ReDI School of
Digital Integration

Welcome to Backend Development (A25)

We use tech to connect human potential and opportunity with dignity & humility

Agenda

1. Introductions 👋
2. Course overview & goals 🎯
3. Python and Git review 💻

1. Introductions

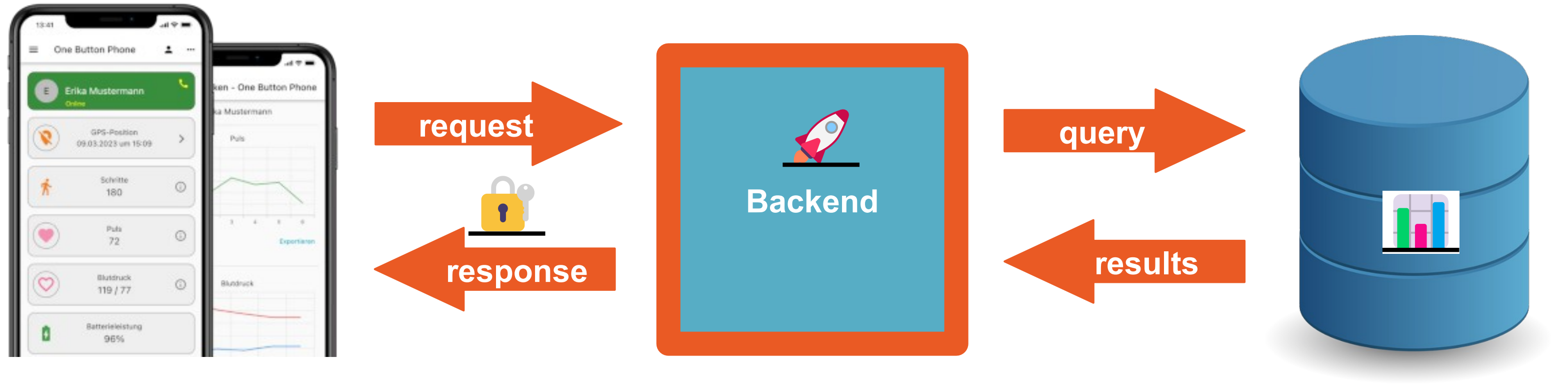
2. Course overview & goals 🎯

A look at this semeste

Date	Time	Location	Session #	Topic
Mon, Sep 9	19:00 – 21:00	Campus	1	Introduction + Welcome + Set up
Wed, Sep 11	19:00 – 21:00	Zoom	2	Python Review + Git review (if needed)
Mon, Sep 16	19:00 – 21:00	Campus	3	Testing
Wed, Sep 18	19:00 – 21:00	Zoom	4	Testing
Mon, Sep 23	19:00 – 21:00	Campus	5	Mini project 1
Wed, Sep 25	19:00 – 21:00	Zoom	6	Rest, HTTP & Working with APIs in Python
Mon, Sep 30	19:00 – 21:00	Campus	7	APIs and FastAPI
Wed, Oct 2				NO CLASS TODAY
Mon, Oct 7	19:00 – 21:00	Campus	8	Review / Practice / Q&A
Wed, Oct 9	19:00 – 21:00	Zoom	9	Static Resources and Templating
Mon, Oct 14	19:00 – 21:00	Campus	10	Mini project 2
Wed, Oct 16	19:00 – 21:00	Zoom	11	Intro to databases
Mon, Oct 21	19:00 – 21:00	Campus	12	Nonrelational databases
Wed, Oct 23	19:00 – 21:00	Zoom	13	Interacting with Databases
Mon, Oct 28	19:00 – 21:00	Campus	14	Putting API & Databases together
Wed, Oct 30	19:00 – 21:00	Zoom	15	Final project intro / review / q&a
Mon, Nov 4				CAREER WEEK
Wed, Nov 6				NO CLASSES THIS WEEK
Mon, Nov 11	19:00 – 21:00	Campus	16	Turning ideas into code
Wed, Nov 13	19:00 – 21:00	Zoom	17	Code performance
Mon, Nov 18	19:00 – 21:00	Campus	18	Mini project 3
Wed, Nov 20	19:00 – 21:00	Zoom	19	Data Structures and Algorithms
Mon, Nov 25	19:00 – 21:00	Campus	20	Design Patterns
Wed, Nov 27	19:00 – 21:00	Zoom	21	Project / final q&a / review
Sun, Dec 1	0:00			STUDENT DEADLINE: Submit final projects + presentations on GC
Mon, Dec 2	19:00 – 21:00	Campus	22	Final Project Presentations
Wed, Dec 4	19:00 – 21:00	Zoom	23	Final Project Presentations // Final day of class
Dec 5-Dec 9				NO CLASSES // Teachers grade projects
Mon, Dec 9	0:00			TEACHER DEADLINE: Submit final grades on GC
Fri, Dec 13				DEMO DAY

In this course, we will learn how to...

- Create a **Backend** with **Python**
- Use **databases**
- Use **common good practices** and **patterns**



Mobile app / Website

Python application

Database

In this course, we will learn by...

- Designing and implementing **RESTful APIs** using **FastAPI** and **MongoDB**, including **database schema design**, query operations, and handling HTTP requests and responses
- Securing an API with best practices for **authentication** and **authorization**
- Optimizing the performance of APIs using **caching techniques** and understanding the **benefits and limitations** of different caching strategies
- Applying **common software design patterns** to improve the maintainability and scalability of code

How to be successful

- **Come to class** and **participate**
    - Follow along on your own device during demos/lectures
    - **Be curious** and **ask questions** *(during lessons or anytime in slack)*
    - Do the **challenges** and **homework**
    - For online class, **cameras ON** *(or you will be marked as absent)*
- Submit and present a **final project** *(details will come later in the semester)*
- Engage with Career Services
    - Attend **2 workshops**
    - Complete 1 recommended module on the **IBM SkillsBuild**
    - See learner hub for details

3. Python & Git review 💻

Interactive

Getting started - Tools

Python

- Interpreted language
- Simple grammar
- Dynamic data type variables
- Multi-platform
- Widely used *(updated documentation and constant fixes)*

Git

- Source control *(also known as Version Control System)*
- Multi-platform
- Widely used *(updated documentation and constant fixes)*

Getting started - Installation

- Install **Python** and **Git**
- Make Python and Git available in the **$PATH** environment variable
- Install a **code editor**
- Open a **terminal**

Getting started - Git configuration

- Get Git installed version: ***$ git --version***
- Configure your identity:
  $ git config --global user.email "you@example.com" *(fill this with your email !)*

  $ git config --global user.name "your name" *(fill this with your name !)*
- Corroborate configuration: ***$ git config --global --list***

Getting started - Create a local git repository

- Create a new directory: ***$ mkdir -p ~/Codes/lesson_1 && cd ~/Codes/lesson_1***
- Create a local git repository in the current directory: ***$ git init --initial-branch main***
- Create first content:

  $ echo *"Lesson 1"* ***>> README.md***

  $ curl *https://raw.githubusercontent.com/github/gitignore/master/Python.gitignore* ***>> .gitignore***

  A well maintained ignored file for Python, which includes the virtual environment directory
- List new files: ***$ git status*** *(it should not be empty !)*
- Initialize the repository: ***$ git add --all*** *(this will include .gitignore and README.md)*
- First commit: ***$ git commit -m "chore: initialize repository"***
- List all the commits: ***$ git log***

Getting started: Python configuration

- Get Python installed version: ***$ python3 --version***
- Create a virtual environment: ***$ python3 -m venv venv*** *(it will create the directory venv)*
- Activate the virtual environment: ***$ source ./venv/bin/activate*** *(to deactivate: **$ deactivate**)*

  *Once the virtual environment is activated, we can use the **python** command instead of **python3***

- Get ***pip*** installed version: ***$ python -m pip --version***
- List the installed packages: ***$ python -m pip freeze*** *(it should be empty !)*
- Install a package with pip: ***$ python -m pip install pytest***
- List again the install packages: ***$ python -m pip freeze*** *(it should not be empty !)*
- Store of the installed packages: ***$ python -m pip freeze > requirements.txt***

Getting started - Let’s develop

- List uncommitted changes: ***$ git status*** *(it should appear requirements.txt !)*
- Inspect uncommitted changes: ***$ git diff***
- List all git branches: ***$ git branch -vv***
- Create a new branch: ***$ git branch feature/lesson-1***
- Track installed packages: ***$ git add requirements.txt***
- Commit changes: ***$ git commit -m "feature: track Python requirements"***
- List all commits: ***$ git log***

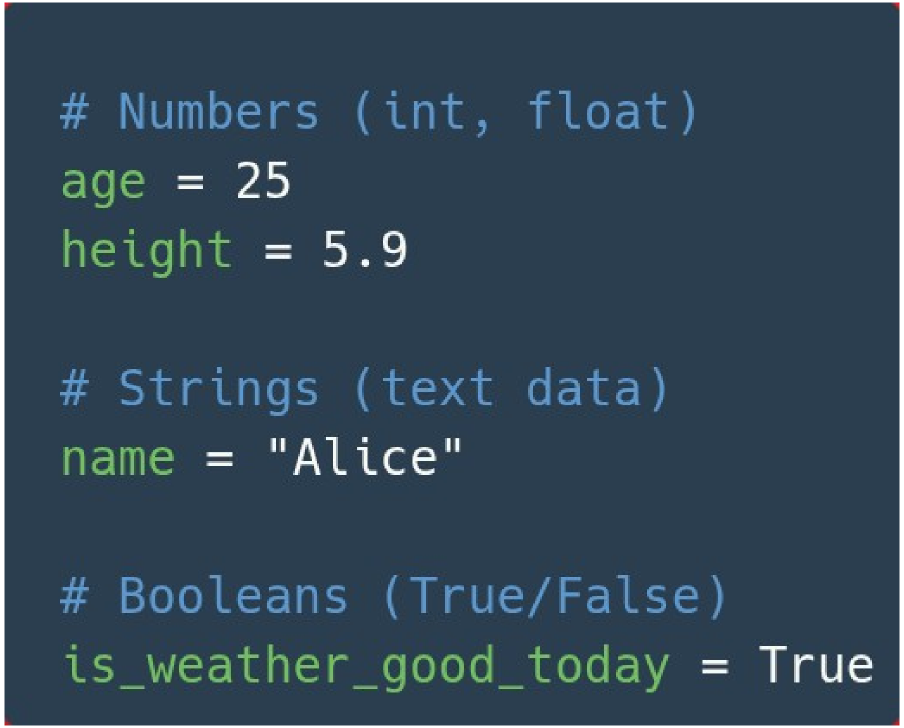
Getting started - done!

We have everything to start 🚀

Python Recap

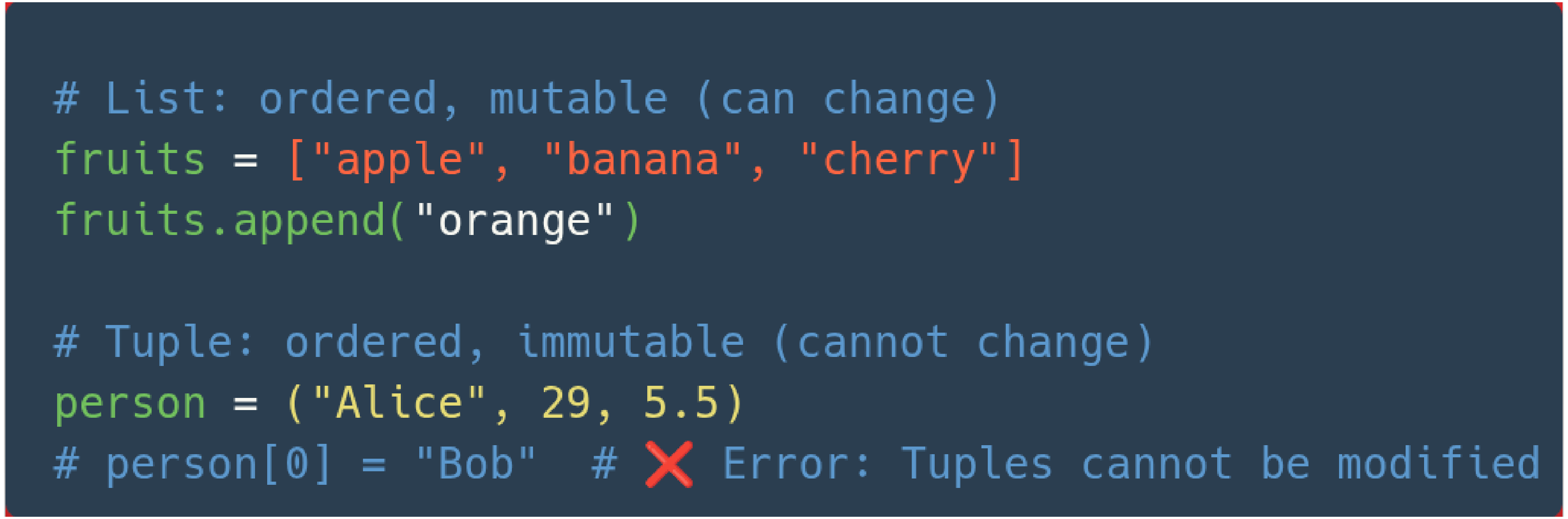
- Data types
- Operators
- Loops
- List comprehension
- Functions
- Lambda Expressions

Python Recap: Primitive Data Types



```
# Numbers (int, float)
age = 25
height = 5.9

# Strings (text data)
name = "Alice"

# Booleans (True/False)
is_weather_good_today = True
```

Python Recap: Lists and Tuples

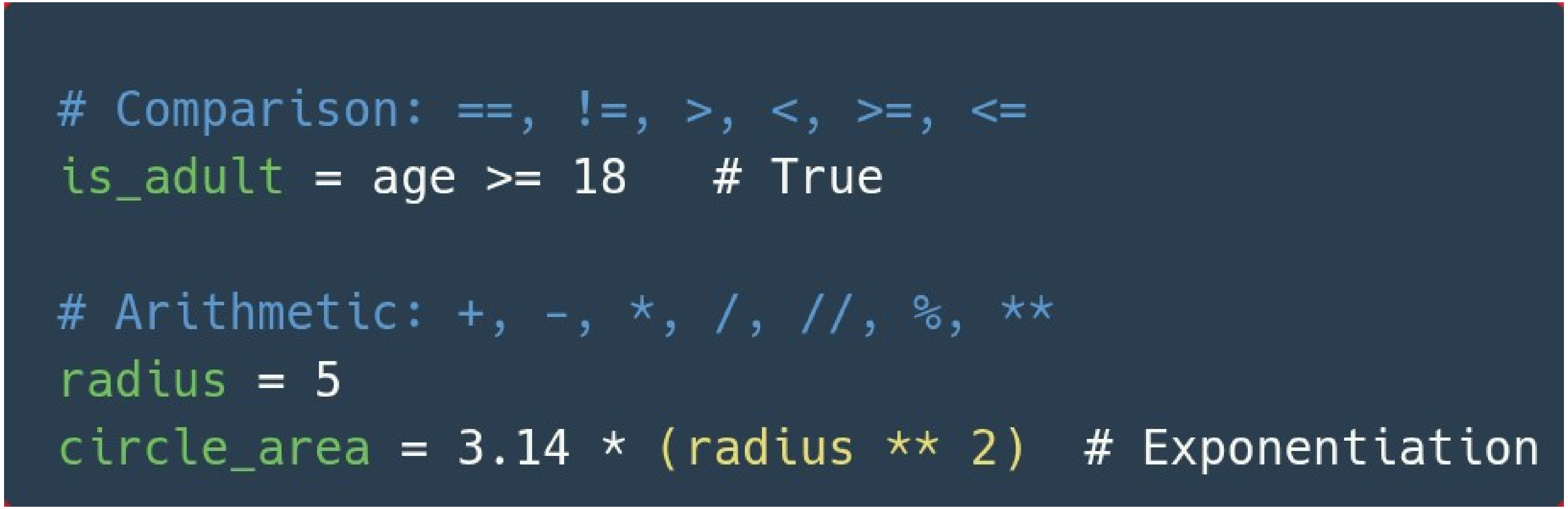


```python
# List: ordered, mutable (can change)
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")

# Tuple: ordered, immutable (cannot change)
person = ("Alice", 29, 5.5)
# person[0] = "Bob"  # ❌ Error: Tuples cannot be modified
```

Python Recap: Dictionaries and Sets

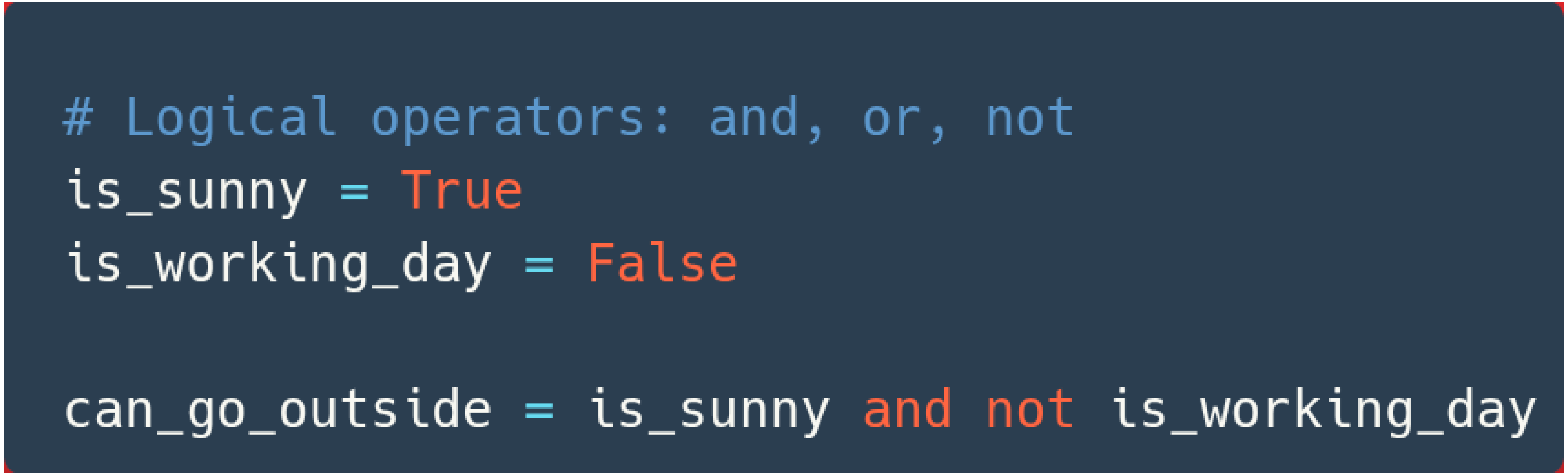
```python
# Dictionary: key → value mapping
student = {"name": "Alice", "age": 25, "city": "Munich"}
student["age"] = 26   # Update value

# Set: unique, unordered items
unique_numbers = {1, 2, 3, 4}
unique_numbers.add(3)  # No effect (already exists)
```

Python Recap: Arithmetic and Comparison



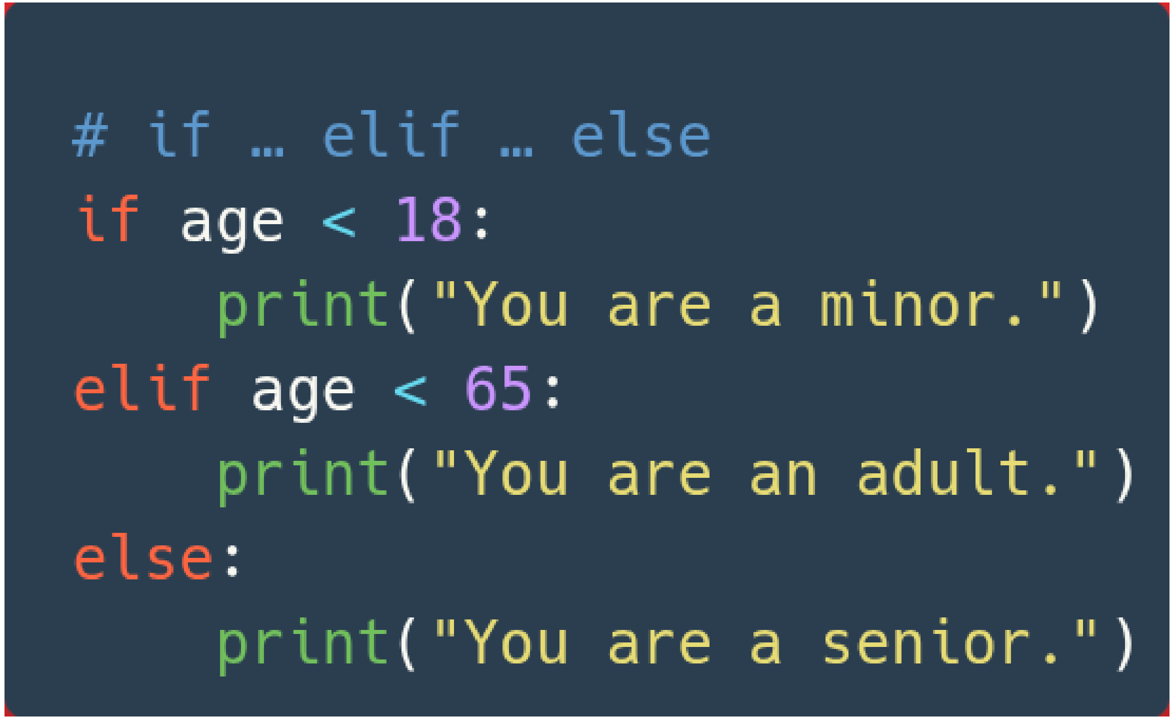
```python
# Comparison: ==, !=, >, <, >=, <=
is_adult = age >= 18   # True

# Arithmetic: +, -, *, /, //, %, **
radius = 5
circle_area = 3.14 * (radius ** 2)  # Exponentiation
```

Python Recap: Logical Operators



```python
# Logical operators: and, or, not
is_sunny = True
is_working_day = False

can_go_outside = is_sunny and not is_working_day
```

Python Recap: Conditional Statements

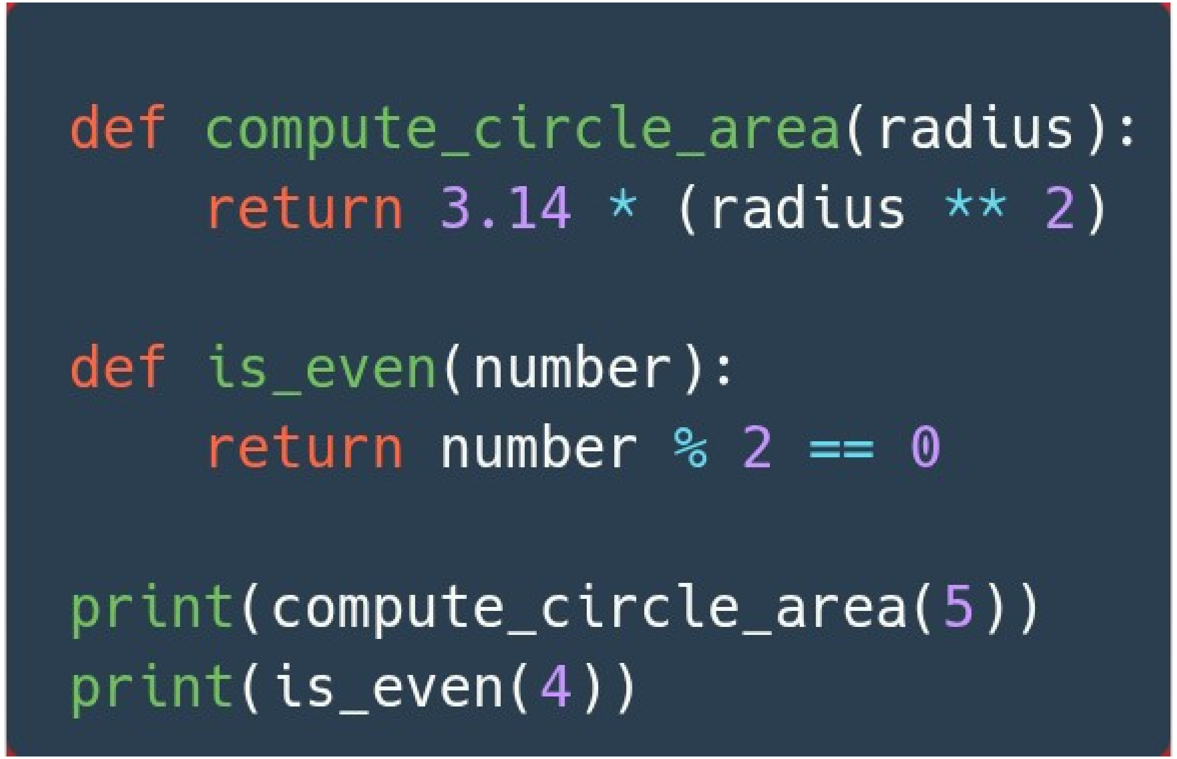


```python
# if … elif … else
if age < 18:
    print("You are a minor.")
elif age < 65:
    print("You are an adult.")
else:
    print("You are a senior.")
```

Python Recap: Using Functions

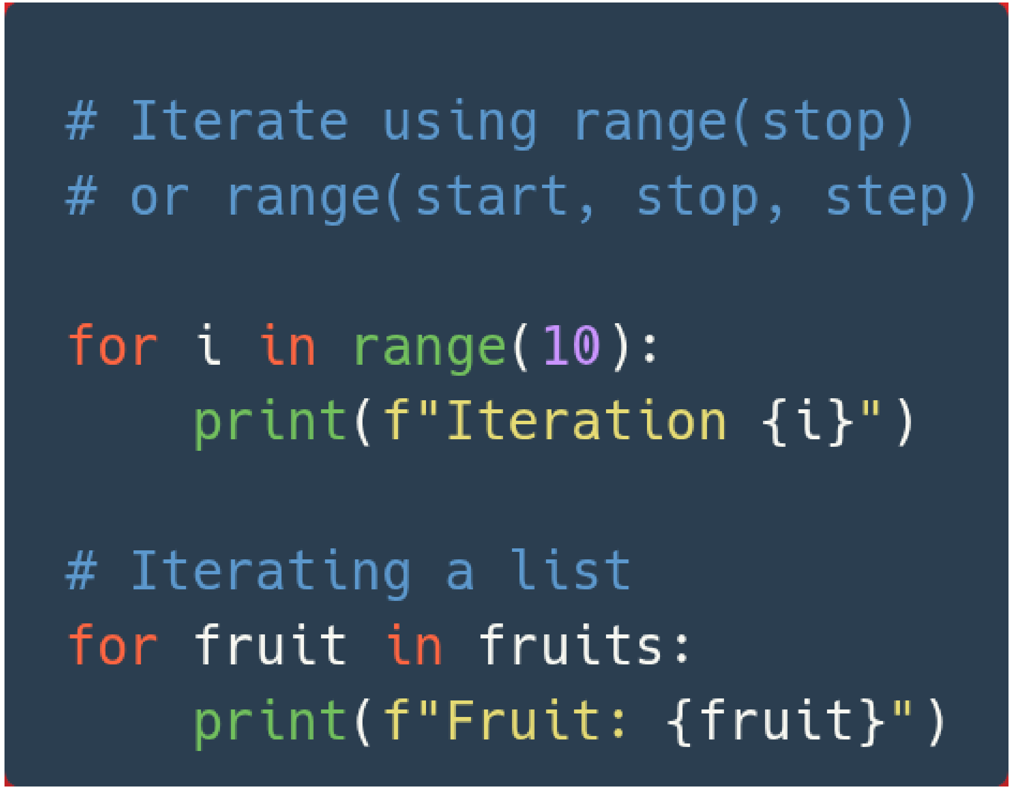
```python
# Built-in functions for math
numbers = [1, 2, 3, 4, 5]
average = sum(numbers) / len(numbers)

# Using datetime to get weekday
import datetime
is_working_day = datetime.date.today().weekday() < 5  # 0–4 = Mon–Fri
```

Python Recap: Defining Functions



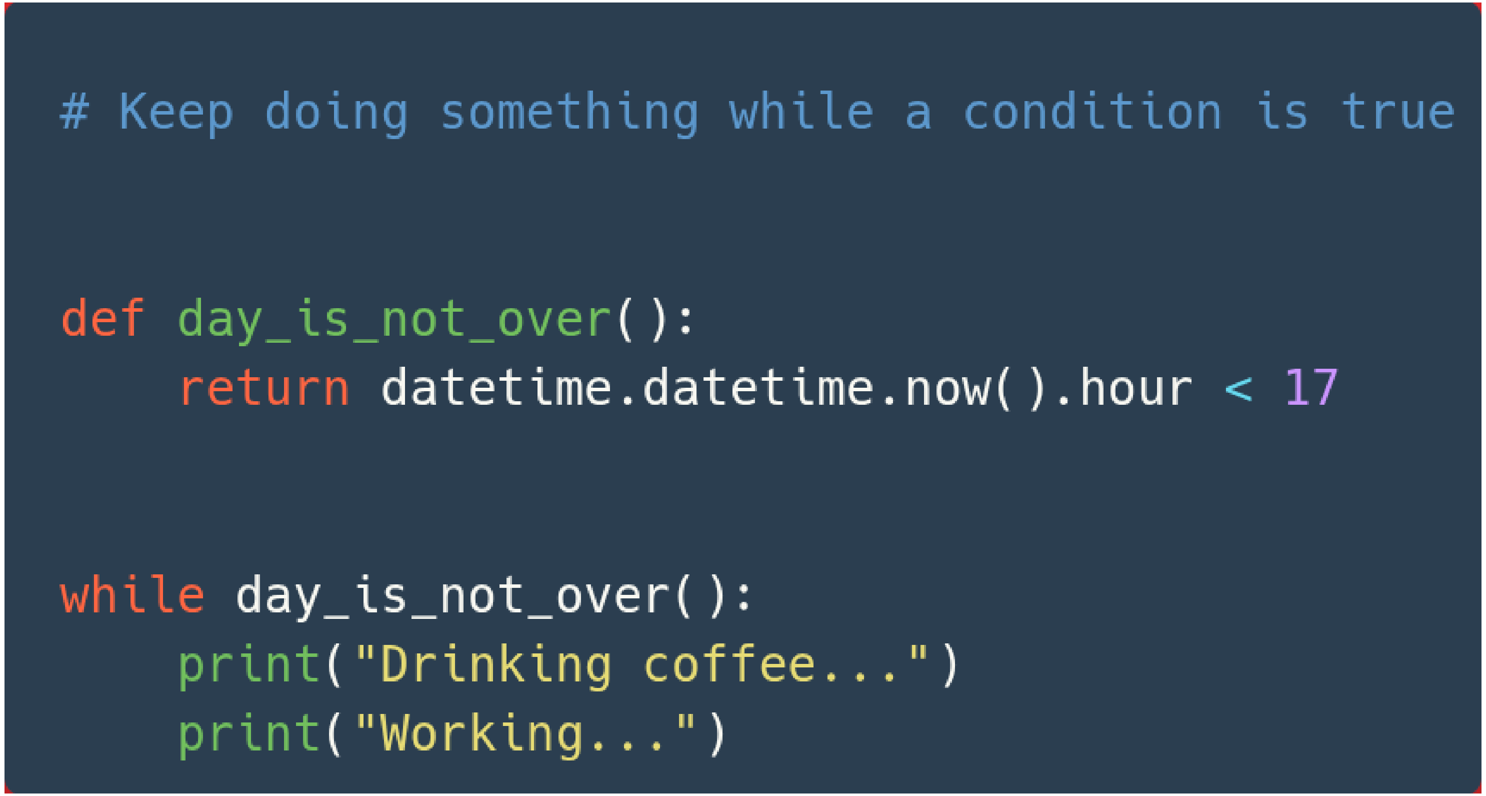
```python
def compute_circle_area(radius):
    return 3.14 * (radius ** 2)

def is_even(number):
    return number % 2 == 0

print(compute_circle_area(5))
print(is_even(4))
```

Python Recap: Iterating with for



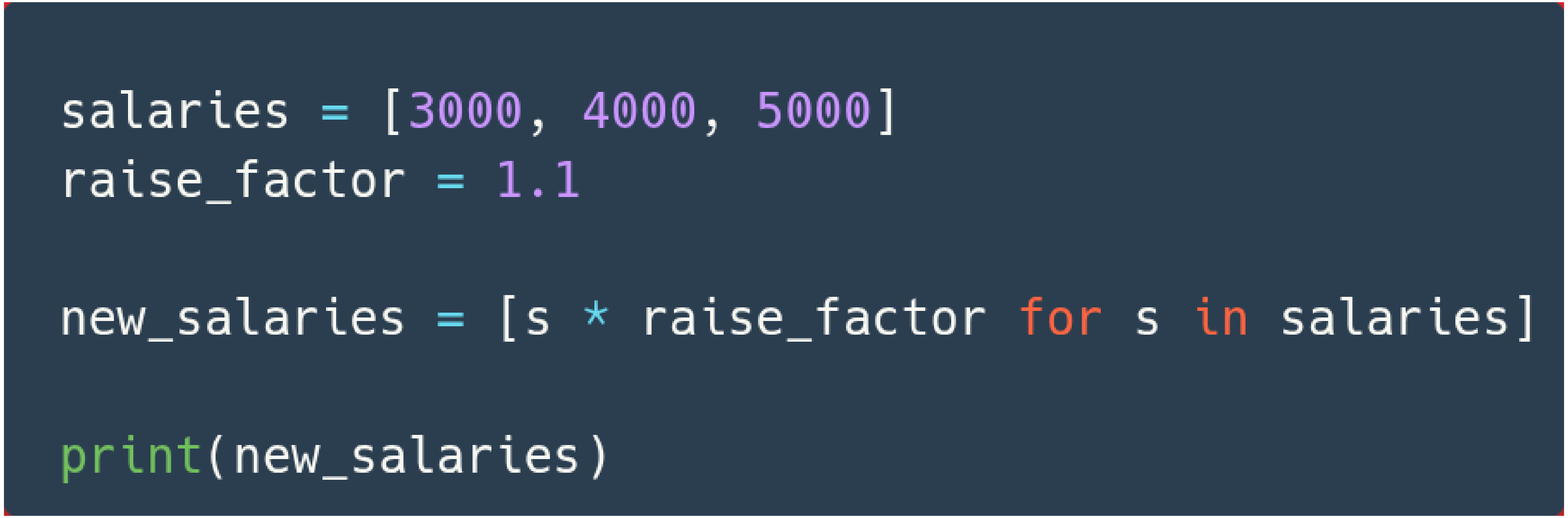
```
# Iterate using range(stop)
# or range(start, stop, step)

for i in range(10):
    print(f"Iteration {i}")

# Iterating a list
for fruit in fruits:
    print(f"Fruit: {fruit}")
```

Python Recap: Looping with while



```
# Keep doing something while a condition is true


def day_is_not_over():
    return datetime.datetime.now().hour < 17


while day_is_not_over():
    print("Drinking coffee...")
    print("Working...")
```

Python Recap: List Comprehension



```python
salaries = [3000, 4000, 5000]
raise_factor = 1.1

new_salaries = [s * raise_factor for s in salaries]

print(new_salaries)
```

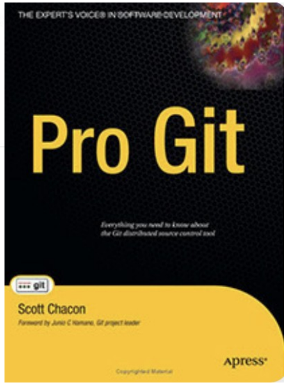
Python Recap: Lambdas

```python
# Lambda is a small anonymous function

purchases = [(10.99, 2), (5.49, 5), (3.99, 1)]  # (price, qty)

# Sort by total cost (price * qty)
purchases.sort(key=lambda item: item[0] * item[1])
```

Links and further materials

Books!

- Fluent Python - *Luciano Ramalho*
- Pro Git - *Scott Chacon*
- Git Book - https://git-scm.com/book/en/v2 *free and in many languages*



Links!

- Python tutorial https://docs.python.org/3/tutorial
- Python turtle https://docs.python.org/3/library/turtle.html
- Conventional commits https://www.conventionalcommits.org

Videos!

- Pretty recent setup of *pip and venv in VS Code* : https://youtu.be/GZbeL5AcTgw
- Install git on Windows: https://www.youtube.com/results?search_query=installing+git+on+windows+11

Homework

Voluntary but good way to get some practice!

- Fork a repository: e.g., https://github.com/bkircher/intro-python
- Create a PR or comment on one: e.g., https://github.com/bkircher/intro-python/pull/1
- Check out your favourite Open Source project and look how people do report issues, create pull requests, and look out for a *CONTRIBUTING.md* (https://en.wikipedia.org/wiki/Contributing_guidelines) or labels like "*good-first-time-contribution*"

Bonus points:

- Refresh your Python and git skills: Create a new empty repository, push to your own GitHub/Gitlab, write a script named *echo.py* that reads continuously from STDIN into a variable and writes it back to STDOUT again. Ctrl+C should exit the script. Share link to *your* repository in Slack with the others!
  - Hints: use input() built-in function to read from STDIN, use print() built-in function to output on STDOUT, use while True for an endless loop and check out KeyboardInterrupt exception!
  - Links:
    - https://docs.python.org/3/library/functions.html
    - https://docs.python.org/3/library/exceptions.html#Exception

Thank You !