



# PYTHON ADVANCED MASTERCLASS

DEEP DIVE  
INTO  
LEARNING PYTHON



# WORKING WITH DATABASES – P1

- Introduction to Databases
- Database Concepts
- Keys
- Introduction to SQLite
- CRUD on SQLite
- Introducing 'Cursor'

```
response = requests.get(url)

# Print response status code (if you want)
if response.status_code != 200:
    print(f"Status: {response.status_code}")

# Print response status code (if you want)
print(f"Status: {response.status_code}")

# Use BeautifulSoup to parse the response
soup = BeautifulSoup(response.content, "html.parser")

# Find all images in the soup
images = soup.find_all("img", attrs={"alt": "image"})

# Print the number of images found
print(f"Number of images: {len(images)}")
```

# WORKING WITH DATABASES – P2

- Tools of the Trade in Big Databases
- Configuring MySQL
- Connection Error Handling
- CRUD in MySQL

```
requests.get(url)

ing response.status_code (if you g
onse.status_code != 200:
nt(f"Status: {response.status_code

nt(f"Status: {response.status_code

BeautifulSoup to parse the respons
BeautifulSoup(response.content, "ht

ing Post images in the soup
= soup.find_all("img", attrs={"alt

loading images
= 0
images:
```



# VIRTUAL ENVIRONMENTS

- What Are Virtual Environments?
- Why Do We Need a VE?
- Package Managers
- Conda vs Pip vs Poetry vs Pipenv
- Virtualenv and VirtualenvWrapper
- Creating and Configuring a VE

```
...load from the websi  
e = requests.get(url)  
  
ing response.status_code (if you g  
onse.status_code != 200:  
nt(f"Status: {response.status_code  
  
nt(f"Status: {response.status_code  
  
BeautifulSoup to parse the respon  
BeautifulSoup(response.content, "ht  
  
ing Post images in the soup  
= soup.find_all("img", attrs={"alt"  
  
loading images  
= 0  
images:
```

# COMPREHENSIONS

- Introduction
- Conditional Expressions
- Filters
- List Comprehensions
- Set and Dictionary Comprehensions
- Generator Expressions
- Nested Comprehensions

```
response = requests.get(url)

# Print the status code (if you want)
print(response.status_code)

# Print the status code (if you want)
print(f"Status: {response.status_code}")

# Use BeautifulSoup to parse the response
soup = BeautifulSoup(response.content, "html.parser")

# Find all images in the soup
images = soup.find_all("img", attrs={"alt": "image"})

# Print the number of images
print(len(images))
```



# DECORATORS

- Functions as 'First Class Objects'
- Inner Functions
- Functions Returning Functions
- Wrapper Function
- Reusable Decorators

```
response = requests.get(url)

# Print response.status_code (if you get a 404 error)
if response.status_code != 200:
    print(f"Status: {response.status_code}")

# Print response.status_code (if you get a 200 OK)
print(f"Status: {response.status_code}")

# Use BeautifulSoup to parse the response
soup = BeautifulSoup(response.content, "html.parser")

# Find all post images in the soup
img_tags = soup.find_all("img", attrs={"alt": "Post image"})

# Print the number of images found
print(f"Number of images: {len(img_tags)}")
```

# GENERATORS

- Stateless Data Types
- Yield vs Return
- Built-in Generator
- Custom Generator
- Memory Efficiency

```
response = requests.get(url)

# Print response.status_code (if you want)
if response.status_code != 200:
    print(f"Status: {response.status_code}")

# Print response.status_code (if you want)
print(f"Status: {response.status_code}")

# Use BeautifulSoup to parse the response
soup = BeautifulSoup(response.content, "html.parser")

# Find all Post images in the soup
images = soup.find_all("img", attrs={"alt": "Post image"})

# Print the number of images found
print(f"Found {len(images)} images")
```



# META CLASSES AND TYPE

- The 'TYPE' Function
- Altering Type of a Class
- Type vs \_\_new\_\_
- Meta Classes
- Custom Class Types

```
...load from the websi  
e = requests.get(url)  
  
ing response.status_code (if you g  
onse.status_code != 200:  
nt(f"Status: {response.status_code  
  
nt(f"Status: {response.status_code  
  
BeautifulSoup to parse the respon  
BeautifulSoup(response.content, "ht  
  
ing Post images in the soup  
= soup.find_all("img", attrs={"alt"  
  
loading images  
= 0  
images:
```



# DUCK TEST

- Duck Typing
- The Philosophy of Duck Typed Languages
- Duck Typing and Inheritance
- Python Viewpoint of Duck Typing
- Dive Even Deeper
- Story of a Billionaire Penguin !

```
...load from the websi  
e = requests.get(url)  
  
ing response.status_code (if you g  
onse.status_code != 200:  
nt(f"Status: {response.status_code  
  
nt(f"Status: {response.status_code  
  
BeautifulSoup to parse the respons  
BeautifulSoup(response.content, "ht  
  
ing Post images in the soup  
= soup.find_all("img", attrs={"alt"  
  
loading images  
= 0  
images:
```

# THREADING - P1

- What is a Thread?
- Starting a Thread
- Daemon Thread
- Joining Threads
- Multi-Threading
- Race Condition

```
response = requests.get(url)

# Print response.status_code (if you want)
if response.status_code != 200:
    print(f"Status: {response.status_code}")

# Print response.status_code (if you want)
print(f"Status: {response.status_code}")

# Use BeautifulSoup to parse the response
soup = BeautifulSoup(response.content, "html.parser")

# Find all post images in the soup
img_tags = soup.find_all("img", attrs={"alt": "image"})

# Print the number of images found
print(f"Number of images: {len(img_tags)}")
```



# THREADING - P2

- Synchronisation
- Deadlocks
- Thread Pools
- Semaphore
- Barrier
- Timer

```
...load from the websi  
e = requests.get(url)  
  
ing response.status_code (if you g  
onse.status_code != 200:  
nt(f"Status: {response.status_code  
  
nt(f"Status: {response.status_code  
  
BeautifulSoup to parse the respon  
BeautifulSoup(response.content, "ht  
  
ing Post images in the soup  
= soup.find_all("img", attrs={"alt  
  
Loading images  
= 0  
images:
```

# ASYNCIO – P1

- Asyncio Explained
- Where Does Asyncio Fit in?
- Why Asyncio is Ridiculously Hard?
- Awaitables
- Rules
- Coroutines
- Chained Coroutines

```
response = requests.get(url)

if response.status_code != 200:
    print(f"Status: {response.status_code}")

print(f"Status: {response.status_code}")

BeautifulSoup to parse the response
BeautifulSoup(response.content, "html")

Find Post images in the soup
images = soup.find_all("img", attrs={"alt": "Post image"})

Loading images
images = []
```



# ASYNCIO – P2

- Asyncio and Queues
- Async Generators
- Async Comprehensions
- Event Loop
- Async Requests

```
...load from the websi
e = requests.get(url)

ing response.status_code (if you g
onse.status_code != 200:
nt(f"Status: {response.status_code

nt(f"Status: {response.status_code

BeautifulSoup to parse the respons
BeautifulSoup(response.content, "ht

ing Post images in the soup
= soup.find_all("img", attrs={"alt

Loading images
= 0
images:
```

# LINKED LISTS

- Introducing Linked Lists
- Singly Linked Lists
- Doubly Linked Lists
- Circular Linked Lists
- Queues
- Stacks
- Deques

```
requests.get(url)

ing response.status_code (if you g
onse.status_code != 200:
nt(f"Status: {response.status_code

nt(f"Status: {response.status_code

BeautifulSoup to parse the respons
BeautifulSoup(response.content, "ht

ing Post images in the soup
= soup.find_all("img", attrs={"alt

Loading images
= 0
images:
```



# API – P1

- HTTP Protocol and HTTP Header
- HTTP Message Types
- Restful API
- Introducing fastAPI
- API Call
- Getting to Know JSON Format
- Working with Restful APIs
- Project: Getting Data From 'NASA' Mars Rover

```
requests.get(url)

ing response.status_code (if you g
onse.status_code != 200:
nt(f"Status: {response.status_code

nt(f"Status: {response.status_code

BeautifulSoup to parse the respons
BeautifulSoup(response.content, "ht

ing Post images in the soup
= soup.find_all("img", attrs={"alt

Loading images
= 0
images:
```

# API – P2

- Writing Your First API
- Path Parameters
- Enumerations
- Query Parameters
- Swagger UI

```
response = requests.get(url)

# Print response.status_code (if you get a 404 error)
if response.status_code != 200:
    print(f"Status: {response.status_code}")

# Print response.status_code (if you get a 200 OK)
print(f"Status: {response.status_code}")

# Use BeautifulSoup to parse the response
soup = BeautifulSoup(response.content, "html.parser")

# Find all images in the soup
images = soup.find_all("img", attrs={"alt": "image"})

# Print the number of images found
print(f"Number of images: {len(images)}")
```



# BONUS: DESIGN PATTERNS

- Singleton
- Observer
- Proxy
- ...

```
... = requests.get(url)

...ing response.status_code (if you g
onse.status_code != 200:
nt(f"Status: {response.status_code

nt(f"Status: {response.status_code

BeautifulSoup to parse the respons
BeautifulSoup(response.content, "ht

ing Post images in the soup
= soup.find_all("img", attrs={"alt

Loading images
= 0
images:
```

# BONUS: DESIGN STRATEGIES

- Monolithic
- Microservice
- Test Driven
- ...

```
response = requests.get(url)

if response.status_code != 200:
    print(f"Status: {response.status_code}")

soup = BeautifulSoup(response.content, "html.parser")

img_tags = soup.find_all("img", attrs={"alt": "image"})

print(f"Found {len(img_tags)} images")
```



# NOTES

## WEBSITES

- [www.python.org](http://www.python.org)
- [www.realpython.com](http://www.realpython.com)

## TEXTS

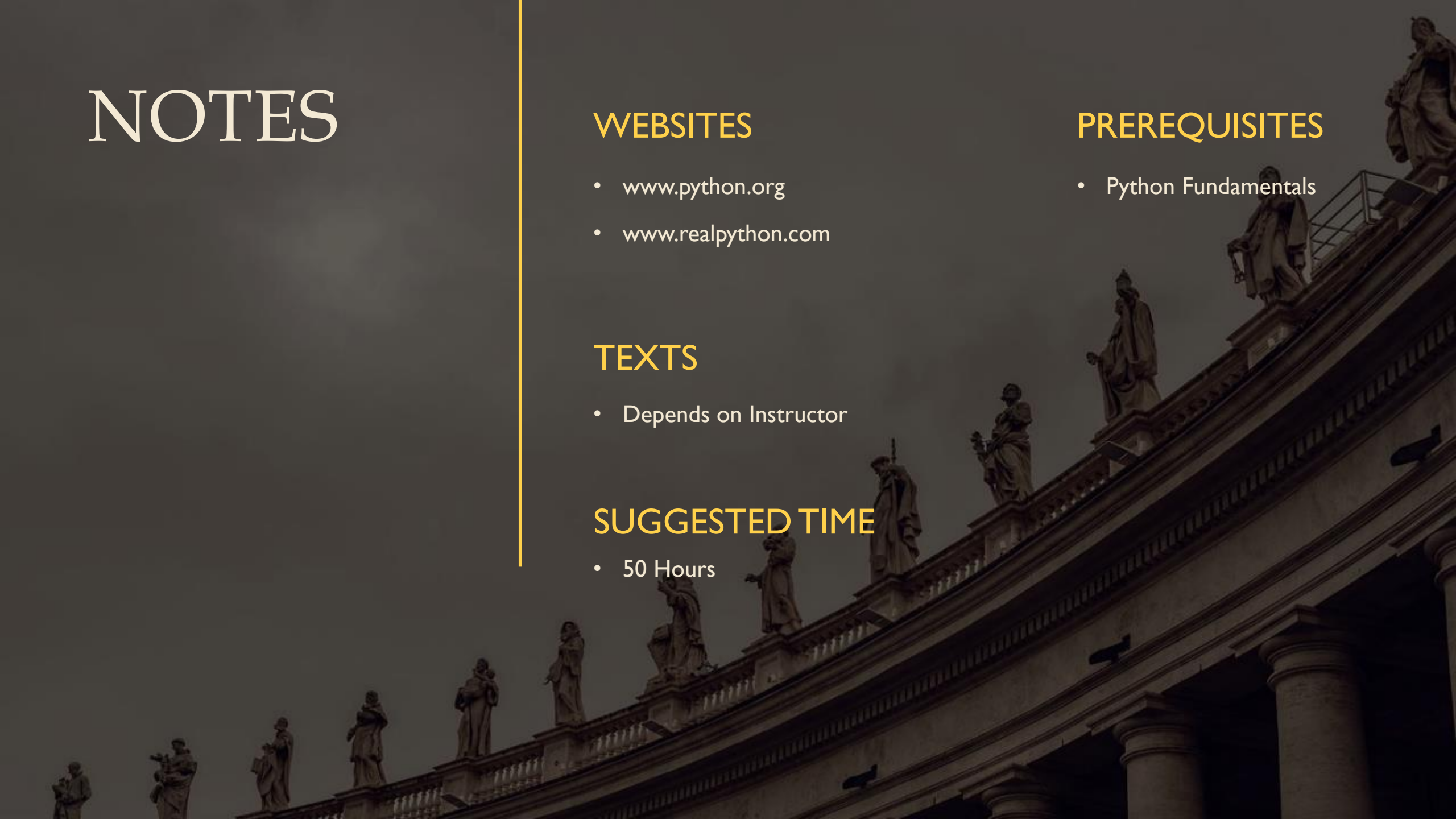
- Depends on Instructor

## SUGGESTED TIME

- 50 Hours

## PREREQUISITES

- Python Fundamentals



# DOCUMENT HISTORY

Author	Version	Revision	Date / Time	Department	Validity
Mehdi Shokri	1.0.0		16-05-2023	Development	3 Months

