Course Objectives and Outcomes | i |  |
|  | Evaluation plan | i |  |
|  | Instructions to the Students | ii |  |
| 1 | JQuery | 1 – 7 |  |
| 3 | Python Basics | 13 – 31 |  |
| 4 | Python Objects and Classes | 32 – 41 |  |
| 13 | References | 100 |  |
|  |  |  |  |

**Course Objectives**

* Acquire in-depth understanding of web application architecture.
* Understand techniques to improve user experience in web applications.
* Gain knowledge about how to interact with database and ReST API’s.

**Course Outcomes**

At the end of this course, students will be able to

* Ability to develop a basic website using a modern web development tool.
* Ability to design websites with better look and feel.
* Ability to create real-world web applications that interacts with database and ReST API’s.

**Evaluation plan**

* Internal Assessment Marks : 60%

➢ Continuous Evaluation : 40%

Continuous evaluation component (for each evaluation):5 marks

The assessment will depend on punctuality, program execution, maintaining the observation note and answering the questions in viva voce. ➢ Project Evaluation : 20%

* End semester assessment of two hour duration: 40 %

**INSTRUCTIONS TO THE STUDENTS**

**Pre- Lab Session Instructions**

* 1. Students should carry the Lab Manual Book and the required stationery to every lab session
  2. Be in time and follow the institution dress code
  3. Must Sign in the log register provided
  4. Make sure to occupy the allotted seat and answer the attendance
  5. Adhere to the rules and maintain the decorum
  6. Students must come prepared for the lab in advance

**In- Lab Session Instructions**

* Follow the instructions on the allotted exercises
* Show the program and results to the instructors on completion of experiments
* On receiving approval from the instructor, copy the program and results in the Lab record
* Prescribed textbooks and class notes can be kept ready for reference if required

**General Instructions for the exercises in Lab**

* Implement the given exercise individually and not in a group.
* The programs should meet the following criteria:

o Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs. o Observation book should be complete with program, proper input output clearly showing the parallel execution in each process.

* Plagiarism (copying from others) is strictly prohibited and would invite severe penalty in evaluation.
* The exercises for each week are divided under three sets:
  + - Solved example
    - Lab exercises - to be completed during lab hours
    - Additional Exercises - to be completed outside the lab or in the lab to enhance the skill
* In case a student misses a lab class, he/ she must ensure that the experiment is completed during the repetition lab with the permission of the faculty concerned but credit will be given only to one day’s experiment(s).
* Questions for lab tests and examination are not necessarily limited to the questions in the manual, but may involve some variations and / or combinations of the questions.
* A sample note preparation is given as a model for observation.

**THE STUDENTS SHOULD NOT**

* Bring mobile phones or any other electronic gadgets to the lab.
* Go out of the lab without permission.

**Lab No : 1 Date:**

# JQuery

**Objectives:**

In this lab, student will be able to

1. Develop responsive web pages using jquery
2. Familiarize with DOM manipulation and animations

jQuery is a fast and concise JavaScript library to develop web based application.

Here is the list of important core features supported by jQuery −

* *DOM manipulation* − The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle.
* *Event handling* − The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
* *AJAX Support* − The jQuery eases developing a responsive and feature rich site using AJAX technology.
* *Animations* − The jQuery comes with plenty of built-in animation effects which you can use in your websites.
* *Lightweight* − The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
* *Cross Browser Support* − The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
* *Latest Technology* − The jQuery supports CSS3 selectors and basic XPath syntax.

You can download jQuery library from <https://jquery.com/download/>on your local machine and include it in your HTML code.

Solved Example:

<html>

<head>

<title>The jQuery Example</title>

<script type = "text/javascript" src = "jquery-3.4.1.js">

</script>

<script type = "text/javascript" language = "javascript">

$(document).ready(function() {

$("div").click(function() {alert("Hello, world!");});

});

</script>

</head>

<body>

<div id = "mydiv">

Click on this to see a dialogue box.

</div> </body>

</html>

A good rule of thumb is to put your JavaScript programming (all your <script> tags) after any other content inside the <head> tag, but before the closing </head> tag. The *$(document).ready()* function is a built-in jQuery function that waits until the HTML for a page loads before it runs your script.

When a web browser loads an HTML file, it displays the contents of that file on the screen and also the web browser remembers the HTML tags, their attributes, and the order in which they appear in the file—this representation of the page is called the *Document Object Model*, or DOM for short.

***Selector:*** jQuery offers a very powerful technique for selecting and working on a collection of elements—CSS selectors. The basic syntax is like this:

$('selector') use a CSS class selector like this:

$('.submenu')

<script type = "text/javascript" language = "javascript">

$(document).ready(function() {

$("p").css("background-color", "yellow");

$("#myid").css("background-color", "red");

});

</script>

</head>

<body>

<div>

<p class = "myclass">This is a paragraph.</p>

<p id = "myid">This is second paragraph.</p>

<p>This is third paragraph.</p>

</div> </body>

We can select tag available with the given class in the DOM. For example $('.someclass') selects all elements in the document that have a class name as some-class.

***Get And Set Atrributes:***

<script type = "text/javascript" language = "javascript">

$(document).ready(function() { var title = $("p").attr("title");

$("#divid").text(title);

$("#myimg").attr("src", "/jquery/images/jquery.jpg");

});

</script>

</head>

<body>

<div>

<p title = "Bold and Brave">This is first paragraph.</p>

<p id = "myid">This is second paragraph.</p>

<div id = "divid"></div>

<img id = "myimg" alt = "Sample image" />

</div> </body>

</html>

You can replace a complete DOM element with the specified HTML or DOM elements.

*selector.replaceWith( content )*

<script type = "text/javascript" language = "javascript">

$(document).ready(function() {

$("div").click(function () {

$(this).replaceWith("<h1>JQuery is Great</h1>");

});

});

</script>

## *Events*

To make your web page interactive, you write programs that respond to events.

Mouse events: click, dblclick, mousedown, mouseup, mouseover, etc

Document/Window Events: load, resize, scroll, unload etc

Form Events: submit, reset, focus, and change

<script type = "text/javascript" language = "javascript">

$(document).ready(function() {

$('#button').click(function() {

$(this).val("Stop that!");

}); // end click

});

</script>

</head>

<body>

<div id = "mydiv">

Click on this to see a dialogue box.

<input type="button" id="button">

</div>

</body>

• The hover( over, out ) method simulates hovering (moving the mouse on, and off, an object).

<script type = "text/javascript" language = "javascript">

$(document).ready(function() {

$('div').hover(

function () {

$(this).css({"background-color":"red"});

},

function () {

$(this).css({"background-color":"blue"});

}

);

}); </script>

The bind() method is a more flexible way of dealing with events than jQuery’s event specific functions like click() or mouseover(). It not only lets you specify an event and a function to respond to the event, but also lets you pass additional data for the event-handling function to use.

$('#theElement').bind('click', function() {

// do something interesting

}); // end bind

• checked selector selects all checked check-boxes or radio buttons. Let us understand this with an example.

<html>

<head>

<title></title>

<script src="jquery-1.11.2.js"></script>

<script type="text/javascript">

$(document).ready(function () { $('#btnSubmit').click(function () {

var result = $('input[type="radio"]:checked'); if (result.length > 0) {

$('#divResult').html(result.val() + " is checked");

} else {

$('#divResult').html("No radio button checked");

}

});

});

</script>

</head>

<body style="font-family:Arial"> Gender :

<input type="radio" name="gender" value="Male">Male

<input type="radio" name="gender" value="Female">Female

<input id="btnSubmit" type="submit" value="submit" />

<div id="divResult">

</div>

</body>

</html>

• The each() method in jQuery is used to execute a function for each matched element.

<html>

<head>

<title></title>

<script src="jquery-1.11.2.js"></script>

<script type="text/javascript">

$(document).ready(function () { $('#btnSubmit').click(function () {

var result = $('input[type="checkbox"]:checked'); if (result.length > 0) {

var resultString = result.length + " checkboxe(s) checked<br/>"; result.each(function () {

resultString += $(this).val() + "<br/>";

});

$('#divResult').html(resultString);

}

else {

$('#divResult').html("No checkbox checked");

}

});

});

</script>

</head>

<body style="font-family:Arial">

Skills :

<input type="checkbox" name="skills" value="JavaScript" />JavaScript

<input type="checkbox" name="skills" value="jQuery" />jQuery

<input type="checkbox" name="skills" value="C#" />C#

<input type="checkbox" name="skills" value="VB" />VB <br /><br />

<input id="btnSubmit" type="submit" value="submit" />

<br /><br />

<div id="divResult">

</div>

</body>

</html>

## *The animate() Method*

The jQuery animate() method is used to create custom animations.

$(*selector*).animate({*params*}*,speed,callback*);

The required params parameter defines the CSS properties to be animated.

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the animation completes.

$("button").click(function(){

$("div").animate({left:'250px'});

});

**Exercises:**

### Write a web page which contains table with 3 X 3 dimensions (fill some data) and one image. Style the rows with alternate color. Move the table and image together from right to left when button is clicked.

<html>

<head>

<title>Applications of Cognitive Computing</title>

<style>

  table {

    width: 50%;

    margin: 20px auto;

  }

  th, td {

    border: 1px solid black;

    padding: 8px;

    text-align: center;

  }

  th {

    background-color: #f2f2f2;

  }

  tr:nth-child(even) {

    background-color: #ddd;

  }

  img {

    display: block;

    width: 50%;

    margin: 0 auto;

  }

  h1 {

    text-align: center;

  }

</style>

</head>

<body>

</br>

<img id="mimg" src="C:\Users\Fatima Khan\OneDrive - Manipal Education (MENA) FZ LLC\Desktop\6th sem\ITLAB\Experiment 1\CogComp.jpeg">

<h1>

    Applications Of Cognitive Computing.

</h1>

<table id="table">

  <tr>

    <th>Ref No.</th>

    <th>Case Study</th>

    <th>Purpose</th>

  </tr>

  <tr>

    <td>IBM Journals</td>

    <td>Watson</td>

    <td>Diagnostics and Treatment</td>

  </tr>

  <tr>

    <td>5G Paper</td>

    <td>5G-Health-Cys</td>

    <td>Emotional Monnitoring and Televized Medecine</td>

  </tr>

  <tr>

    <td>ECC Paper</td>

    <td>Edge Cognitive Computing</td>

    <td>Real-Time Patient Monitoring</td>

  </tr>

</table>

<button onclick="moveIT()">Move Image and Table</button>

<script>

  function moveIT() {

    var t = document.querySelector('table');

    var i = document.querySelector('img');

    var pos = 0;

    for(var i=0; ;i++){

        if(i%5000==0){

            if (pos == 100) {

                return;

              }

            else {

                pos++;

                table.style.marginLeft = pos + 'px';

                mimg.style.marginLeft = pos + 'px';

            }

        }

    }

  }

</script>

</body>

</html>

A screenshot of a computer

Description automatically generated

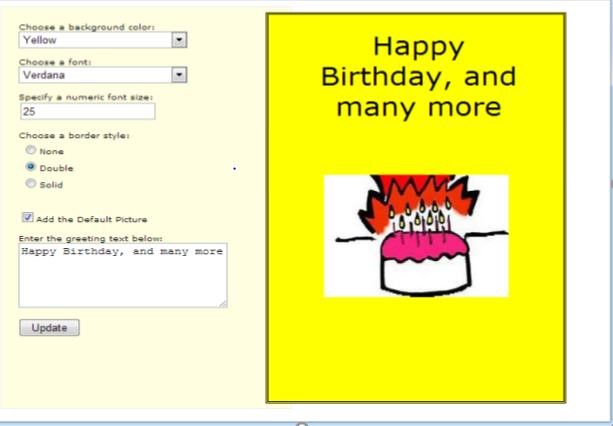
### Design a calculator to perform basic arithmetic operations. Use textboxes and buttons to design web page.

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Basic Calculator</title>
5. <style>
6. .container {
7. width: 300px;
8. margin: 50px auto;
9. padding: 20px;
10. border: 1px solid #ccc;
11. border-radius: 10px;
12. text-align: center;
13. background-color: lightpink;
14. }
15. input[type="text"] {
16. width: 50px;
17. text-align: right;
18. }
19. </style>
20. <script>
21. document.addEventListener("DOMContentLoaded", function() {
22. function calculate(operation) {
23. var num1 = parseFloat(document.getElementById('num1').value);
24. var num2 = parseFloat(document.getElementById('num2').value);
25. var res;
27. if (operation === 'add') {
28. res = num1 + num2;
29. } else if (operation === 'sub') {
30. res = num1 - num2;
31. } else if (operation === 'mul') {
32. res = num1 \* num2;
33. } else if (operation === 'div') {
34. res = num2 !== 0 ? num1 / num2 : 'Error';
35. }
36. document.getElementById('result').value = res;
37. }
38. document.getElementById('addBtn').addEventListener('click', function() { calculate('add'); });
39. document.getElementById('subBtn').addEventListener('click', function() { calculate('sub'); });
40. document.getElementById('mulBtn').addEventListener('click', function() { calculate('mul'); });
41. document.getElementById('divBtn').addEventListener('click', function() { calculate('div'); });
42. });
43. </script>
44. </head>
45. <body>
46. <div class="container">
47. <h2>Calculator</h2>
48. <form id="calcForm">
49. <input type="text" id="num1"> +
50. <input type="text" id="num2">
51. <br><br>
52. <button type="button" id="addBtn">Add</button>
53. <button type="button" id="subBtn">Subtract</button>
54. <button type="button" id="mulBtn">Multiply</button>
55. <button type="button" id="divBtn">Divide</button>
56. <br><br>
57. <input type="text" id="result" readonly>
58. </form>
59. </div>
60. </body>
61. </html>

A screenshot of a calculator

Description automatically generated

### 3) Create a web page to design a birthday card shown below.



<!DOCTYPE html>

<html>

<head>

    <title>Birthday Card</title>

    <style>

        body {

            font-family: Arial, sans-serif;

        }

        .form-container {

            width: 300px;

            float: left;

            padding: 20px;

            border-radius: 10px;

        }

        .card-container {

            width: 300px;

            height: 400px;

            float: left;

            margin-left: 20px;

            text-align: center;

            border: 2px double black;

            background-color: yellow;

            padding: 20px;

        }

        .card-container img {

            width: 100px;

            height: 100px;

        }

    </style>

    <script>

        document.addEventListener("DOMContentLoaded", function() {

            function updateCard() {

                var bgcolor = document.getElementById('bgcolor').value;

                var font = document.getElementById('font').value;

                var fontsize = document.getElementById('fontsize').value + 'px';

                var borderStyle = document.querySelector('input[name="border"]:checked').value;

                var message = document.getElementById('message').value;

                var defaultImg = document.getElementById('defaultImg').checked;

                var card = document.getElementById('card');

                card.style.backgroundColor = bgcolor;

                card.style.fontFamily = font;

                card.style.fontSize = fontsize;

                card.style.borderStyle = borderStyle;

                document.getElementById('greetingText').innerText = message;

                if (defaultImg) {

                    document.getElementById('greetingImage').style.display = 'block';

                } else {

                    document.getElementById('greetingImage').style.display = 'none';

                }

            }

            document.getElementById('updateBtn').addEventListener('click', updateCard);

        });

    </script>

</head>

<body>

    <div class="form-container">

        <label for="bgcolor">Choose a background color:</label><br>

        <select id="bgcolor">

            <option value="yellow">Yellow</option>

            <option value="pink">Pink</option>

            <option value="lightblue">Light Blue</option>

        </select><br><br>

        <label for="font">Choose a font:</label><br>

        <select id="font">

            <option value="Arial">Arial</option>

            <option value="Verdana">Verdana</option>

            <option value="Times New Roman">Times New Roman</option>

        </select><br><br>

        <label for="fontsize">Specify a numeric font size:</label><br>

        <input type="number" id="fontsize" value="25"><br><br>

        <label>Choose a border style:</label><br>

        <input type="radio" name="border" value="none"> None<br>

        <input type="radio" name="border" value="double" checked> Double<br>

        <input type="radio" name="border" value="solid"> Solid<br><br>

        <input type="checkbox" id="defaultImg" checked> Add the Default Picture<br><br>

        <label for="message">Enter the greeting text below:</label><br>

        <textarea id="message" rows="4" cols="30">Happy Birthday, and many more</textarea><br><br>

        <button id="updateBtn">Update</button>

    </div>

    <div class="card-container" id="card">

        <p id="greetingText">Happy Birthday, and many more</p>

        <img id="greetingImage" src="ss.png" alt="Birthday Cake">

    </div>

</body>

</html>

A screenshot of a computer

Description automatically generated

### 4) Design a webpage. The page contains:

* + Dropdown list with HP, Nokia, Samsung, Motorola, Apple as items.
  + Checkbox with Mobile and Laptop as items.
  + Textbox where you enter quantity.
  + There is a button with text as ‘Produce Bill’.

On Clicking Produce Bill button, alert should be displayed with total amount.

<!DOCTYPE html>

<html>

<head>

<title>Product Order Form</title>

<style>

.container {

width: 300px;

margin: 50px auto;

padding: 20px;

border: 1px solid #ccc;

border-radius: 10px;

background-color: lightpink;

}

</style>

<script>

document.addEventListener("DOMContentLoaded", function() {

document.getElementById('billBtn').addEventListener('click', function() {

var product = document.getElementById('product').value;

var quantity = document.getElementById('quantity').value;

var categories = [];

if (document.getElementById('mobile').checked) categories.push('Mobile');

if (document.getElementById('laptop').checked) categories.push('Laptop');

var totalAmount = quantity \* 100; // Assuming each product costs 100 for simplicity

alert('Product: ' + product + '\nCategory: ' + categories.join(', ') + '\nQuantity: ' + quantity + '\nTotal Amount: ' + totalAmount);

});

});

</script>

</head>

<body>

<div class="container">

<h2>Order Form</h2>

<form id="orderForm">

<label for="product">Choose a product:</label><br>

<select id="product">

<option value="HP">HP</option>

<option value="Nokia">Nokia</option>

<option value="Samsung">Samsung</option>

<option value="Motorola">Motorola</option>

<option value="Apple">Apple</option>

</select><br><br>

<label for="category">Choose a category:</label><br>

<input type="checkbox" id="mobile" value="Mobile"> Mobile<br>

<input type="checkbox" id="laptop" value="Laptop"> Laptop<br><br>

<label for="quantity">Enter quantity:</label><br>

<input type="number" id="quantity" min="1" value="1"><br><br>

<button type="button" id="billBtn">Produce Bill</button>

</form>

</div>

</body>

</html>

A screenshot of a computer screen

Description automatically generated

### Additional Exercise:

### Implement the bouncing ball using animate() function.

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Bouncing Ball</title>
5. <style>
6. body {
7. margin: 0;
8. overflow: hidden;
9. }
10. #ball {
11. width: 50px;
12. height: 50px;
13. background-color: red;
14. border-radius: 50%;
15. position: absolute;
16. top: 0;
17. left: 50%;
18. }
19. </style>
20. <script>
21. document.addEventListener("DOMContentLoaded", function() {
22. var ball = document.getElementById('ball');
23. var direction = 1;
24. var speed = 2;
25. var gravity = 0.2;
26. var bounceFactor = 0.7;
27. var ballTop = 0;
28. var velocity = 0;
29. function animateBall() {
30. var windowHeight = window.innerHeight;
31. velocity += gravity;
32. ballTop += velocity;
33. if (ballTop + ball.offsetHeight >= windowHeight) {
34. ballTop = windowHeight - ball.offsetHeight;
35. velocity \*= -bounceFactor;
36. }
37. if (ballTop <= 0) {
38. ballTop = 0;
39. velocity \*= -bounceFactor;
40. }
41. ball.style.top = ballTop + 'px';
42. requestAnimationFrame(animateBall);
43. }
44. animateBall();
45. });
46. </script>
47. </head>
48. <body>
49. <div id="ball"></div>
50. </body>
51. </html>

A screenshot of a computer

Description automatically generated

### Write a web page which displays image and show the sliding text on the image.

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Sliding Text on Image</title>
5. <style>
6. .container {
7. position: relative;
8. width: 50%;
9. }
10. .image {
11. display: block;
12. width: 100%;
13. height: auto;
14. }
15. .text {
16. position: absolute;
17. bottom: 0;
18. background: rgba(0, 0, 0, 0.5);
19. color: #f1f1f1;
20. width: 100%;
21. text-align: center;
22. animation: slideText 5s infinite;
23. }
24. @keyframes slideText {
25. 0% {bottom: -50px;}
26. 50% {bottom: 0;}
27. 100% {bottom: -50px;}
28. }
29. </style>
30. </head>
31. <body>
32. <div class="container">
33. <img src="ss.png" alt="Sample Image" class="image">
34. <div class="text">Sliding Text</div>
35. </div>
36. </body>
37. </html>



**Lab No:3 Date:**

# Python Basics

**Objectives:**

In this lab, student will be able to

1. Familiarize with the python programming language
2. Understand the usage of python primitives, data structures and functions

Python was created by Guido van Rossum in the early 90s. It is now one of the most popular languages in existence.

# Single line comments start with a number symbol.

""" Multiline strings can be written using three "s, and are often used as documentation.

"""

####################################################

## 1. Primitive Datatypes and Operators

####################################################

# You have numbers

3 # => 3

# Math is what you would expect

1 + 1 # => 2

8 - 1 # => 7

10 \* 2 # => 20

35 / 5 # => 7.0

# Integer division rounds down for both positive and negative numbers. 5 // 3 # => 1

1. // 3 # => -2

5.0 // 3.0 # => 1.0 # works on floats too

-5.0 // 3.0 # => -2.0

# The result of division is always a float

10.0 / 3 # => 3.3333333333333335

# Modulo operation

7 % 3 # => 1

# Exponentiation (x\*\*y, x to the yth power)

2\*\*3 # => 8

# Enforce precedence with parentheses

1 + 3 \* 2 # => 7

(1 + 3) \* 2 # => 8

# Boolean values are primitives (Note: the capitalization)

True # => True

False # => False

# negate with not not True # => False

not False # => True

# Boolean Operators

# Note "and" and "or" are case-sensitive

True and False # => False

False or True # => True

# True and False are actually 1 and 0 but with different keywords

True + True # => 2

True \* 8 # => 8

False - 5 # => -5

# Comparison operators look at the numerical value of True and False

0 == False # => True 1 == True # => True

2 == True # => False

-5 != False # => True

# Using boolean logical operators on ints casts them to booleans for evaluation, but their non-cast value is returned

# Don't mix up with bool(ints) and bitwise and/or (&,|) bool(0) # => False

bool(4) # => True bool(-6) # => True 0 and 2 # => 0

-5 or 0 # => -5

# Equality is ==

1. == 1 # => True
2. == 1 # => False

# Inequality is !=

1. != 1 # => False
2. != 1 # => True

# More comparisons

1 < 10 # => True

1. > 10 # => False
2. <= 2 # => True

2 >= 2 # => True

# Seeing whether a value is in a range

1. < 2 and 2 < 3 # => True
2. < 3 and 3 < 2 # => False

# Chaining makes this look nicer

1. < 2 < 3 # => True
2. < 3 < 2 # => False

# (is vs. ==) is checks if two variables refer to the same object, but == checks # if the objects pointed to have the same values. a = [1, 2, 3, 4] # Point a at a new list, [1, 2, 3, 4] b = a # Point b at what a is pointing to b is a # => True, a and b refer to the same object b == a # => True, a's and b's objects are equal b = [1, 2, 3, 4] # Point b at a new list, [1, 2, 3, 4]

b is a # => False, a and b do not refer to the same object b == a # => True, a's and b's objects are equal

# Strings are created with " or '

"This is a string."

'This is also a string.'

# Strings can be added too! But try not to do this.

"Hello " + "world!" # => "Hello world!"

# String literals (but not variables) can be concatenated without using '+'

"Hello " "world!" # => "Hello world!"

# A string can be treated like a list of characters

"This is a string"[0] # => 'T'

# You can find the length of a string

len("This is a string") # => 16

# You can also format using f-strings or formatted string literals (in Python 3.6+) name = "Reiko"

f"She said her name is {name}." # => "She said her name is Reiko"

# You can basically put any Python statement inside the braces and it will be output in the string.

f"{name} is {len(name)} characters long." # => "Reiko is 5 characters long."

# None is an object

None # => None

# Don't use the equality "==" symbol to compare objects to None # Use "is" instead. This checks for equality of object identity.

"etc" is None # => False

None is None # => True

# None, 0, and empty strings/lists/dicts/tuples all evaluate to False.

# All other values are True bool(0) # => False bool("") # => False bool([]) # => False bool({}) # => False

bool(()) # => False

####################################################

## 2. Variables and Collections

####################################################

# Python has a print function print("I'm Python. Nice to meet you!") # => I'm Python. Nice to meet you!

# By default the print function also prints out a newline at the end.

# Use the optional argument end to change the end string.

print("Hello, World", end="!") # => Hello, World!

# Simple way to get input data from console input\_string\_var = input("Enter some data: ") # Returns the data as a string # Note: In earlier versions of Python, input() method was named as raw\_input()

# There are no declarations, only assignments. # Convention is to use lower\_case\_with\_underscores some\_var = 5

some\_var # => 5

# Accessing a previously unassigned variable is an exception.

# See Control Flow to learn more about exception handling.

some\_unknown\_var # Raises a NameError

# if can be used as an expression

# Equivalent of C's '?:' ternary operator

"yahoo!" if 3 > 2 else 2 # => "yahoo!"

# Lists store sequences li = []

# You can start with a prefilled list other\_li = [4, 5, 6]

# Add stuff to the end of a list with append li.append(1) # li is now [1] li.append(2) # li is now [1, 2] li.append(4) # li is now [1, 2, 4] li.append(3) # li is now [1, 2, 4, 3] # Remove from the end with pop

li.pop() # => 3 and li is now [1, 2, 4]

# Let's put it back li.append(3) # li is now [1, 2, 4, 3] again.

# Access a list like you would any array

li[0] # => 1

# Look at the last element

li[-1] # => 3

# Looking out of bounds is an IndexError li[4] # Raises an IndexError

# You can look at ranges with slice syntax.

# The start index is included, the end index is not # (It's a closed/open range for you mathy types.) li[1:3] # Return list from index 1 to 3 => [2, 4] li[2:] # Return list starting from index 2 => [4, 3] li[:3] # Return list from beginning until index 3 => [1, 2, 4] li[::2] # Return list selecting every second entry => [1, 4] li[::-1] # Return list in reverse order => [3, 4, 2, 1]

# Use any combination of these to make advanced slices

# li[start:end:step]

# Make a one layer deep copy using slices li2 = li[:] # => li2 = [1, 2, 4, 3] but (li2 is li) will result in false.

# Remove arbitrary elements from a list with "del"

del li[2] # li is now [1, 2, 3]

# Remove first occurrence of a value li.remove(2) # li is now [1, 3]

li.remove(2) # Raises a ValueError as 2 is not in the list

# Insert an element at a specific index

li.insert(1, 2) # li is now [1, 2, 3] again

# Get the index of the first item found matching the argument li.index(2) # => 1

li.index(4) # Raises a ValueError as 4 is not in the list

# You can add lists

# Note: values for li and for other\_li are not modified. li + other\_li # => [1, 2, 3, 4, 5, 6]

# Concatenate lists with "extend()" li.extend(other\_li) # Now li is [1, 2, 3, 4, 5, 6]

# Check for existence in a list with "in"

1 in li # => True

# Examine the length with "len()"

len(li) # => 6

# Tuples are like lists but are immutable. tup = (1, 2, 3) tup[0] # => 1

tup[0] = 3 # Raises a TypeError

# Note that a tuple of length one has to have a comma after the last element but # tuples of other lengths, even zero, do not. type((1)) # => <class 'int'> type((1,)) # => <class 'tuple'>

type(()) # => <class 'tuple'>

# You can do most of the list operations on tuples too len(tup) # => 3

tup + (4, 5, 6) # => (1, 2, 3, 4, 5, 6) tup[:2] # => (1, 2)

2 in tup # => True

# You can unpack tuples (or lists) into variables a, b, c = (1, 2, 3) # a is now 1, b is now 2 and c is now 3

# You can also do extended unpacking

a, \*b, c = (1, 2, 3, 4) # a is now 1, b is now [2, 3] and c is now 4 # Tuples are created by default if you leave out the parentheses d, e, f = 4, 5, 6 # tuple 4, 5, 6 is unpacked into variables d, e and f

# respectively such that d = 4, e = 5 and f = 6 # Now look how easy it is to swap two values e, d = d, e # d is now 5 and e is now 4

# Dictionaries store mappings from keys to values empty\_dict = {}

# Here is a prefilled dictionary

filled\_dict = {"one": 1, "two": 2, "three": 3}

# Note keys for dictionaries have to be immutable types. This is to ensure that # the key can be converted to a constant hash value for quick look-ups.

# Immutable types include ints, floats, strings, tuples. invalid\_dict = {[1,2,3]: "123"} # => Raises a TypeError: unhashable type: 'list' valid\_dict = {(1,2,3):[1,2,3]} # Values can be of any type, however.

# Look up values with [] filled\_dict["one"] # => 1 # Get all keys as an iterable with "keys()". We need to wrap the call in list()

# to turn it into a list. We'll talk about those later. Note - for Python

# versions <3.7, dictionary key ordering is not guaranteed. Your results might # not match the example below exactly. However, as of Python 3.7, dictionary # items maintain the order at which they are inserted into the dictionary.

list(filled\_dict.keys()) # => ["three", "two", "one"] in Python <3.7 list(filled\_dict.keys()) # => ["one", "two", "three"] in Python 3.7+

# Get all values as an iterable with "values()". Once again we need to wrap it # in list() to get it out of the iterable. Note - Same as above regarding key # ordering. list(filled\_dict.values()) # => [3, 2, 1] in Python <3.7 list(filled\_dict.values()) # => [1, 2, 3] in Python 3.7+

# Check for existence of keys in a dictionary with "in"

"one" in filled\_dict # => True

1 in filled\_dict # => False

# Looking up a non-existing key is a KeyError filled\_dict["four"] # KeyError

# Use "get()" method to avoid the KeyError

filled\_dict.get("one") # => 1 filled\_dict.get("four") # => None

# The get method supports a default argument when the value is missing filled\_dict.get("one", 4) # => 1

filled\_dict.get("four", 4) # => 4

# "setdefault()" inserts into a dictionary only if the given key isn't present filled\_dict.setdefault("five", 5) # filled\_dict["five"] is set to 5 filled\_dict.setdefault("five", 6) # filled\_dict["five"] is still 5

# Adding to a dictionary

filled\_dict.update({"four":4}) # => {"one": 1, "two": 2, "three": 3, "four": 4} filled\_dict["four"] = 4 # another way to add to dict

# Remove keys from a dictionary with del

del filled\_dict["one"] # Removes the key "one" from filled dict

# From Python 3.5 you can also use the additional unpacking options {'a': 1, \*\*{'b': 2}} # => {'a': 1, 'b': 2}

{'a': 1, \*\*{'a': 2}} # => {'a': 2}

# Sets store ... well sets empty\_set = set()

# Initialize a set with a bunch of values. Yeah, it looks a bit like a dict. Sorry. some\_set = {1, 1, 2, 2, 3, 4} # some\_set is now {1, 2, 3, 4}

# Similar to keys of a dictionary, elements of a set have to be immutable. invalid\_set = {[1], 1} # => Raises a TypeError: unhashable type: 'list' valid\_set = {(1,), 1}

# Add one more item to the set filled\_set = some\_set

filled\_set.add(5) # filled\_set is now {1, 2, 3, 4, 5}

# Sets do not have duplicate elements

filled\_set.add(5) # it remains as before {1, 2, 3, 4, 5}

# Do set intersection with & other\_set = {3, 4, 5, 6}

filled\_set & other\_set # => {3, 4, 5}

# Do set union with |

filled\_set | other\_set # => {1, 2, 3, 4, 5, 6}

# Do set difference with - {1, 2, 3, 4} - {2, 3, 5} # => {1, 4}

# Do set symmetric difference with ^

{1, 2, 3, 4} ^ {2, 3, 5} # => {1, 4, 5}

# Check if set on the left is a superset of set on the right

{1, 2} >= {1, 2, 3} # => False

# Check if set on the left is a subset of set on the right

{1, 2} <= {1, 2, 3} # => True

# Check for existence in a set with in

2 in filled\_set # => True

10 in filled\_set # => False

# Make a one layer deep copy

filled\_set = some\_set.copy() # filled\_set is {1, 2, 3, 4, 5} filled\_set is some\_set # => False

####################################################

## 3. Control Flow and Iterables

####################################################

# Let's just make a variable

some\_var = 5

# Here is an if statement. Indentation is significant in Python! # Convention is to use four spaces, not tabs. # This prints "some\_var is smaller than 10" if some\_var > 10: print("some\_var is totally bigger than 10.") elif some\_var < 10: # This elif clause is optional. print("some\_var is smaller than 10.") else: # This is optional too. print("some\_var is indeed 10.")

"""

For loops iterate over lists prints:

dog is a mammal cat is a mammal mouse is a mammal

""" for animal in ["dog", "cat", "mouse"]:

# You can use format() to interpolate formatted strings print("{} is a mammal".format(animal))

"""

"range(number)" returns an iterable of numbers from zero to the given number prints:

0

1

2

3

""" for i in range(4):

print(i)

"""

"range(lower, upper)" returns an iterable of numbers from the lower number to the upper number prints:

4

5

6

7 """ for i in range(4, 8):

print(i)

"""

"range(lower, upper, step)" returns an iterable of numbers from the lower number to the upper number, while incrementing by step. If step is not indicated, the default value is 1. prints:

4

6 """ for i in range(4, 8, 2):

print(i)

"""

To loop over a list, and retrieve both the index and the value of each item in the list prints: 0 dog

1. cat
2. mouse """ animals = ["dog", "cat", "mouse"] for i, value in enumerate(animals):

print(i, value)

"""

While loops go until a condition is no longer met.

prints:

0

1

2

1. """ x = 0 while x < 4: print(x)

x += 1 # Shorthand for x = x + 1

# Handle exceptions with a try/except block try:

# Use "raise" to raise an error raise IndexError("This is an index error") except IndexError as e: pass # Pass is just a no-op. Usually you would do recovery here. except (TypeError, NameError): pass # Multiple exceptions can be handled together, if required. else: # Optional clause to the try/except block. Must follow all except blocks print("All good!") # Runs only if the code in try raises no exceptions

finally: # Execute under all circumstances print("We can clean up resources here")

# Instead of try/finally to cleanup resources you can use a with statement with open("myfile.txt") as f: for line in f: print(line)

# Writing to a file contents = {"aa": 12, "bb": 21} with open("myfile1.txt", "w+") as file:

file.write(str(contents)) # writes a string to a file

with open("myfile2.txt", "w+") as file:

file.write(json.dumps(contents)) # writes an object to a file

# Reading from a file with open('myfile1.txt', "r+") as file:

contents = file.read() # reads a string from a file print(contents)

# print: {"aa": 12, "bb": 21}

with open('myfile2.txt', "r+") as file:

contents = json.load(file) # reads a json object from a file print(contents)

# print: {"aa": 12, "bb": 21}

# Python offers a fundamental abstraction called the Iterable.

# An iterable is an object that can be treated as a sequence. # The object returned by the range function, is an iterable.

filled\_dict = {"one": 1, "two": 2, "three": 3} our\_iterable = filled\_dict.keys() print(our\_iterable) # => dict\_keys(['one', 'two', 'three']). This is an object that implements our Iterable interface.

# We can loop over it. for i in our\_iterable: print(i) # Prints one, two, three

# However we cannot address elements by index. our\_iterable[1] # Raises a TypeError

# An iterable is an object that knows how to create an iterator. our\_iterator = iter(our\_iterable)

# Our iterator is an object that can remember the state as we traverse through it.

# We get the next object with "next()". next(our\_iterator) # => "one"

# It maintains state as we iterate. next(our\_iterator) # => "two"

next(our\_iterator) # => "three"

# After the iterator has returned all of its data, it raises a StopIteration exception next(our\_iterator) # Raises StopIteration

# We can also loop over it, in fact, "for" does this implicitly! our\_iterator = iter(our\_iterable) for i in our\_iterator:

print(i) # Prints one, two, three

# You can grab all the elements of an iterable or iterator by calling list() on it.

list(our\_iterable) # => Returns ["one", "two", "three"]

list(our\_iterator) # => Returns [] because state is saved

# The indention of each line controls whether it is within a loop, if statement, etc. -- there are no { } to define blocks of code. This use of indentation in Python is unusual, but it's logical and you get used to it.

####################################################

## 4. Functions

####################################################

# Use "def" to create new functions def add(x, y): print("x is {} and y is {}".format(x, y))

return x + y # Return values with a return statement

# Calling functions with parameters

add(5, 6) # => prints out "x is 5 and y is 6" and returns 11

# Another way to call functions is with keyword arguments add(y=6, x=5) # Keyword arguments can arrive in any order.

# You can define functions that take a variable number of

# positional arguments def varargs(\*args):

return args

varargs(1, 2, 3) # => (1, 2, 3)

# You can define functions that take a variable number of

# keyword arguments, as well def keyword\_args(\*\*kwargs):

return kwargs

# Let's call it to see what happens

keyword\_args(big="foot", loch="ness") # => {"big": "foot", "loch": "ness"}

# You can do both at once, if you like def all\_the\_args(\*args, \*\*kwargs):

print(args) print(kwargs)

"""

all\_the\_args(1, 2, a=3, b=4) prints:

(1, 2)

{"a": 3, "b": 4}

"""

# When calling functions, you can do the opposite of args/kwargs!

# Use \* to expand tuples and use \*\* to expand kwargs.

args = (1, 2, 3, 4) kwargs = {"a": 3, "b": 4} all\_the\_args(\*args) # equivalent to all\_the\_args(1, 2, 3, 4) all\_the\_args(\*\*kwargs) # equivalent to all\_the\_args(a=3, b=4)

all\_the\_args(\*args, \*\*kwargs) # equivalent to all\_the\_args(1, 2, 3, 4, a=3, b=4)

# Returning multiple values (with tuple assignments) def swap(x, y): return y, x # Return multiple values as a tuple without the parenthesis. # (Note: parenthesis have been excluded but can be included)

x = 1 y = 2 x, y = swap(x, y) # => x = 2, y = 1

# (x, y) = swap(x,y) # Again parenthesis have been excluded but can be included.

# Function Scope

x = 5

def set\_x(num):

# Local var x not the same as global variable x x = num # => 43

print(x) # => 43

def set\_global\_x(num):

global x print(x) # => 5

x = num # global var x is now set to 6

print(x) # => 6

set\_x(43)

set\_global\_x(6)

# Python has first class functions

def create\_adder(x): def adder(y): return x + y

return adder

add\_10 = create\_adder(10)

add\_10(3) # => 13

# There are also anonymous functions

(lambda x: x > 2)(3) # => True

(lambda x, y: x \*\* 2 + y \*\* 2)(2, 1) # => 5

# There are built-in higher order functions list(map(add\_10, [1, 2, 3])) # => [11, 12, 13] list(map(max, [1, 2, 3], [4, 2, 1])) # => [4, 2, 3]

list(filter(lambda x: x > 5, [3, 4, 5, 6, 7])) # => [6, 7]

# We can use list comprehensions for nice maps and filters

# List comprehension stores the output as a list which can itself be a nested list

[add\_10(i) for i in [1, 2, 3]] # => [11, 12, 13]

[x for x in [3, 4, 5, 6, 7] if x > 5] # => [6, 7]

# You can construct set and dict comprehensions as well.

{x for x in 'abcddeef' if x not in 'abc'} # => {'d', 'e', 'f'}

{x: x\*\*2 for x in range(5)} # => {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}

####################################################

## 5. Modules

####################################################

# You can import modules import math

print(math.sqrt(16)) # => 4.0

# You can get specific functions from a module from math import ceil, floor print(ceil(3.7)) # => 4.0

print(floor(3.7)) # => 3.0

# You can import all functions from a module.

# Warning: this is not recommended from math import \*

# You can shorten module names import math as m

math.sqrt(16) == m.sqrt(16) # => True

# Python modules are just ordinary Python files. You # can write your own, and import them. The name of the # module is the same as the name of the file.

# You can find out which functions and attributes # are defined in a module. import math

dir(math)

# If you have a Python script named math.py in the same # folder as your current script, the file math.py will # be loaded instead of the built-in Python module. # This happens because the local folder has priority # over Python's built-in libraries.

Example:

Write a python program to display content of a file.

#!/usr/bin/python -tt

"""

-tt flag above detects space/tab indent problems

"""

# sys is one of many available modules of library code, import to use.

# sys.argv is the list of command line arguments.

import sys

# defines a global variable a = 123

# defines a 'cat' function which takes a filename def cat(filename):

"""Given filename, print its text contents."""

print filename, '=======' f = open(filename, 'r')

for line in f: # goes through a text file line by line print line, # trailing comma inhibits the ending print-newline # alternative, read the whole file into a single string: # text = f.read() f.close()

def main():

# sys.argv contains command line arguments.

# This assigns a list of all but the first arg into a local 'args' var. args = sys.argv[1:]

# important syntax -- loop of variable 'filename' over the args list. for filename in args:

# detect scary filenames: if/else and/or/not if filename == 'voldemort' or filename == 'vader':

print 'this file is very worrying' cat(filemane, 123, bad\_variable)

# important point: errors in above line only caught if it is run else:

# regular case cat(filename)

print 'all done' # this print is outside the loop, due to its indentation

# Standard boilerplate at end of file to call main() function. if \_\_name\_\_ == '\_\_main\_\_':

main()

Exercises:

### Write a python program to implement simple calculator which perform addition, subtraction, multiplication, and division.

def add(x, y):

return x + y

def sub(x, y):

return x - y

def mul(x, y):

return x \* y

def div(x, y):

return x / y if y != 0 else "Error! Division by zero."

def calc():

print("1. Add\n2. Subtract\n3. Multiply\n4. Divide")

c = input("Choose(1/2/3/4): ")

a = float(input("First number: "))

b = float(input("Second number: "))

if c == '1':

print("Result:", add(a, b))

elif c == '2':

print("Result:", sub(a, b))

elif c == '3':

print("Result:", mul(a, b))

elif c == '4':

print("Result:", div(a, b))

else:

print("Invalid input")

calc()

A screenshot of a computer

Description automatically generated

### Write a python program to reverse a content a file and store it in another file.

def rev\_file(in\_file, out\_file):

with open(in\_file, 'r') as f:

data = f.read()

with open(out\_file, 'w') as f:

f.write(data[::-1])

rev\_file('test.txt', 'str1.txt')

A screenshot of a computer code

Description automatically generated

### Write a python program to implement binary search with recursion.

def bin\_search(arr, low, high, x):

if high >= low:

mid = (high + low) // 2

if arr[mid] == x:

return mid

elif arr[mid] > x:

return bin\_search(arr, low, mid - 1, x)

else:

return bin\_search(arr, mid + 1, high, x)

else:

return -1

arr = [2, 3, 4, 10, 40]

x = 10

res = bin\_search(arr, 0, len(arr) - 1, x)

print("Index:", res if res != -1 else "Not found")

A computer screen shot of a code

Description automatically generated

### Write a python program to sort words in alphabetical order.

def sort\_words(s):

words = s.split()

words.sort()

return ' '.join(words)

s = "hello world this is a test string"

print("Sorted:", sort\_words(s))

A screenshot of a computer code

Description automatically generated

Additional exercises

### Write a python program to select smallest element from a list in an expected linear time.

def find\_smallest(arr):

return min(arr)

# Example usage:

arr = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]

print("Smallest element:", find\_smallest(arr))

A screenshot of a computer

Description automatically generated

### Write a python program to implement bubble sort.

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

return arr

# Example usage:

arr = [64, 34, 25, 12, 22, 11, 90]

sorted\_arr = bubble\_sort(arr)

print("Sorted array:", sorted\_arr)

A screenshot of a computer code

Description automatically generated

### Write a python program to multiply two matrices

def multiply\_matrices(A, B):

result = [[0 for \_ in range(len(B[0]))] for \_ in range(len(A))]

for i in range(len(A)):

for j in range(len(B[0])):

for k in range(len(B)):

result[i][j] += A[i][k] \* B[k][j]

return result

# Example usage:

A = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]

B = [

[9, 8, 7],

[6, 5, 4],

[3, 2, 1]

]

result = multiply\_matrices(A, B)

print("Resultant matrix:")

for row in result:

print(row)

A screenshot of a computer program

Description automatically generated

**Lab No :4 Date:**

## Python Objects and Classes

**Objectives:**

In this lab, student will be able to

1. Understand the python classes
2. Learn how to use class objects and methods

Python is an object oriented programming language. Unlike procedure oriented

programming, where the main emphasis is on functions, object oriented programming

stresses on objects.

An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object.

We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

As many houses can be made from a house's blueprint, we can create many objects from a class. An object is also called an instance of a class and the process of creating this object is called **instantiation**.

**Defining a Class in Python**

Like function definitions begin with the [def](https://www.programiz.com/python-programming/keyword-list#def) keyword in Python, class definitions begin with a [class](https://www.programiz.com/python-programming/keyword-list#class) keyword.

The first string inside the class is called docstring and has a brief description about the class. Although not mandatory, this is highly recommended.

Here is a simple class definition.

class MyNewClass:

'''This is a docstring. I have created a new class''' pass

A class creates a new local [namespace](https://www.programiz.com/python-programming/namespace) where all its attributes are defined. Attributes

|  |  |  |  |
| --- | --- | --- | --- |
| may be data or functions.   |  | | --- | | \_\_ |  |  | | --- | | \_\_doc\_\_ |   There are also special attributes in it that begins with double underscores . For example, gives us the docstring of that class.  As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class. | |
| class Person:  "This is a person class" age = 10    def greet(self): print('Hello')      # Output: 10  print(Person.age)    # Output: <function Person.greet> print(Person.greet) | |

# Output: 'This is my second class' print(Person.\_\_doc\_\_)

**Output**

10

<function Person.greet at 0x7fc78c6e8160>

This is a person class

**Creating an Object in Python**

We saw that the class object could be used to access different attributes.

It can also be used to create new object instances (instantiation) of that class. The procedure to create an object is similar to a [function](https://www.programiz.com/python-programming/function) call.

>>> harry = Person()

|  |  |  |
| --- | --- | --- |
| This will create a new object instance named | harry | . We can access the attributes of |

objects using the object name prefix.

Attributes may be data or method. Methods of an object are corresponding functions of that class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | Person.greet |   This means to say, since is a function object (attribute of | | | |
| class), | | Person.greet | will be a method object. |
| class Person:  "This is a person class" age = 10    def greet(self): print('Hello')      # create a new object of Person class harry = Person()    # Output: <function Person.greet> print(Person.greet)    # Output: <bound method Person.greet of <\_\_main\_\_.Person object>> print(harry.greet)    # Calling object's greet() method  # Output: Hello harry.greet() | | | |

**Output**

<function Person.greet at 0x7fd288e4e160>

<bound method Person.greet of <\_\_main\_\_.Person object at 0x7fd288e9fa30>> Hello

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| You may have noticed the | | self | | parameter in function definition inside the class but we | | | | |
| called the method simply as | | | |  | | --- | | harry.greet() | | | | without any [arguments.](https://www.programiz.com/python-programming/function-argument) It still worked.  This is because, whenever an object calls its method, the object itself is passed as the | | |
|  | | |
| first argument. So, | harry.greet() | | | | translates into | | Person.greet(harry) | . |
| In general, calling a method with a list of n arguments is equivalent to calling the corresponding function with an argument list that is created by inserting the method's object before the first argument.   |  | | --- | | self |   For these reasons, the first argument of the function in class must be the object itself. This is conventionally called . It can be named otherwise but highly recommend to follow the convention. | | | | | | | | |

Another Example:

# We use the "class" statement to create a class

**class** **Human**:

# A class attribute. It is shared by all instances of this class species **=** "H. sapiens"

# Basic initializer, this is called when this class is instantiated.

# Note that the double leading and trailing underscores denote objects

# or attributes that are used by Python but that live in user-controlled

# namespaces. Methods(or objects or attributes) like: \_\_init\_\_, \_\_str\_\_,

# \_\_repr\_\_ etc. are called special methods (or sometimes called dunder methods) # You should not invent such names on your own.

**def** \_\_init\_\_(self, name):

# Assign the argument to the instance's name attribute self**.**name **=** name

# Initialize property self**.**\_age **=** 0

# An instance method. All methods take "self" as the first argument **def** **say**(self, msg):

**print**("{name}: {message}"**.**format(name**=**self**.**name, message**=**msg))

# Another instance method **def** **sing**(self):

**return** 'yo... yo... microphone check... one two... one two...'

# A class method is shared among all instances

# They are called with the calling class as the first argument

**@classmethod**

**def** **get\_species**(cls): **return** cls**.**species

# A static method is called without a class or instance reference

**@staticmethod**

**def** **grunt**():

**return** "\*grunt\*"

# A property is just like a getter.

# It turns the method age() into an read-only attribute of the same name.

# There's no need to write trivial getters and setters in Python, though.

**@property**

**def** **age**(self):

**return** self**.**\_age

# This allows the property to be set

**@age.setter** **def** **age**(self, age): self**.**\_age **=** age

# This allows the property to be deleted

**@age.deleter**

**def** **age**(self): **del** self**.**\_age

# When a Python interpreter reads a source file it executes all its code. # This \_\_name\_\_ check makes sure this code block is only executed when this # module is the main program. **if** \_\_name\_\_ **==** '\_\_main\_\_':

# Instantiate a class

1. **=** Human(name**=**"Ian")

i**.**say("hi") # "Ian: hi"

1. **=** Human("Joel")

j**.**say("hello") # "Joel: hello"

# i and j are instances of type Human, or in other words: they are Human objects

# Call our class method

i**.**say(i**.**get\_species()) # "Ian: H. sapiens"

# Change the shared attribute

Human**.**species **=** "H. neanderthalensis"

i**.**say(i**.**get\_species()) # => "Ian: H. neanderthalensis"

j**.**say(j**.**get\_species()) # => "Joel: H. neanderthalensis"

# Call the static method **print**(Human**.**grunt()) # => "\*grunt\*"

# Cannot call static method with instance of object

# because i.grunt() will automatically put "self" (the object i) as an argument

**print**(i**.**grunt()) # => TypeError: grunt() takes 0 positional arguments but 1 was given

# Update the property for this instance

i**.**age **=** 42

# Get the property

i**.**say(i**.**age) # => "Ian: 42"

j**.**say(j**.**age) # => "Joel: 0" # Delete the property **del** i**.**age

# i.age # => this would raise an AttributeError

Example: Write a Python program to convert an integer to a roman numeral.

class py\_solution: def int\_to\_Roman(self, num):

val = [

1000, 900, 500, 400,

100, 90, 50, 40,

10, 9, 5, 4,

1

]

syb = [

"M", "CM", "D", "CD",

"C", "XC", "L", "XL",

"X", "IX", "V", "IV",

"I"

]

roman\_num = ''

i = 0 while num > 0: for \_ in range(num // val[i]):

roman\_num += syb[i]

num -= val[i] i += 1

return roman\_num

print(py\_solution().int\_to\_Roman(1))

print(py\_solution().int\_to\_Roman(4000))

Exercises:

### Write a Python class to get all possible unique subsets from a set of distinct integers Input:[4,5,6] Output : [[], [6], [5], [5, 6], [4], [4, 6], [4, 5], [4, 5, 6]]

class Subsets:

def \_\_init\_\_(self, nums):

self.nums = nums

def get\_subsets(self):

res = []

self.\_backtrack(0, [], res)

return res

def \_backtrack(self, index, path, res):

res.append(path)

for i in range(index, len(self.nums)):

self.\_backtrack(i + 1, path + [self.nums[i]], res)

s = Subsets([4, 5, 6])

print(s.get\_subsets())

A screenshot of a computer program

Description automatically generated

### Write a Python class to find a pair of elements (indices of the two numbers) from a given array whose sum equals a specific target number. Input: numbers= [10,20,10,40,50,60,70], target=50 Output: 3, 4

class PairWithSum:

def \_\_init\_\_(self, nums, target):

self.nums = nums

self.target = target

def find\_pair(self):

num\_to\_index = {}

for i, num in enumerate(self.nums):

complement = self.target - num

if complement in num\_to\_index:

return num\_to\_index[complement], i

num\_to\_index[num] = i

return None

p = PairWithSum([10, 20, 10, 40, 50, 60, 70], 50)

print(p.find\_pair())

A screenshot of a computer code

Description automatically generated

### Write a Python class to implement pow(x, n).

class Power:

def \_\_init\_\_(self, x, n):

self.x = x

self.n = n

def calculate\_power(self):

return self.\_pow(self.x, self.n)

def \_pow(self, x, n):

if n == 0:

return 1

half = self.\_pow(x, n // 2)

if n % 2 == 0:

return half \* half

else:

return half \* half \* x

# Example usage:

p = Power(2, 3)

print(p.calculate\_power())

A screenshot of a computer program

Description automatically generated

### Write a Python class which has two methods get\_String and print\_String. get\_String accept a string from the user and print\_String print the string in upper case.

class StringManipulation:

def \_\_init\_\_(self):

self.string = ""

def get\_string(self):

self.string = input("Enter a string: ")

def print\_string(self):

print(self.string.upper())

s = StringManipulation()

s.get\_string()

s.print\_string()

A screenshot of a computer program

Description automatically generated

### Additional Exercises:

### Write a Python class to find validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. These brackets must be close in the correct order, for example "()" and "()[]{}" are valid but "[)", "({[)]" and "{{{" are invalid.

class Parentheses:

def is\_valid(self, s):

stack = []

mapping = {')': '(', '}': '{', ']': '['}

for char in s:

if char in mapping:

top\_element = stack.pop() if stack else '#'

if mapping[char] != top\_element:

return False

else:

stack.append(char)

return not stack

p = Parentheses()

print(p.is\_valid("()"))

print(p.is\_valid("()[]{}"))

print(p.is\_valid("[)"))

print(p.is\_valid("({[)]}"))

print(p.is\_valid("{{{"))

A screenshot of a computer program

Description automatically generated

### Write a Python class to reverse a string word by word.

class ReverseWords:

def reverse(self, s):

return ' '.join(s.split()[::-1])

rw = ReverseWords()

print(rw.reverse("Hello world"))

print(rw.reverse("Python is fun"))

A screen shot of a computer code

Description automatically generated

### Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

class Circle:

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return 3.14159 \* self.radius \*\* 2

def perimeter(self):

return 2 \* 3.14159 \* self.radius

# Example usage:

c = Circle(5)

print("Area:", c.area())

print("Perimeter:", c.perimeter())

A screenshot of a computer

Description automatically generated

of **100**