



NYP LIT

Python Games Workshop Day 1

By Planning Committee

26th Dec 2022



all copyrights reserved for LIT CLUB





Pre Workshop Preparations

Make sure you have these prepared before the workshop:

1. Python and PyCharm/Vscode
2. Flask installed in Python environment
 - a. Install flask using 'pip install flask' in your terminal

Note! You may refer to the PDF if you are unsure how to install these or reach out to our LIT members for help via Discord



Agenda

Recap on basic HTML, CSS
JavaScript and Python

Introduction to Object-Oriented
Programming (OOP)

Introduction to Flask and Jinja

Coding hands-on



Recap on basic HTML, CSS and JavaScript

[Back to Agenda Page](#)





HTML basics

<html> creates a HTML document

<head> includes title and other info you
want hidden

<body> includes all information you want
displayed

<title> page title seen on the tab and
when bookmarked

```
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
</head>
<body>

  <h1>This is a Heading</h1>
  <p>This is a paragraph.</p>

</body>
</html>
```



HTML texts and formatting

<h1> to **<h6>** creates headers that range from being the largest to the smallest respectively

This is heading 1

This is heading 2

This is heading 3

<p> creates a new paragraph for a block of text

`<p>Creates a new paragraph</p>`

**** emphasized text displayed in italic

You *have* to hurry up!

HTML texts and formatting

emphasized text displayed in bold

This text is important!

**
**

to create space or break text into a new line

```
<p2>This is my list. <br>
The tag makes a break <br>
Very helpful. <br><p2>
```

<hr>

to insert a horizontal line

HTML

HTML is a Hypertext Markup Language.

HR Tag

HR tag is used to draw a horizontal line within the texts to separate content.



HTML classes and id

<div> is a division tag, which are containers that divide a page into sections

<div id = 'example1'> apply the id, 'example1', style onto the elements in this specific <div> tag

<div class = 'example2'> apply the class, 'example2', style onto the elements in this specific <div> tag

**** acts as a container to modify short pieces of text



HTML links

`` `<a>` is used to retrieve hyperlinks

'href' contains the path/url to a different page

'target' specifies how the linked document is opened

`` `` is used to display images

'src' contains the path/url to an image

'alt' displays the contained text as an alternative if the image fails to load



HTML other media

```
<video src="video.mp4" width="300" height="240" controls></video>
```

‘src’ contains the path/url to the video

‘width’ and ‘height’ is used to control the video display size in pixels

‘controls’ controls provide video controls for the video



HTML lists

`` creates an unordered list

`` creates an ordered list

`` contains each list item

Top 5 Most populated countries in the World

1. China
2. India
3. USA
4. Indonesia
5. Pakistan

Fruits and Veggies

1. Fruits
 - Apple
 - Banana
 - Oranges
2. Vegetables
 - Spinach
 - Carrot
 - Onion

Shopping List

- Bread
- Milk
- Vegetables
- Fruits

HTML tables

<table> indicates and creates a table

<th> indicates and creates table header (which is in bold)

<tr> indicates and creates a table row

<td> indicates and creates a table cell in a table row

<td colspan = '2'> table cells will span across 2 columns

<td rowspan = '2'> table cells will span across 2 rows

```
7 <table border="1">
8   <tr>
9     <th>Firstname</th>
10    <th>Lastname</th>
11    <th>Age</th>
12  </tr>
13  <tr>
14    <td>Aryan</td>
15    <td>Gupta</td>
16    <td>23</td>
17  </tr>
18  <tr>
19    <td>John</td>
20    <td>Reece</td>
21    <td>32</td>
22  </tr>
23  <tr>
24    <td>Samntha</td>
25    <td>Groves</td>
26    <td>41</td>
27  </tr>
28 </table>
```

| Day | Temperature (°C) | | | |
|-----------|------------------|----------|----------|-----------|
| Country | Singapore | Malaysia | Thailand | Indonesia |
| Sunday | 30 | 30 | 34 | 31 |
| Monday | 31 | 34 | 35 | 30 |
| Tuesday | 31 | 34 | 34 | 31 |
| Wednesday | 31 | 33 | 33 | 31 |
| Thursday | 31 | 33 | 34 | 32 |
| Friday | 31 | 31 | 35 | 32 |
| Saturday | 31 | 32 | 36 | 32 |

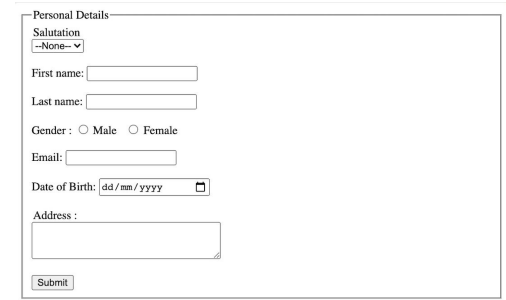
HTML forms

<form> indicates and creates a form

<input> used to define an input type

<label> indicates and creates a label for the field

<option> creates an option to select from for form fields



```
<form>
  First name: <input type = "text" name = "first_name" />
  <br>
  Last name: <input type = "text" name = "last_name" />
</form>
```



HTML forms

`<form action ="/submit.html" method = "get">`

'action' specifies the target site to send the client to after successful or failed form submission attempt

'method' specifies how the data is handled

`<input id="username" name="username" type="text">`

Example used to specify id, name, type, and value of the input field

CSS basics

'p' in this case targets a specific HTML element

```
p {  
  color: green;  
}
```

'*' in this case targets all elements of any type

```
* {  
  font-family: Verdana;  
}
```

Any element that comes after '#' represents the id attribute

```
<h1 id='large-title'> ... </h1>
```

```
#large-title {  
  
}
```

CSS basics

Any element that comes after '.' represents a class attribute

```
<p class='brand'>Sole Shoe Company</p>
```

```
.brand {  
}
```

Control 'height' and 'weight' values of an element in pixels

```
p {  
  height: 80px;  
  width: 240px;  
}
```

Border controls the border around an element

CSS basics

Margin controls the space outside the box

```
p {  
  margin: 6px 10px 5px 12px;  
}
```

```
p {  
  border: 1px solid aquamarine;  
  margin: 20px;  
}
```

Padding controls the space between the content and border of a box

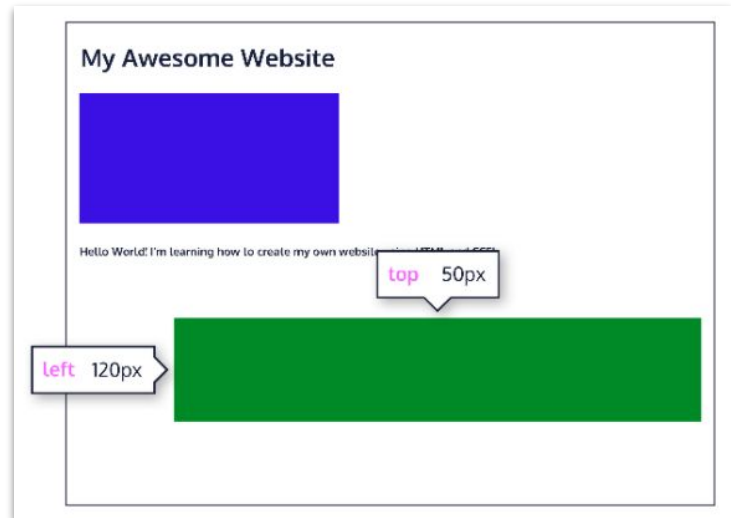
```
p.content-header {  
  border: 3px solid coral;  
  padding: 10px;  
}
```

```
p.content-header {  
  padding: 6px 11px 4px 9px;  
}
```

CSS position

‘Relative’ positions an element relative to its default static position

```
.green-box {  
  background-color: green;  
  position: relative;  
  top: 50px;  
  left: 120px;  
}
```



CSS position

‘absolute’ positions an element so that all other elements ignore it



CSS position

'fixed' positions an element so that it will remain when scrolling

```
.title {  
  position: fixed;  
  top: 0px;  
  left: 0px;  
}
```



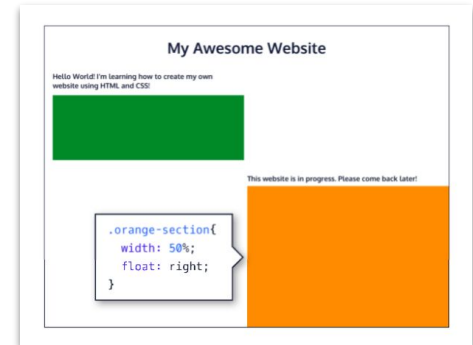
CSS display

‘Inline’ displays elements in the same line while ‘Block’ displays the elements in a block

```
h1 {  
  display: inline;  
}
```

‘Float’ makes an element stick to one side of the site

```
.green-section {  
  width: 50%;  
  height: 150px;  
}  
  
.orange-section {  
  background-color: orange;  
  width: 50%;  
  float: right;  
}
```



‘Text-align’ specifies to which side is the element centered to

```
h1 {  
  text-align: right;  
}
```



CSS colours

‘Colour’ styles the element’s foreground colour

‘Background-color’ styles the element’s background colour

```
h1 {  
  color: red;  
  background-color: blue;  
}
```

CSS colours

You may refer to <https://htmlcolorcodes.com/color-names/> for HTML-supported color names, or you can use RGB or Hex codes

17 Standard Color Keywords

| | | | | |
|-------------------|--------------------|-------------------|-------------------|------------------|
| maroon #800000 | red #ff0000 | orange #ffa500 | yellow #ffff00 | olive #808000 |
| purple #800080 | fuchsia #ff00ff | white #ffffff | lime #00ff00 | green #008000 |
| navy #000080 | blue #0000ff | aqua #00ffff | teal #008080 | |
| black #000000 | silver #c0c0c0 | gray #808080 | | |

```
h1 {  
  color: rgb(23, 45, 23);  
}
```

```
color: rgba(234, 45, 98, 0.33);
```

```
darkseagreen: #8FBC8F  
sienna:       #A0522D  
saddlebrown:  #8B4513  
brown:        #A52A2A  
black:         #000000 or #000  
white:         #FFFFFF or #FFF  
aqua:          #00FFFF or #0FF
```



CSS fonts

‘Font-family’ is used to specify the font you want to use for texts

```
h1 {  
  font-family: 'Times New Roman';  
}
```

‘Font-size’ controls the size of the font

```
p {  
  font-size: 18px;  
}
```




JavaScript

`console.log()` – to write and print codes into the console

`document.write()` – to write codes on a HTML File

```
const unknown1 = 'foo';  
console.log(typeof unknown1); // Output:  
string  
  
const unknown2 = 10;  
console.log(typeof unknown2); // Output:  
number  
  
const unknown3 = true;  
console.log(typeof unknown3); // Output:  
boolean
```



JavaScript

Increment and Decrement

1. ++ - Increment
2. -- - Decrement

String Concatenation

1. + - Concatenation - Add strings and variables together

Property

1. . - retrieve property info by appending a period to a string

Arithmetic

1. + - Add
2. - - Subtract
3. * - Multiply
4. / - Divide
5. = - Assignment

Assignment

1. += - Add and Equal - a += 1, short for a = a + 1
2. -= - Sub and Equal - a -= 1, short for a = a - 1



JavaScript variables

There are 3 ways to create/declare a variable.

1. var – assign a variable to var using the assignment operator(=)
2. let – declare a variable without assigning a value, signal that the variable can be reassigned to a different value
3. const – used in cases where the value shouldn't be changed

```
var fruit = 'apple';  
document.write(fruit);  
document.write('<br/>');  
  
let color = 'yellow';  
document.write(color + '<br/>');  
color = 'red';  
document.write(color + '<br/>');  
  
const subject = 'Math';  
document.write(subject + '<br/>')
```

JavaScript conditionals

if – execute if the condition is true for if statement

else if – execute if else if statement is true

else – none of the statements above are true

```
let stopLight = 'yellow';

if (stopLight === 'red') {
  console.log('Stop!');
} else if (stopLight === 'yellow') {
  console.log('Slow down.');
```

```
} else if (stopLight === 'green') {
  console.log('Go!');
} else {
  console.log('Caution, unknown!');
}
```

```
let sale = true;

sale = false;

if(sale) {
  console.log('Time to buy!');
}
else{
  console.log('Time to wait for a sale.');
```

```
}
```



JavaScript operators

Comparison

1. < – less than
2. > – greater than
3. <= – less than or equal to
4. >= – greater than or equal to
5. === – is equal to
6. !== – is not equal to

Logical

1. && – and
2. || – or
3. ! – not

JavaScript functions

A function is created so that we can call the same function as many times as we want to perform a specific task

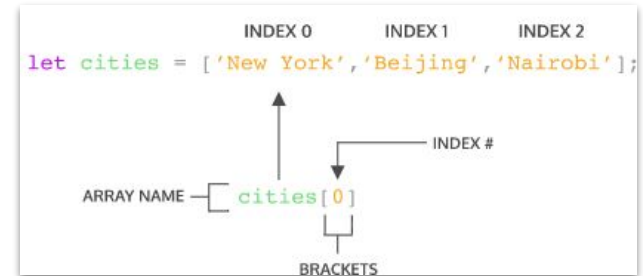
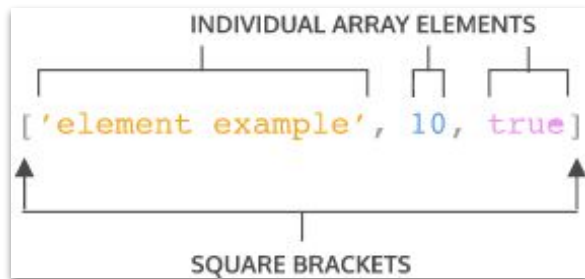
```
function calculateArea(width, height) {  
  const area = width * height;  
  return area;  
}
```

RETURN KEYWORD RETURN VALUE

```
① function getGreeting() {  
  ③   console.log("Hello, World!");  
  }  
  ② getGreeting();  
  ④ // Code after function call
```

JavaScript arrays

An array is created by wrapping items in square brackets []. It can store elements of any data type



```
let seasons = ['Winter', 'Spring', 'Summer', 'Fall'];

seasons[3] = 'Autumn';
console.log(seasons);
//Output: ['Winter', 'Spring', 'Summer', 'Autumn']
```



JavaScript arrays

Methods

- .push() - add items to the end of an array
- .pop() - remove the last item in an array
- .shift() - remove the first item in an array
- .unshift() - add items to the front of an array
- .slice() - extracts a section of array and returns a new array
- .indexOf - return the index of an element

Property

- .length - number of items in an array

JavaScript loops

While loops achieve the same outcome as a for loops but is used when we don't know how many times the loop should run

For loop loops a code until desired

```
for (let counter = 0; counter < 4;
counter++){
  console.log(counter);
}
```

```
const animals = ['Grizzly Bear', 'Sloth',
'Sea Lion'];
for (let i = 0; i < animals.length; i++){
  console.log(animals[i]);
}
```

```
// A for loop that prints 1, 2, and 3
for (let counterOne = 1; counterOne < 4;
counterOne++){
  console.log(counterOne);
}

// A while loop that prints 1, 2, and 3
let counterTwo = 1;
while (counterTwo < 4) {
  console.log(counterTwo);
  counterTwo++;
}
```

JavaScript objects

Create objects with key-value pairs in the code block

Access the value of the key-value pair using dot notation or bracket notation

```
let spaceship = {  
  homePlanet: 'Earth',  
  color: 'silver'  
};  
spaceship.homePlanet; // Returns 'Earth',  
spaceship.color; // Returns 'silver',
```

```
let spaceship = {  
  'Fuel Type': 'Turbo Fuel',  
  'Active Duty': true,  
  homePlanet: 'Earth',  
  numCrew: 5  
};  
spaceship['Active Duty']; // Returns  
true  
spaceship['Fuel Type']; //  
Returns 'Turbo Fuel'  
spaceship['numCrew']; // Returns 5  
spaceship['!!!!!!!!!!!!!!!']; // Returns  
undefined
```

```
// An object literal with two key-value  
pairs  
let spaceship = {  
  'Fuel Type': 'diesel',  
  color: 'silver'  
};
```



Coding Hands On Section 1

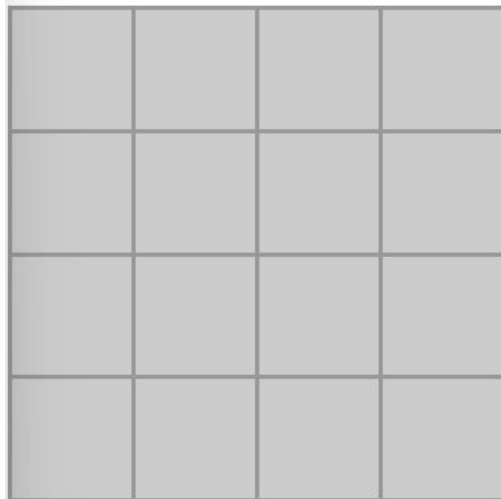
[Back to Agenda Page](#)





HTML, CSS, JS

1. Create and display a 5x5 square board on your website using HTML tables and style it using CSS. Example:



HTML, CSS, JS

2. With reference to the piece of code below, using a function, create a piece of text that, when clicked on, will change into a different piece of text

```
<!DOCTYPE html>
<html>
<body>

<button id="demo" onclick="myFunction()">Click me to change my HTML content (innerHTML).</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed!";
}
</script>

</body>
</html>
```



Recap on Python

[Back to Agenda Page](#)





Python data types

Integers (int) are whole numbers (can be negative or positive)

Floating-point numbers (float) represents numbers with decimals

Strings (str) are used for concatenation and usually for non-numeric values and not for arithmetic operators. Strings are contained in quotations as such:

```
print("Let's print out a string!")
```

```
my_str = "Hello World"
```



Python basics

input() function is used to obtain an input from the user using a prompt

filter() function used to exclude items in an iterable object

```
name = input("Hi! What's your name? ")
print("Nice to meet you " + name + "!")
```

```
age = input("How old are you ")
print("So, you are already " + str(age) + " years old, "
      + name + "!")
```

```
ages = [5, 12, 17, 18, 24, 32]
```

```
def myFunc(x):
    if x < 18:
        return False
    else:
        return True
```

```
adults = filter(myFunc, ages)
```

```
for x in adults:
    print(x)
```


Python functions

To create a function, use the def keyword followed by the function name.

```
def name():  
    print("What's your name?")  
  
name()
```

```
def add_numbers(x, y, z):  
    a = x + y  
    b = x + z  
    c = y + z  
    print(a, b, c)  
  
add_numbers(1, 2, 3)
```



Python lists

Each value inside a list is called an item and they are placed within square brackets to indicate it is a list

You can use `.append` (add item to back of list) or `.insert` (add item to specified index in the list) methods to add new items to existing lists

```
my_list = [1, 2, 3]
my_list2 = ["a", "b", "c"]
my_list3 = ["4", d, "book", 5]
```

```
beta_list = ["apple", "banana", "orange"]
beta_list.append("grape")
print(beta_list)
```

```
beta_list = ["apple", "banana", "orange"]
beta_list.insert(2, "grape")
print(beta_list)
```



Python lists

You can use `.remove()` or `.pop()` methods to remove an item from the list

```
beta_list = ["apple", "banana", "orange"]  
beta_list.remove("apple")  
print(beta_list)
```

```
beta_list = ["apple", "banana", "orange"]  
beta_list.pop()  
print(beta_list)
```

You can also use a for loop for lists to perform certain functions

```
for x in range(1,4):  
    beta_list += ['fruit']  
    print(beta_list)
```

Python if else

Python uses if, elif, and else statements

```
x = 35
if x > 20:
    print("Above twenty,")
if x > 30:
    print("and also above 30!")
```

```
a = 45
b = 45
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```

```
if age < 4:
    ticket_price = 0
elif age < 18:
    ticket_price = 10
else: ticket_price = 15
```

As well as if not in statements to check for appearance of the variable in the list

```
new_list = [1, 2, 3, 4]
x = 10
if x not in new_list:
    print("'x' isn't on the list, so this is True!")
```



Python loops

For loop loops a code until the condition stated in the first statement is fulfilled

```
for x in "apple":  
    print(x)
```

While loop loops a set of statements as long as the condition for them is true

```
#print as long as x is less than 8  
  
i = 1  
while i < 8:  
    print(x)  
    i += 1
```

Use the statement break to stop a loop even if the condition is met

```
if i == 4:  
    break
```



Python troubleshooting

'try' is used to try out a piece of code and in the case that it doesn't work, run the codes after 'except', and 'else' is used to run codes even if there is an error

```
my_dict = {"a":1, "b":2, "c":3}

try:
    value = my_dict["a"]
except KeyError:
    print("A KeyError occurred!")
else:
    print("No error occurred!")
```



Coding Hands On

Section 2

[Back to Agenda Page](#)





Python

1. Create a script such that it firstly prompts the user to input a temperature in degrees
 - a. The script will output 'cold' for temperatures below 20 degrees
 - b. The script will output 'hot' for temperatures above 37 degrees
 - c. The script will output 'neither hot nor cold' for anything in between.
 - d. Make use of 'try' and 'except' functions if you need to.



Break

[Back to Agenda Page](#)





Quiz

[Back to Agenda Page](#)





Let's test your understanding so far!

Join at: <https://quizizz.com/join>

Game Pin: **520 1880**



Introduction to Object Oriented Programming

[Back to Agenda Page](#)





What is Object-Oriented Programming?

Object-Oriented Programming (OOP) is a method of structuring a program by bundling related properties and behaviors into individual objects

We will be cover the following concepts:

1. Classes and Objects
2. Methods
3. Encapsulation
4. Inheritance
5. Polymorphism
6. Abstraction



Classes and Objects

A **class** is a collection of instance variables and related methods that define a particular object type

```
class Dog:  
    pass
```

An **object** is an instantiation of a class

```
obj = Dog()
```



Classes and Objects

Classes are initialized using the `__init__()` method, also known as a **constructor**

The `__init__()` method can be given any number of parameters, but the first parameter will always be a variable called `self`

```
def __init__(self):  
    pass
```

Classes and Objects

Example: Define a class named Dog with attributes such as name and age

```
class Dog:

    def __init__(self, name, age):
        self.name = name
        self.age = age

obj = Dog("Cookie", 3)
```




Methods

Methods are functions defined inside the body of a class and are used to define the behaviors of an object

Methods

Example: Define a method named `breed()` in the `Dog` class

```
def breed(self, breed):  
    return f"{self.name} is a {self.age} year old {breed}"  
  
obj = Dog("Cookie", 3)  
print(obj.breed("corgi"))
```

```
Cookie is a 3 year old corgi
```



Encapsulation

Encapsulation describes the concept of bundling data and methods within a single unit

A class is an example of **encapsulation** as it binds all the instance variables and methods into a single unit

Encapsulation

Example: Implement Encapsulation using a class

```
class Employee:
    def __init__(self, name, project):
        self.name = name
        self.project = project

    def work(self):
        return f"{self.name} is working on {self.project}"
```



Encapsulation

Encapsulation allows us to restrict accessing variables and methods directly

Encapsulation also prevents accidental data modification by creating private data members and methods within a class



Encapsulation

Access modifiers limit access to the variables and methods of a class

1. Public member - accessible anywhere from outside the class

```
self.name = name
```

2. Private member - accessible within the class

```
self._project = project
```

3. Protected member - accessible within the class and its subclasses

```
self.__salary = salary
```



Inheritance

Inheritance is a way of creating a new class by using details of an existing class without modifying it

The newly formed class is a derived class, also known as a child class

The existing class is a base class, also known as a parent class



Inheritance

For a child class to inherit from its parent class, the **super()** method must be used along with the **__init__** method

The **super()** function gives access to methods and properties of a parent or sibling class

The **super()** function will return an object that represents the parent class

Inheritance

Example: Define a child class named Puppy that inherit the functions of its parent class Dog

```
class Puppy(Dog):  
    def __init__(self, name, age, sound):  
        super().__init__(name, age)  
        self.sound = sound
```

Inheritance

Example: Define a `get_sound` function in the `Puppy` class

```
def get_sound(self):  
    return (f"{self.name} goes {self.sound}")  
  
pup = Puppy("Cupcake", 1, "woof")  
print(pup.get_sound())
```

Cupcake goes woof



Polymorphism

Polymorphism is the ability of an object to take many forms

Simply put, **polymorphism** allows us to perform the same action in many different ways



Polymorphism

Example:

The built-in function `len()` calculates the length of an object depending upon its type

If an object is a string, it returns the count of characters but if an object is a list, it returns the count of items in a list

The `len()` method treats an object as per its class type



Polymorphism

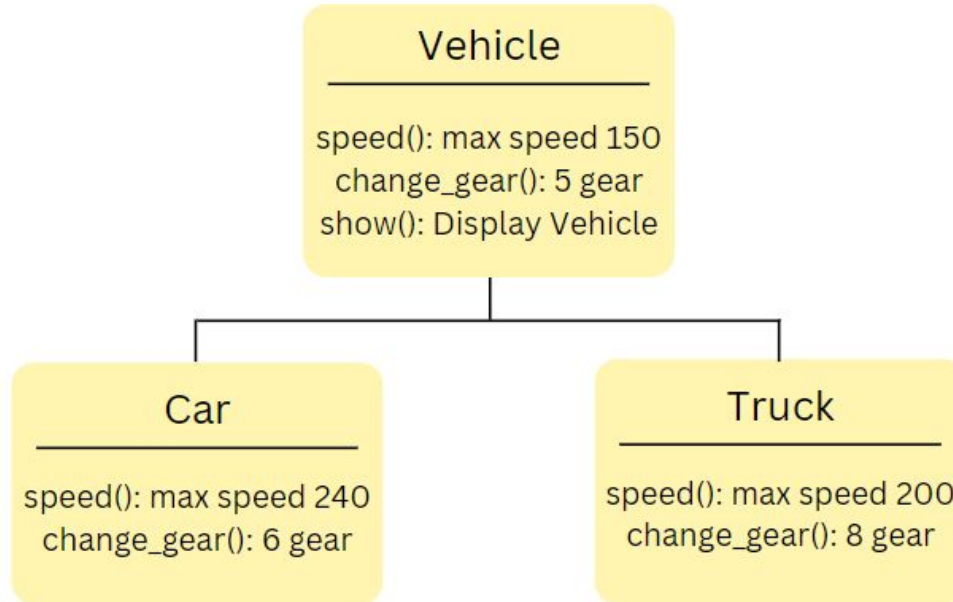
Polymorphism is mainly used with **Inheritance**

Using **Method Overriding Polymorphism**, we can define methods in the child class that have the same name as the methods in the parent class but produces a different output



Polymorphism

Example:





Abstraction

Abstraction hides the internal functionality of the function from the users

The user is familiar with “what the function does” but the user does not know “how the function does what it does”

This enables the user to implement complex logic without having to understand or think about all the hidden background/back-end complexity



Abstraction

Example:

When we use the TV remote to increase the volume, we do not know how pressing a key increases the volume of the TV

We only know that the “+” button on the TV remote will increase the volume



Coding Hands On

Section 3

[Back to Agenda Page](#)





Object-Oriented Programming

1. Write a Python program to create a Vehicle class that includes the following
 - a. name, max_speed, and mileage instance attributes
 - b. seating_capacity() method that will display the name and capacity of passengers the vehicle can contain

```
The seating capacity of a ModelX is 10 passengers
```



Object-Oriented Programming

2. Create a child class Bus that will inherit all the variables and methods of Vehicle class
 - a. Give the capacity argument of Bus.seating_capacity() a default value of 50

```
The seating capacity of a School Volvo is 50 passengers
```



Object-Oriented Programming

3. Write a Python program to create a Computer class that includes the following
 - a. A private variable maxprice with a default value of 900
 - b. sell() method that displays the selling price
 - c. A setter method setMaxPrice()

Attempt to change the value of the private variable maxprice using
c. `__maxprice = 1000`. Are you able to modify the value?

Attempt to change the value of the private variable maxprice using the
setter method `setMaxPrice()`. Are you able to modify the value?



Introduction to Flask and Jinja

[Back to Agenda Page](#)





Flask and Jinja

Flask is a web development framework with a built-in development server and a debugger

Templates are used to expand the functionality of a web application

Jinja is a fast, expressive, extensible template engine

Data is able to be shared and processed before being turned into content and sent back to the client



Coding Hands On

Section 4

[Back to Agenda Page](#)





Flask

1. Open a new project and virtual environment in Python

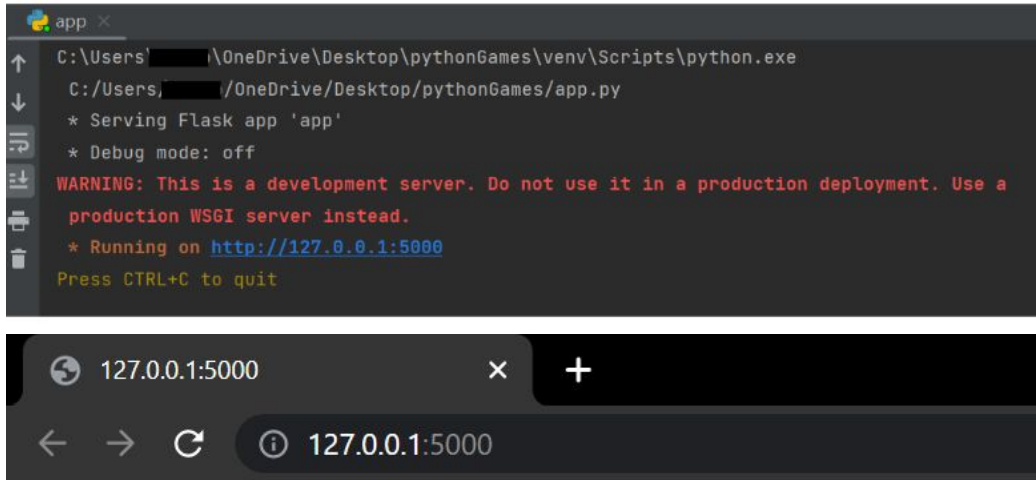
Use pip to install the Flask library

Create a new file called app.py and complete the following steps

- a. Import the Flask library
- b. Create the Flask instance
- c. Create an app route("/") with a hello() function that returns "Hello World!"
- d. Run the app using `__name__ == "__main__"`

Flask

It should look like this



```
app x
C:\Users\██████\OneDrive\Desktop\pythonGames\venv\Scripts\python.exe
C:/Users/██████/OneDrive/Desktop/pythonGames/app.py
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

127.0.0.1:5000

127.0.0.1:5000

Hello World!

Flask

Create a new directory called templates with the files base.html and home.html and copy the following code into base.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.
css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOh
cWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <title>{% block title %} {% endblock %}</title>
</head>
<body>
```

```
<nav class="navbar navbar-expand-md navbar-light bg-light">
  <a class="navbar-brand" href="{{ url_for('home') }}">FlaskBlog</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="{{ url_for('about') }}">About</a>
      </li>
    </ul>
  </div>
</nav>
<div class="container">
  {% block content %} {% endblock %}
</div>
</body>
</html>
```



Flask

Copy the following code into home.html

```
{% extends 'base.html' %}  
  
{% block content %}  
<h1>{% block title %} Welcome to FlaskBlog {% endblock %}</h1>  
{% endblock %}
```



Flask

2. Create a new file in the templates directory called about.html

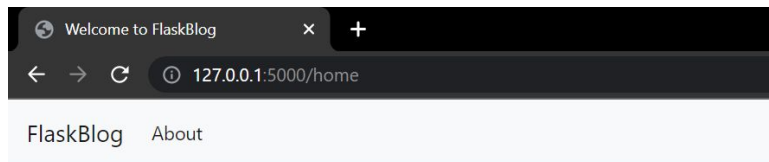
Modify the code used for index.html to display the title “Learn More About FlaskBlog” in about.html

In app.py, complete the following steps

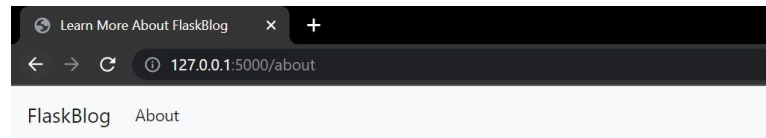
- a. Import `render_template` from flask
- b. Create an app route for index.html and about.html
- c. Run app.py to ensure the web application works

Flask

You should see this



Welcome to FlaskBlog



Learn More About FlaskBlog



Please help us fill

In this feedback form here:

<https://forms.gle/2pqYz3XbormXvrNM7>

