



PYTHON TRAINING

Prepared By: Iotasol



Enclosed

Contents

Introduction	4
Day 1:	5
Day 2:	6
Day 3:	7
Assignment – I	7
Assignment – II	9
Day 6:	10
Day 7:	15
Day 8:	18
Day 9: Database Connectivity	19
Day 10-12:	20
Day 13-15:	22
Day 16-25: Project	23

Document Change History

Version	Change	Created / modified by Date Modified	Reviewed by Date Reviewed	Approved by Date Approved
1.0	Draft created	Bhavkeerat Singh 21-Apr-23	Core SME Group	

Introduction

This document is intended to provide you with a solid foundation in Python and Flask web framework programming. By going through the exercises outlined in this document, you will learn fundamental concepts and coding techniques that form the building blocks of programming. Once you master these basic concepts, you will be able to tackle more complex concepts and programming languages with greater ease and at a faster pace. The assignments are structured in a step-by-step manner, with each exercise building on the concepts learned in the previous one. It is essential to complete each task without skipping any questions and without seeking assistance from your peers or the internet.

Welcome to exciting world of Programming

GET SET READY AND GO

ALL THE VERY BEST!!!!

Day 1:

Topics:

1. Setup python in your system after downloading it from the official website.
<https://www.python.org/downloads/>
2. Start with python syntax.
3. Learn about -
 - Variables
 - Datatypes
 - Numbers
 - Casting
 - Strings
 - Booleans
 - Taking User input
 - Operators
 - String formatting
 - Slicing strings
 - PIP

Important notes

- Learn about PIP properly as it will help you to install modules.
- Learn about Booleans properly and read about cases when a condition returns True or False

Day 2:

Topics:

1. What are conditionals? (If – Else, If – Elif)
2. Range method
3. Loops – While and for loop
4. Learn about in-Built Data Structures provided in python with their in-build methods:
 - Lists
 - Dictionary
 - Tuples
 - Sets
5. Learn about some important python built-in functions.
 - a. abs()
 - b. round()
 - c. pow()
 - d. map()
 - e. filter()
 - f. zip()
 - g. reversed()

Day 3:

Topics:

1. What are functions and how to declare and call a function?
2. What are lambda functions?
3. Error Handling using try Except
4. File Handling – Learn about how to read ,write and delete a file
5. Some important python modules to learn:
 - JSON – Used for dealing with JSON data
 - Requests – Used if you want to hit a web request

Important notes

- While learning python exception handling. Do learn about how you can raise custom exceptions
- Make sure to have a basic understanding of JSON before learning JSON module

Assignment – I

1. Create a function to find whether a given number (accept from the user) is even or odd, print out an appropriate message to the user.
2. Create a function to calculate the sum of the first n positive integers.
3. Accept three inputs from user that are name, age and gender. Print “Hello <name>, Your gender is <gender> and you age is <age>” using string formatting and f-string
4. Create a function to print below list using for loops

```
[1, 2, 'apple', 'sam', 5.6]
```

5. Create a function to print below dictionary (only values not keys) using for loops.

```
{'a': 1, 'b': 'apple'}
```

6. You are given a list of lists:

```
input_list = [  
    ["cat1", "a"],  
    ["cat1", "b"],  
    ["cat2", "c"],  
    ["cat2", "d"],  
    ["cat3", "e"]  
]
```

```
{  
    "cat1": ["a", "b"],  
    "cat2": ["c", "d"],  
    "cat3": ["e"]  
}
```

Expected out is a dictionary where first item from these lists represent dictionary keys and second value represents value.

7. Create and enter this data in a csv file using file handling.

```
input_list = [  
    ['Thomas', 22, 'CSE', 'England'],  
    ['George', 25, 'CSE', 'America'],  
    ['Sam', 26, 'ME', 'England']  
]
```

Day 4-5:

Topics:

1. Learn about Object Oriented Programming in Python
2. What are Classes and objects and implement them.
3. What is Inheritance and implement it.
4. What is `__init__()` function
5. What is `__str__()` function
6. What is 'self' parameter?
7. Is 'self' a keyword and can we use another word in place of 'self'.
8. Pass statement
9. Why is `super()` used?
10. Learn about important python built in functions : -

https://www.w3schools.com/python/python_ref_functions.asp

Important notes

- You will see 'self' parameter very often, so revise it properly and learn about it in detail.

Assignment – II

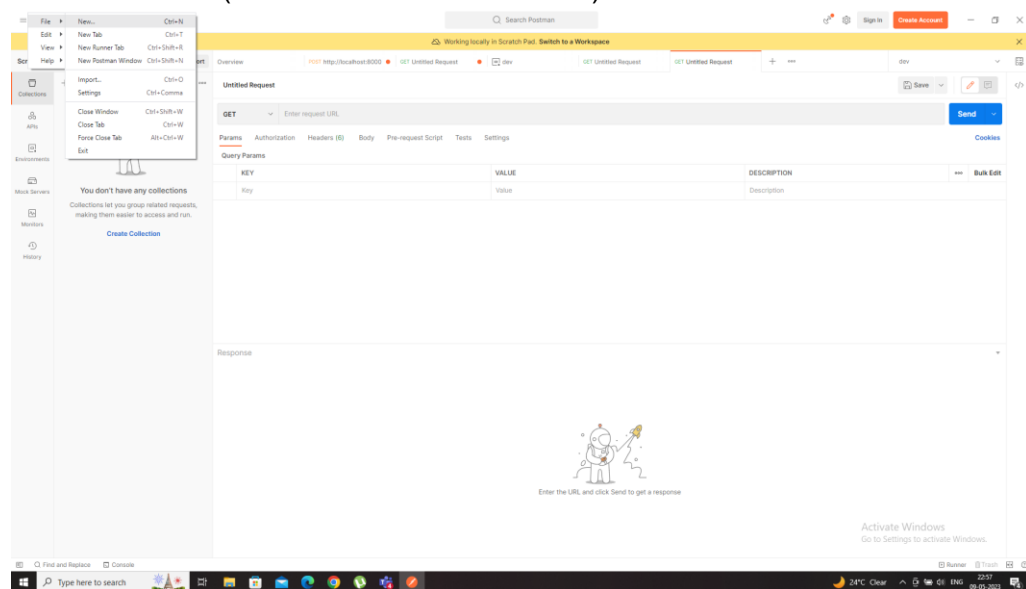
1. Create a **BankAccount** class with attributes **account_number**, **balance**, and **interest_rate**.
2. Create methods in the **BankAccount** class to deposit and withdraw money from the account, and to calculate the interest based on the interest rate and the account balance. $\text{Interest} = \text{balance} * \text{interest_rate}$
3. Create an object of the **BankAccount** class, and call the appropriate methods to deposit money, withdraw money, and calculate the interest.
4. Create a **SavingsAccount** class that inherits from the **BankAccount** class and add a new attribute **minimum_balance**.
5. Override the **withdraw** method in the **SavingsAccount** class to check if the withdrawal will bring the balance below the minimum balance, and if so, print an error message and do not allow the withdrawal.
6. Create an object of the **SavingsAccount** class, and call the appropriate methods to deposit money, withdraw money, and calculate the interest.

Day 6:

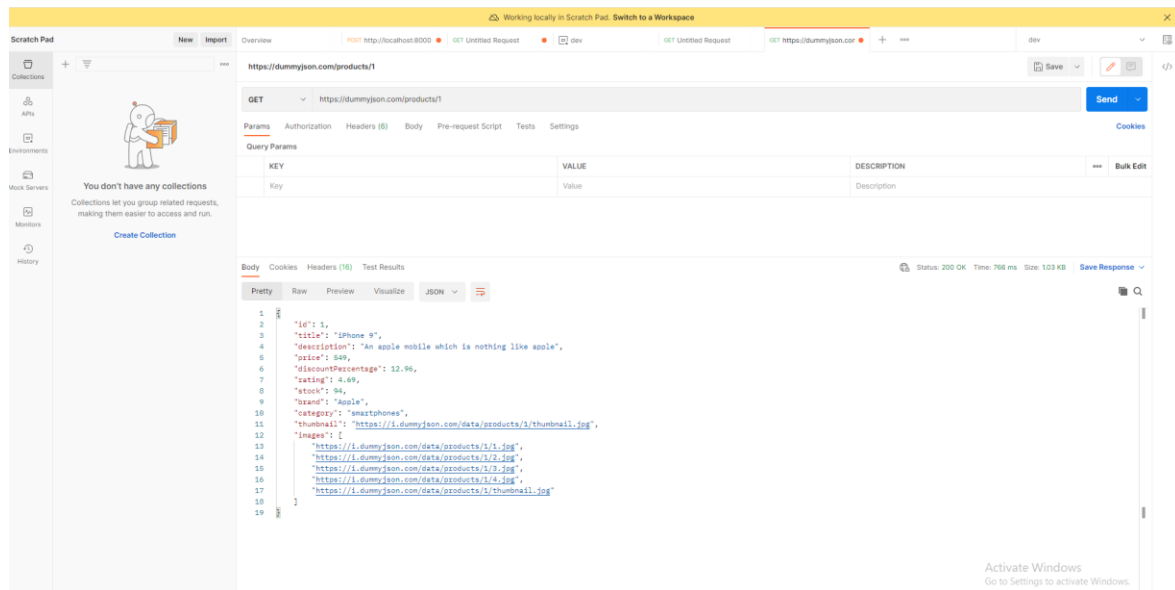
Now we are starting with FLASK framework:

Topics:

1. Introduction to Flask .
2. Understanding HTTP methods. (GET, POST, PUT, DELETE etc.)
3. Understand status codes.
4. JSON response using jsonify().
5. POSTMAN for testing API's -
 - A. Install Postman.
 - B. After Installing Postman. Follow below steps to test an API.
 - Select the three line menu from top-left corner.
 - Go to File->New.(Zoom the document to view)



- Enter the api to want to test. Select the type of request(GET, POST, PUT) etc. From the dropdown and hit run. For example : testing the url:
<https://dummyjson.com/products/1>

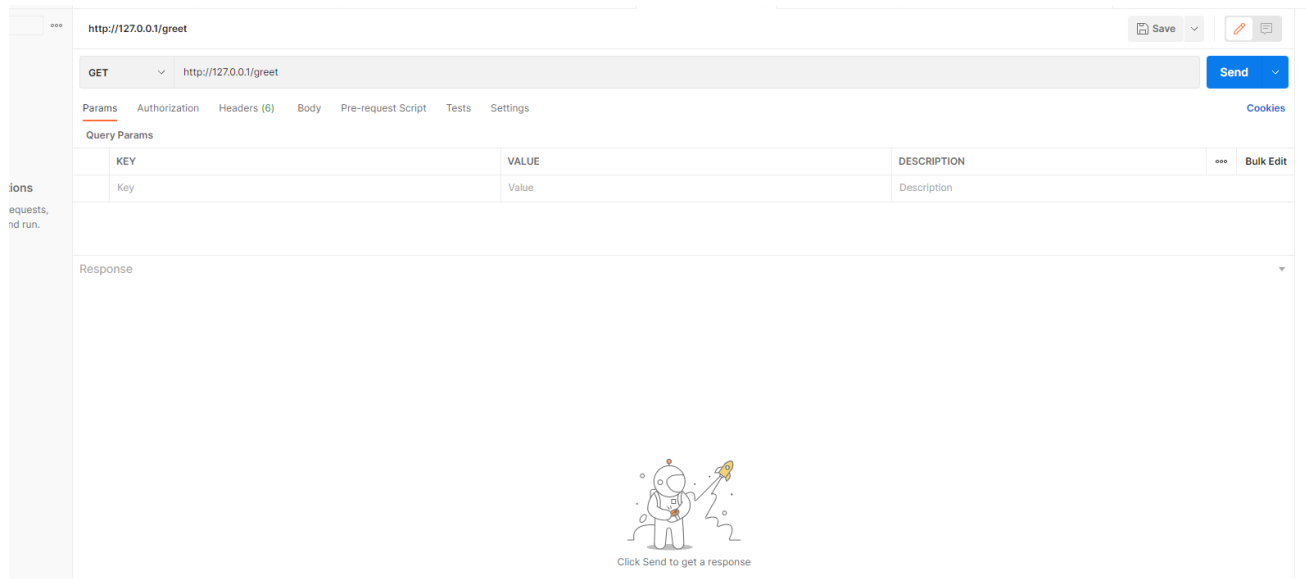


- The API's you will be creating could be tested in postman.

6. Flask routing.

TASK - Create a simple Flask app that returns a JSON response for a greeting message.

A. Greet - GET – This method will greet user. You can test this api and upcoming API's using postman. For example:



Expected Output:

```
json Copy code
{
  "message": "Hello, World!"
}
```

B. Show Products– GET – This method will return all available products from a static dictionary as a JSON response

Expected Output:

```
{
  "product_id": 1,
  "product_name": "Smartphone",
  "product_desc": "6.5-inch screen, 5G",
  "product_cost": 499.99,
  "product_category": "Electronics",
  "quantity_available": 100
},
{
  "product_id": 2,
  "product_name": "Sneakers",
  "product_desc": "Running shoes, size 10",
  "product_cost": 89.99,
  "product_category": "Footwear",
  "quantity_available": 50
},
{
  "product_id": 3,
  "product_name": "T-shirt",
  "product_desc": "Cotton fabric, XL",
  "product_cost": 19.99,
  "product_category": "Clothing",
  "quantity_available": 200
},
{
  "product_id": 4,
  "product_name": "Wireless Earbuds",
  "product_desc": "Bluetooth, noise-cancelling",
  "product_cost": 129.99,
  "product_category": "Electronics",
  "quantity_available": 75
},
{
  "product_id": 5,
  "product_name": "Backpack",
  "product_desc": "Water-resistant, laptop sleeve",
  "product_cost": 79.99,
  "product_category": "Accessories",
  "quantity_available": 40
}
```

C. Show Customers– GET – This method will return all the customers from a static dictionary as a JSON response

Expected Output:

```
{
  "product_id": 1,
  "product_name": "Smartphone",
  "product_desc": "6.5-inch screen, 5G",
  "product_cost": 499.99,
  "product_category": "Electronics"
},
{
  "product_id": 4,
  "product_name": "Wireless Earbuds",
  "product_desc": "Bluetooth, noise-cancelling",
  "product_cost": 129.99,
  "product_category": "Electronics"
}
```

Day 7:

Important Note: On day 6 , You learnt how to create a basic API and return JSON responses. We did not use this with front end (HTML pages). From now on we will create HTML templates for all the endpoints which will help you understand how the data moves between front-end and back-end.

Topics:

- a. Request and response objects.
- b. `url_for()`
- c. Flask Templates and jinja templating language
https://www.tutorialspoint.com/flask/flask_templates.htm
- d. Static Files.
https://www.tutorialspoint.com/flask/flask_static_files.htm
- e. `redirect()`
- f. Handling errors and exceptions

TASK -

1. Create a template (HTML page) for the **Show Products** endpoint and **Show Customers** endpoint created on Day 6 and bind the data with this html template. You can style the page according to your preference. Here is a sample html page. You can Create a similar page to show list of customers.

Product Catalog

Product Name	Product Description	Product Cost	Product Category	Product Quantity	Action
Product 1	A great product for all your needs.	\$19.99	Category 1	10	View Details
Product 2	Another great product for all your needs.	\$24.99	Category 2	5	View Details
Product 3	A third great product for all your needs.	\$14.99	Category 1	20	View Details
Product 4	Yet another great product for all your needs.	\$29.99	Category 3	8	View Details
Product 5	The final great product for all your needs.	\$9.99	Category 2	15	View Details

3. Create a method to view product details. In the previous Show details page , user will click View Details to navigate to this page. Here is a sample product details page. You can create a similar page to view customer details. User can also update product information

Product Detail

⋮

↔

⋮

Product Information

Product Name:

Flask T-shirt

Description:

A comfortable t-shirt with the flask logo

Category:

Clothing

Price:

19.99

Quantity:

5

Save Changes

Day 8:

TOPICS:

1. Modulating flask app.
2. Flask Blueprints and their use cases.
3. Creating a new blueprint and registering it with the Flask app.
4. Defining routes and endpoints within the blueprint.

TASK -

1. Divide the flask app in 2 modules (Blueprints)
2. The endpoints related to products will be in products blueprint and similarly the Customers endpoints will be in customers' blueprint.
3. Add below methods in the flask app.
 - Add an endpoint to filter products by category.
 - Add an endpoint to sort results by name, product cost and products category.
 - Add an endpoint to search a product by its name.
 - Add an endpoint to search users by name, sort by name.

Day 9: Database Connectivity

TOPICS:

1. Connect your flask app with a database (Postgres preferred.)
2. Learn about Flask SQL Alchemy.
3. Learn about creating models.
4. Learn about Flask-migrate.
5. Learn what migrations are and how to create and apply migrations using flask migrate.
6. What is ORM?
7. Learn how to add, delete, fetch and update data using sql alchemy

TASK

1. Add a model to the Flask app for storing customer information.

```
{
  "customer_id": 1,
  "customer_name": "John Smith",
  "email": "john.smith@example.com",
  "phone": "555-1234"
}
```

2. Modify the previous **Show Customers** method to fetch customers data from database instead of a dictionary.

Day 10-12:

TASK-

- Now the task is to use models for all the objects that are Products and Customers.
- Create Models for all the above objects in this format.

Customers:			
Column Name	Data Type	Nullable	Primary Key
customer_id	integer	not null	Yes, autoincrement
customer_name	varchar(255)	not null	
email	varchar(255)	not null	
phone	varchar(20)	not null	
Products:			
Column Name	Data Type	Nullable	Primary Key
product_id	integer	not null	Yes, autoincrement
product_name	varchar(255)	not null	
price	decimal(10, 2)	not null	
description	text	not null	

- All the operations and endpoints defined earlier need to be modified according to sql alchemy
- All the data needs to be fetched from database using Sql Alchemy.
- For all entities there needs to be CRUD(Create, Read, Update, Delete) operations.
- Also create different blueprints(modules) for all objects (Products, Customers)
- So, these are the functionalities that will be available in the flask app.

Products -

- Show All Products with pagination (Learn about pagination)
- Create/Add a Product.
- Delete/Remove a Product
- Update/Edit a Product
- Read/Get a Product
- Filter products by category.
- Sort products based on name and cost.

Customers -

- ix. Show All Customers with pagination.
- x. Create/Add a Customer.
- xi. Delete/Remove a Customer
- xii. Update/Edit a Customer's Information
- xiii. Read/Get a Customer

Create HTML forms for create methods.

Day 13-15:

TOPICS:

1. What is Authentication and authorization
2. Difference between Authentication and authorization.
3. Sessions and Cookies.
4. JWT Authentication <https://www.bacancytechnology.com/blog/flask-jwt-authentication>

TASK

1. Add JWT authentication to the app.
2. Only a signed in user can access this flask app. If he tries to access a page without signing in, he must be redirected to login page.
2. Add A field Role for the customers like ADMIN, STANDARD USER. A standard user can view, update or delete its own products only. Also the standard user cannot view other customers list and customer's details.
3. Limit all endpoints for signed in users only which includes operations like insert product, delete product and others.



Day 16-25: Project

Create a Flask Blog app with below endpoints

1. **POST /users** - Register a new user.
2. **POST /login** - Log in an existing user.
3. **GET /authors** - Get a list of all authors.
4. **GET /authors/:author_id** - Get details of a specific author.
5. **POST /authors** - Create a new author with name, bio and profile image.
6. **PATCH /authors/:author_id** - Update an existing author's details and/or profile image.
7. **DELETE /authors/:author_id** - Delete an existing author.
8. **GET /blog_posts** - Get a list of all blog posts.
9. **GET /blog_posts/:blog_post_id** - Get details of a specific blog post.
10. **POST /blog_posts** - Create a new blog post with title, content and image.
11. **PATCH /blog_posts/:blog_post_id** - Update an existing blog post's detail and/or image.
12. **DELETE /blog_posts/:blog_post_id** - Delete an existing blog post.
13. **GET /blog_posts/:blog_post_id/comments** - Get a list of all comments on a specific blog post.
14. **POST /blog_posts/:blog_post_id/comments** - Create a new comment on a specific blog post with content and author ID.
15. **PATCH /blog_posts/:blog_post_id/comments/:comment_id** - Update an existing comment's content.
16. **DELETE /blog_posts/:blog_post_id/comments/:comment_id** - Delete an existing comment.

User

Field Name	Data Type	Description
id	Integer	Unique ID for the user
username	String	Username of the user
password	String	Password of the user
email	String	Email address of the user
created_at	DateTime	Timestamp of when the user was created
updated_at	DateTime	Timestamp of when the user was last updated

Author

Field Name	Data Type	Description
id	Integer	Unique ID for the author
name	String	Name of the author
bio	String	Biography of the author
profile_image	String	URL or file path of the author's profile image
created_at	DateTime	Timestamp of when the author was created
updated_at	DateTime	Timestamp of when the author was last updated

BlogPost

Field Name	Data Type	Description
id	Integer	Unique ID for the blog post
title	String	Title of the blog post
content	String	Content of the blog post
image	String	URL or file path of the blog post's image
author_id	Integer	ID of the author who created the blog post
created_at	DateTime	Timestamp of when the blog post was created
updated_at	DateTime	Timestamp of when the blog post was last updated

Comment

Field Name	Data Type	Description
id	Integer	Unique ID for the comment
content	String	Content of the comment
author_id	Integer	ID of the author who created the comment
blog_post_id	Integer	ID of the blog post the comment belongs to
created_at	DateTime	Timestamp of when the comment was created
updated_at	DateTime	Timestamp of when the comment was last updated