Week0: Python Environment Setup

- Install Python
- Install VS Code
- Python Virtual Environmnet (pip,pipenv) * for mac use brew
- Git/Github

Week1: Python Language Refresher

- PEP8 Naming Conventions
- String
- List
- Tuple
- Sets
- Dict
- Functions
- Classes
- Standard Libraries
- Generators
- Context Managers

Week2: Build Web apps with Flask: Part1

- Introduction to Flask
- Creating simple flask app
- Creating flask templates
- Creatig summary model
- Adding values through REPL
- Displaying summaries in the frontend

Week3: Build Web apps with Flask: Part2

- Flask Forms
- Forms for summary Model
- Form Validations

Week4:Build Web apps with Flask: Part3(Theming)

- CSS Frontend Framworks
- Custom CSS
- Navbar and Logo

Week5:Build Web apps with Flask: Part4(User Authentication)

Flask-login, Flask-Bcrypt,Flask-Migrate

- Creating User Model
- Creating Users Form
- User routes & templates

Week6: Build Web apps with Flask

- Creating Categories Model,view and tempalte
- Integrating categories with summaries
- Filtering Summaries by category

Week7: App restructuring

Week8: Creating Tests

Week9: Creating Multiple Environment (Development,Testing,Production)

Week10: Automated Deployment With Docker, Travis-CI,Git/Github

Web Application Development with Python(Flask) Workshop

This workshop will teach you how to code with pyhton more efficiently by following the pythonic way of writing python code.

In this workshop we will build a web application using Flask. Flask is a **micro web framework** written in Python. It is classified as a microframework because it does not require particular tools

or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

PEP8 Naming Conventions

What is PEP 8?

- PEP: A Python Enhancement Proposal
- PEP 8: A set of style guidelines for Python
- Widely used
- https://www.python.org/dev/peps/pep-0008/

Why follow PEP 8?

- Readable code
- Mostly for yourself (and other developers)
- User-friendly code
- Following naming conventions is a way to document your code
- Mostly useful for users (which can be developers too) ===== regular_variables Variable names should be lowercase, where necessary separating words by underscores

```
first_name = 'Jaamac'
```

CONSTANTS

In Python, all variables can be modified therefore, real constants don't exist But to indicate that a variable should be treated as if it were a *constant*, names should be uppercase, where necessary separating words by underscores

```
PI = 3.1415 # Constant variable
SERVER_NAME = 'server'
```

function_names()

Names of functions and class methods should be lowercase, where necessary separating words by underscores

```
def add_two_numbers(num1,num2):
    return num1+num2
```

ClassNames

Class names should capitalize the first letter of each word

```
class MyServer:

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

    def __str__(self):
        return self.name
```

conflicting_names_

If a name is already taken, suffix an underscore

```
list_ = [12,34,6]
```

The most important naming conventions

regular_variables

Variable names should be **lowercase**, where necessary separating words by underscores