NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Applied Mathematics and Informatics"

Software Project Report on the Topic:
Development of Materials for Teaching Programming in Python

Fulfilled by:

Student of the Group
БПАД212
Kulakov Denis Alexeevich

____
(signature)

May 31st, 2024
(date)

Checked by the Project Supervisor:

Voznesenskaya Tamara Vasilievna
Associate Professor
Big Data and Information Retrieval School, Faculty of CS, HSE University

____
(signature)

June 2nd, 2024
(date)

Moscow 2024

Table of contents

Abstract

This project aims at addressing the increasing demand for quality secondary-level computer science education. The goal is to ultimately create a complete Python curriculum that is accessible, free to students and open-source. The project uses contemporary software development applications like Docker, Git among others and is designed to ensure that students not only listen and learn the basics of programming but also gain an understanding of how things are really done in today's business climate. Students gain experience creating consistent development environments using containerization, Docker. Learn git: Version control and collaborative work are crucial in any software related work, and git makes you use version control along with a collaborative workflow.

The contents consist of slide decks, code examples, and self-evaluation resources to support an interactive and engaging learning process. It highlights practical examples of Python, showing it in action in all sorts of real-world situations to encourage and motivate students. This project stands out, compared to the current educational programs, for the integration of modern development practices and is stronger in a comparative analysis.

Introduction

The project, 'Developing material for the study of Python programming,' is intended to produce teaching and learning material focused on the teaching of Python programming to secondary school students. Python has become a language of choice for teaching introductory computer science in many places because of its cleanness and flexibility for teaching both basic programming constructs and complex applications. In the end of this description, you will find a recent article on the project's desire to create complete and pedagogically rich learning objects that are neither text-based nor text-centric, meeting the demands of the contemporary academic environment.

The course introduces students to modern industry standards and practices using the contemporary software development tools; namely Docker and Git. In addition, to Docker, we remove the various local development system configuration disparities by providing lightweight and self-contained computing environments. Students learn distributed version control with Git, which is a prelude to collaborative and iterative development workflows leveraged in the software industry This project is a master class to develop an expert understanding and create sophisticated programming concepts based on basics that is directed to more advanced studies for students during their learning experiences.

The end game, is a true educational supplement, something that not only learns your tech, but give you tools to tame the tangled world of modern computational problem-solving. The project aims at systematically considering this goal as an evidence-based aspect of what computer science education contributes to a technology workforce pipeline.

Subject area

The goal of this project is to raise public awareness to computer science education and its emphasis on:

Contemporary software development tools:

- **Docker -** Explains the concepts of containerization, and teaches students to create applications in development environments.

- **Git -** Version control, collaborative workflow, and versioning in developing software.

Real-world applications:

- **Practical Python Programming**, which shows students what they can do with Python and how to use it for all types of real-world scenarios.

Relevance

For a number of reasons, the project has implications that are all too significant in the educational and technological landscape of today.

1. **Wide application**
   - 1.1. Python has broad application as well, for applications from internet development and automation to data analysis and machine learning. Because of this, learning it can be very useful to students learning it can help in applying it differently.
   - 1.2. Its wide use in professional and educational settings helps provide a robust community and plethora of learning resources.
2. **Industry demand**
   - 2.1. Programming skills are in high demand, from tech to finance and healthcare, and many other fields. And python specifically is used very frequently because of how robust it is and how easy it is to learn.
   - 2.2. Career opportunities.
3. **Educational needs**
   - 3.1. Programming is becoming more widespread in the curricula of schools and educational institutions in order to equip students with skills that are highly valued in the job market of the future
   - 3.2. Python has a syntax that is simple and easy to read, making it suitable for beginners to understand most programming concepts. This reduces the barrier to learn and open the door for more people to learn programming.
4. **Integration of modern development practices**
   - 4.1. **Docker usage:** Telling students to use Docker is a good way to make them understand what containers are and why they are important for modern software development. It is practicing what

we preach which is essential for real world application as it gives us practical experience working in creating consistent development environments. Docker is among the rising trends in the hiring market to follow down on — the availability of Docker as the must-have skill has become a joke with tips for getting a DevOps job.

4.2. **Git:** Teaching Git, but this falls in the line with the previous tip and this is just how you usually do things. GIT helps students to manage code changes, collaborate with others on projects and understand how changes are introduced and they track each change. Students going through an educational environment will appear more attractive on the job market when they already have the foundation with how to use Git as it gets used in the majority of development workflows out there today.

Main goal

It is a project aimed at providing basic understanding in programing on Python to school students and general know how of software development.

Main tasks

1. Go through the Materials Given by the Supervisor
2. Review analogues and alternatives to accessible analogues in existence
3. Create the courseware and the concept of the course.
4. Create the materials
5. Write the project report
6. Results

Instruments

1. Google Drive
2. Google Docs
3. Jupyter Notebook
4. Google Slides
5. Git
6. Docker
7. Google Collab
8. SlidesGo

Basic terms and definitions

Programming language – an artificial language designed to communicate and enforce calculations that may be performed by a tool, most generally a computer.

Python — A hopeful programming language that is high-level, interpreted mentioning in readability, and applicable with scripting and software development tasks.

Educational materials - required learning material, such as texts, presentations and interactive exercises.

Git - is a version control system for tracking changes in source code during software development.

Repository (repo) – where the project's files and history are stored.

Commit - this is a save of the work you have done on your project (i.e this is what your project looked like at a particular point in time)

Branch - A distinct line of development.

Branch Merge - Pull changes from various branch or repository.

Clone - This means copying an already existing repository.

Remote - Your repository that is hosted on the internet or a network

Staging area - Where your changes are staged before you commit.

Pull request (merge request) - A request to merge one branch of code into another (usually from a developer's branch into the main or development branches).

Rebasing - A sequence of commits can be moved or combined onto a new base commit with rebasing. It is mostly used for the synchronization of a feature branch with the main branch.

Cherry-picking - taking a commit from one branch and attaching to another.

Stashing - used to temporarily save you changes (not commit)

Docker - is complete ecosystem (platform as a service) based on the OS-level virtualization i.e., containers.

Image - A lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, and dependencies

Container - A container is a running instance of an image.

Docker file - Text file that includes a list of commands to build a Docker image.

Docker hub - a cloud-based registry in which Docker users and partners make use of Docker.

Project implementation

This collection of branches and stages of work is structured to construct a broad educational program which is efficient for secondary school students in the project. This list of tasks can be found below:

1. **Master classes for school students:**

   1.1. Curriculum development: defining a list of Python programming concepts that could be made into master class topics.

Applications of Python

```
                                 from the website
       = requests.get(url)

# checking response.status_code (if you get 502, try
if response.status_code != 200:
    print(f"Status: {response.status_code} - Try
else:
    print(f"Status: {response.status_code}\n")

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

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

# downloading images
```

Real-world Applications:

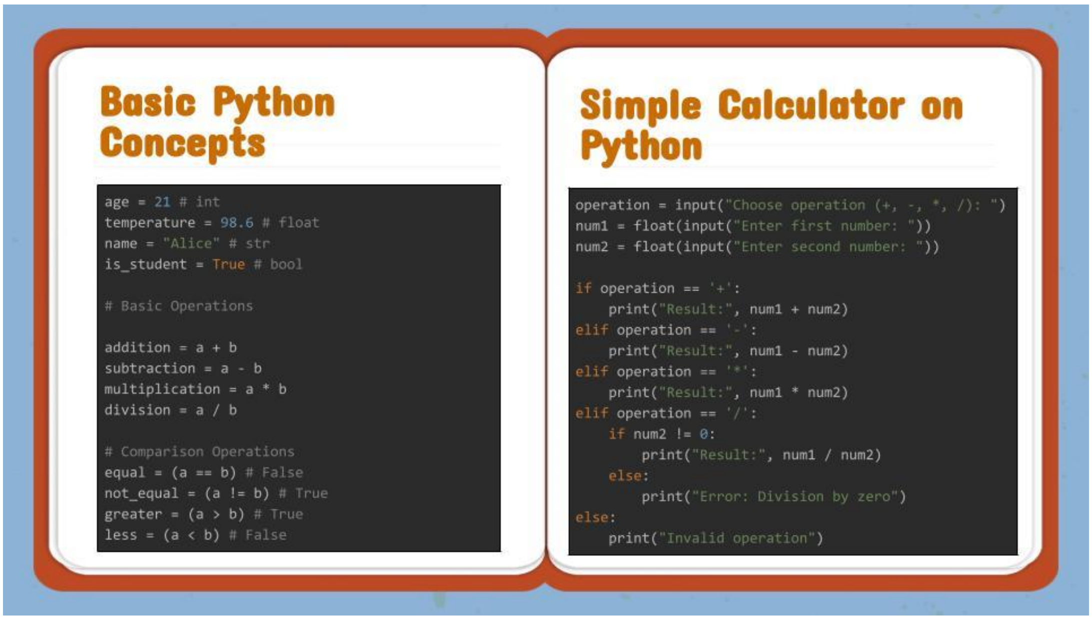
- Web development (Django, Flask)
- Data science (Pandas, NumPy, Matplotlib)
- Automation (scripting, task automation)
- Artificial intelligence (TensorFlow, scikit-learn)

Companies Using Python:

And many more!

Picture 1. Real-world applications for Python

1.2. Creating resource: Create additional material such as slide decks and code examples.



Basic Python Concepts

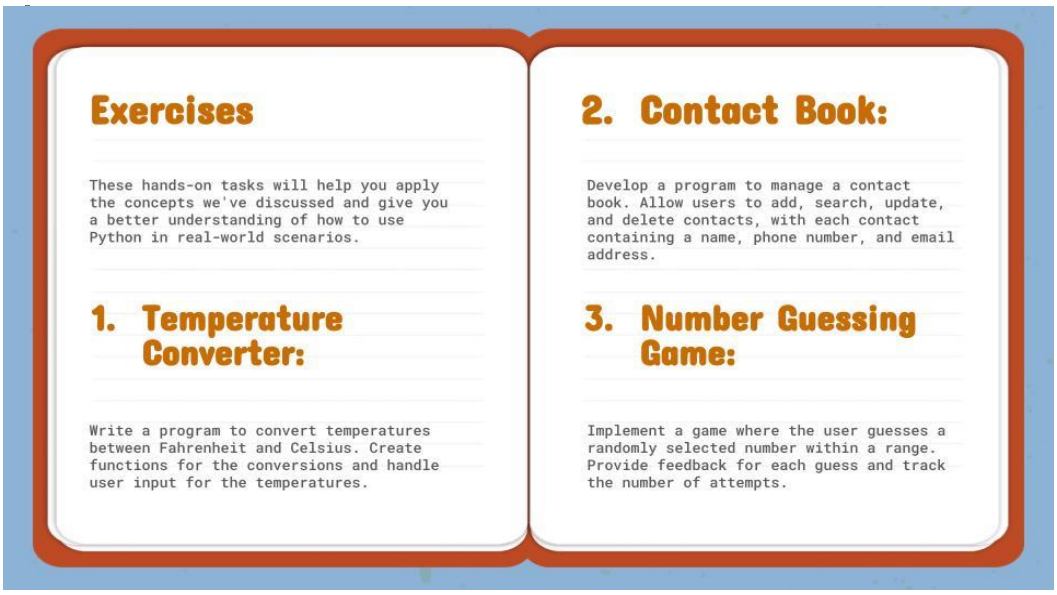
```
age = 21 # int
temperature = 98.6 # float
name = "Alice" # str
is_student = True # bool

# Basic Operations

addition = a + b
subtraction = a - b
multiplication = a * b
division = a / b

# Comparison Operations
equal = (a == b) # False
not_equal = (a != b) # True
greater = (a > b) # True
less = (a < b) # False
```

Simple Calculator on Python

```
operation = input("Choose operation (+, -, *, /): ")
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

if operation == '+':
    print("Result:", num1 + num2)
elif operation == '-':
    print("Result:", num1 - num2)
elif operation == '*':
    print("Result:", num1 * num2)
elif operation == '/':
    if num2 != 0:
        print("Result:", num1 / num2)
    else:
        print("Error: Division by zero")
else:
    print("Invalid operation")
```

Picture 2. Code Snippets

1.3. Create self-assessment instruments to evaluate student understanding, and evaluate how effective the sessions are in the form of master classes.



Exercises

These hands-on tasks will help you apply the concepts we've discussed and give you a better understanding of how to use Python in real-world scenarios.

1. Temperature Converter:

Write a program to convert temperatures between Fahrenheit and Celsius. Create functions for the conversions and handle user input for the temperatures.

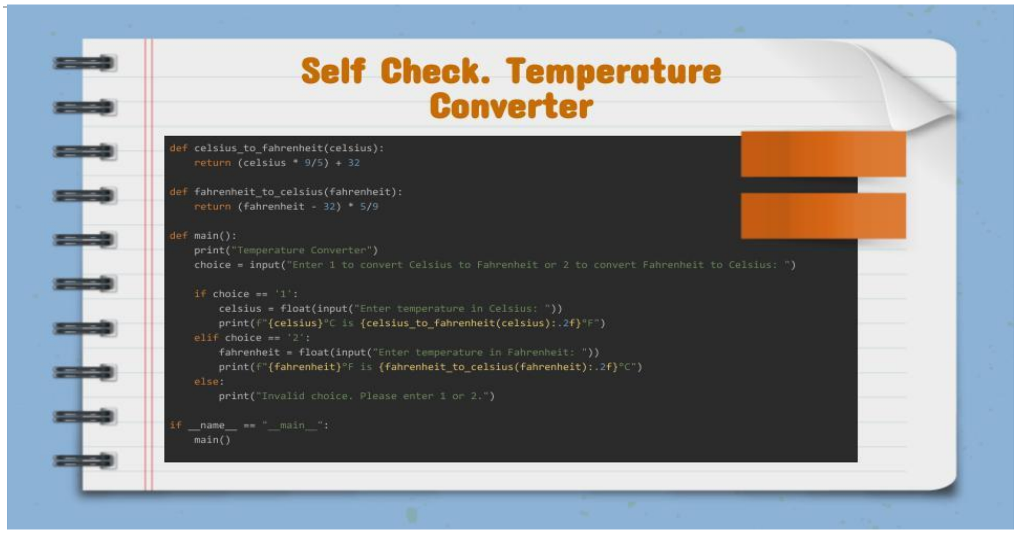
2. Contact Book:

Develop a program to manage a contact book. Allow users to add, search, update, and delete contacts, with each contact containing a name, phone number, and email address.

3. Number Guessing Game:

Implement a game where the user guesses a randomly selected number within a range. Provide feedback for each guess and track the number of attempts.

Picture 3. Exercises example



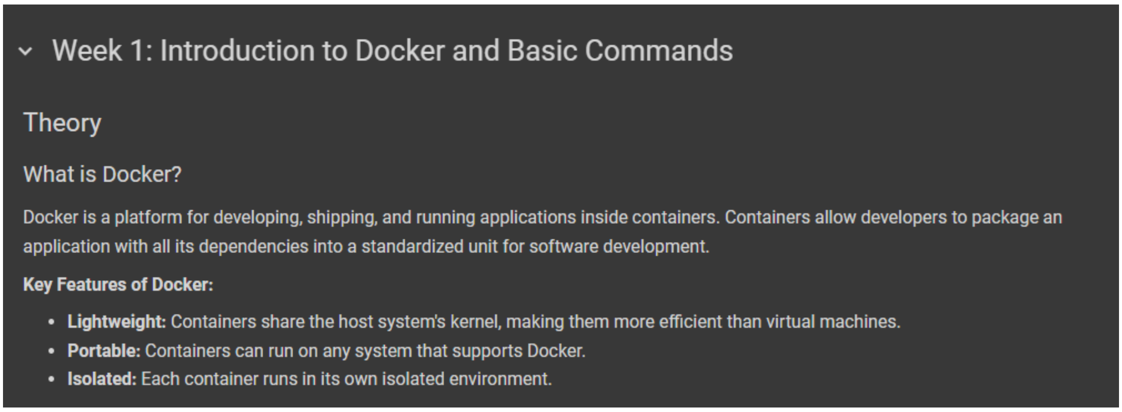
Self Check. Temperature Converter

```
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5/9

def main():
    print("Temperature Converter")
    choice = input("Enter 1 to convert Celsius to Fahrenheit or 2 to convert Fahrenheit to Celsius: ")

    if choice == '1':
        celsius = float(input("Enter temperature in Celsius: "))
        print(f"{celsius}°C is {celsius_to_fahrenheit(celsius):.2f}°F")
    elif choice == '2':
        fahrenheit = float(input("Enter temperature in Fahrenheit: "))
        print(f"{fahrenheit}°F is {fahrenheit_to_celsius(fahrenheit):.2f}°C")
    else:
        print("Invalid choice. Please enter 1 or 2.")

if __name__ == "__main__":
    main()
```

Picture 4. Solutions for self check

2. **Containerization concepts with Docker:**

2.1. Introducing containers, and why it is useful



Week 1: Introduction to Docker and Basic Commands

Theory

What is Docker?

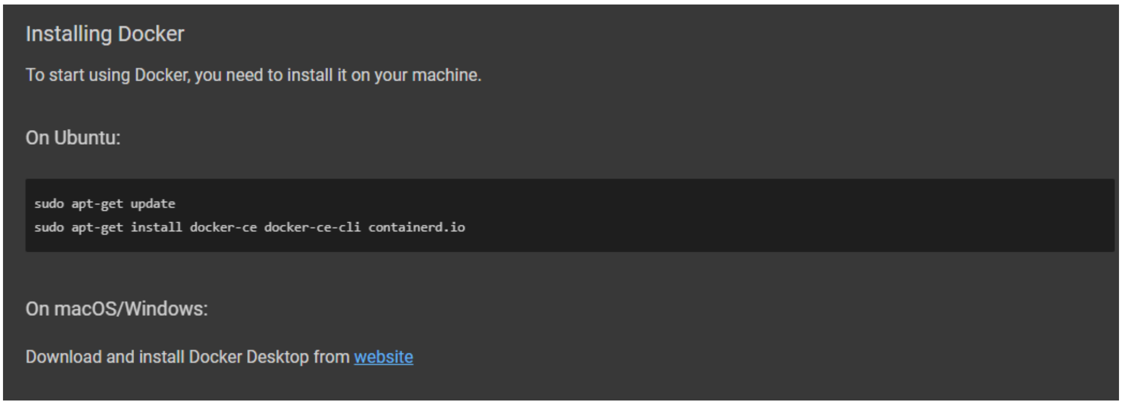
Docker is a platform for developing, shipping, and running applications inside containers. Containers allow developers to package an application with all its dependencies into a standardized unit for software development.

Key Features of Docker:

- **Lightweight:** Containers share the host system's kernel, making them more efficient than virtual machines.
- **Portable:** Containers can run on any system that supports Docker.
- **Isolated:** Each container runs in its own isolated environment.

Picture 5. Key features

2.2. Create a article tutorial on the setup and running of Docker Containers.



Installing Docker

To start using Docker, you need to install it on your machine.

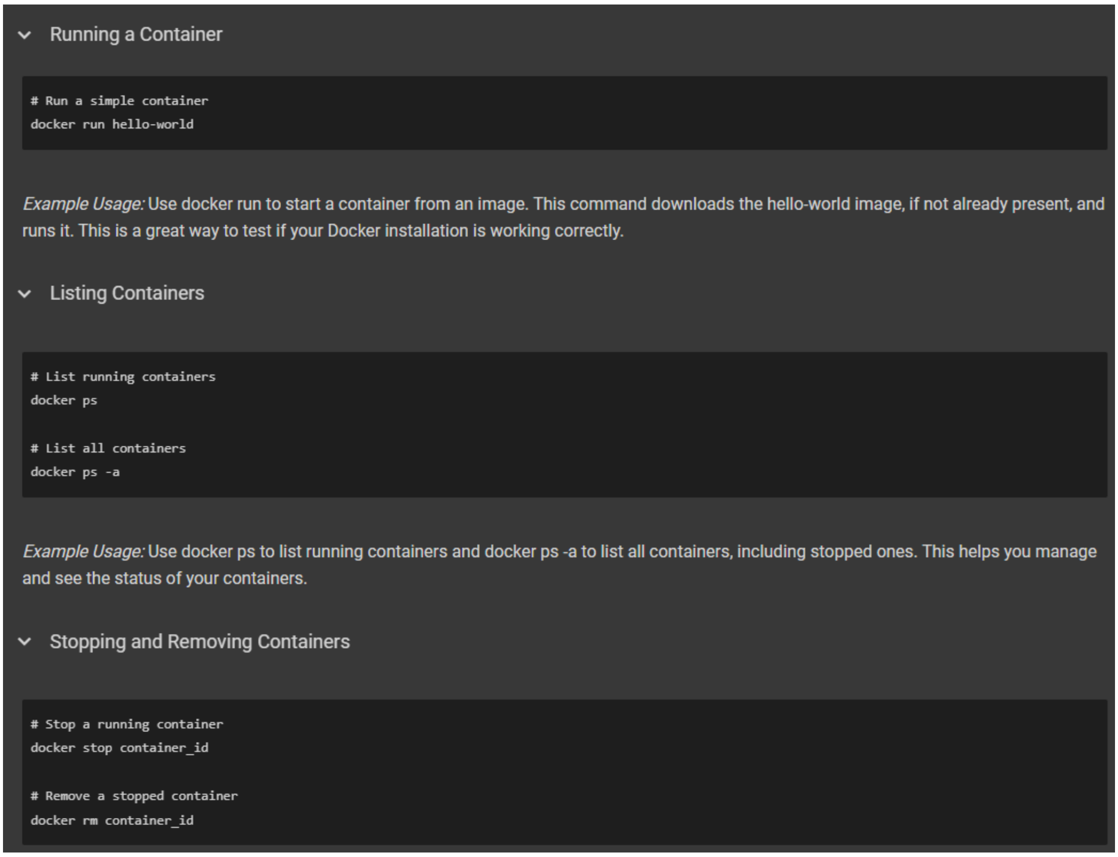
On Ubuntu:

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

On macOS/Windows:

Download and install Docker Desktop from website

Picture 6. Installation guide



Running a Container

```
# Run a simple container
docker run hello-world
```

Example Usage: Use docker run to start a container from an image. This command downloads the hello-world image, if not already present, and runs it. This is a great way to test if your Docker installation is working correctly.

Listing Containers

```
# List running containers
docker ps

# List all containers
docker ps -a
```

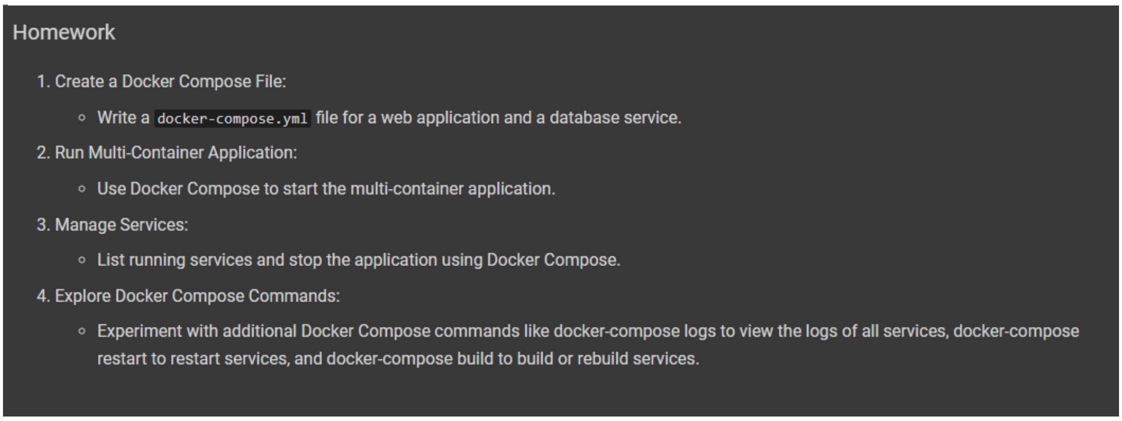
Example Usage: Use docker ps to list running containers and docker ps -a to list all containers, including stopped ones. This helps you manage and see the status of your containers.

Stopping and Removing Containers

```
# Stop a running container
docker stop container_id

# Remove a stopped container
docker rm container_id
```

Picture 7. Management of docker container

2.3. Do practical exercises that lead student on how to build, run and manage Docker containers.

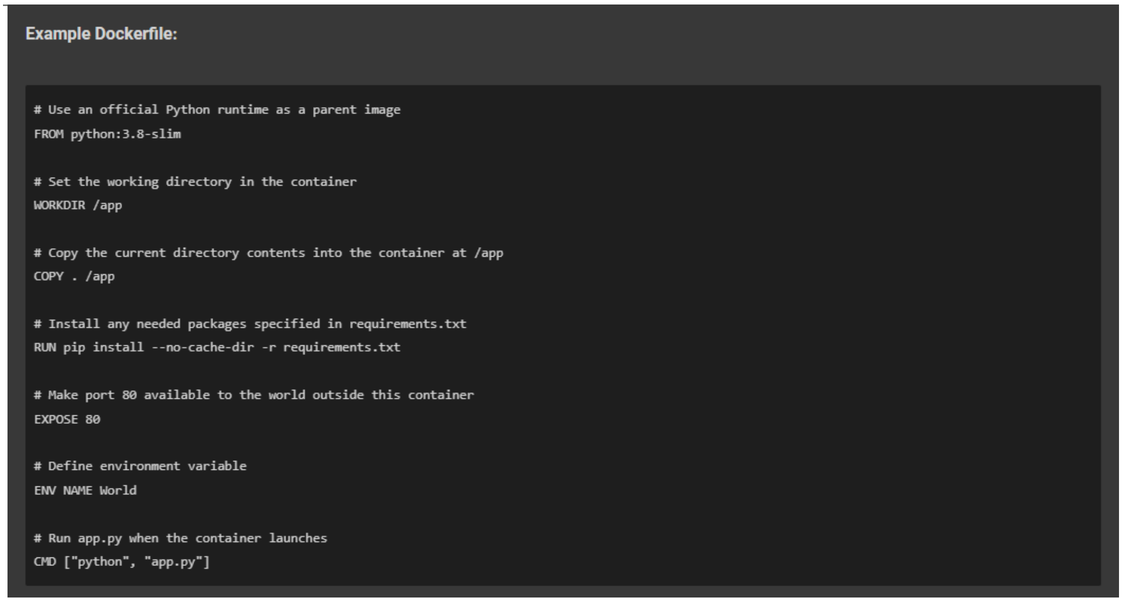


Homework

1. Create a Docker Compose File:
   - Write a `docker-compose.yml` file for a web application and a database service.
2. Run Multi-Container Application:
   - Use Docker Compose to start the multi-container application.
3. Manage Services:
   - List running services and stop the application using Docker Compose.
4. Explore Docker Compose Commands:
   - Experiment with additional Docker Compose commands like docker-compose logs to view the logs of all services, docker-compose restart to restart services, and docker-compose build to build or rebuild services.

Picture 8. Example of exercises

2.4. Integrate Docker learning modules into the Python curriculum to solidify the concept of containerizing software.



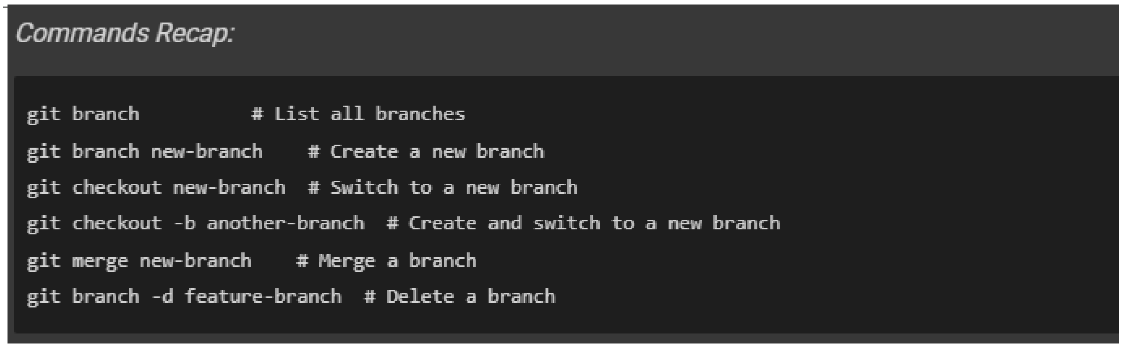
Example Dockerfile:

```
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Picture 9. Usage of Docker with Python parent image

3. Working with Git:

3.1. Prepare a list of starting points for Git commands such as cloning a remote repository, making changes, committing changes, and then pushing changes back to the remote repository.

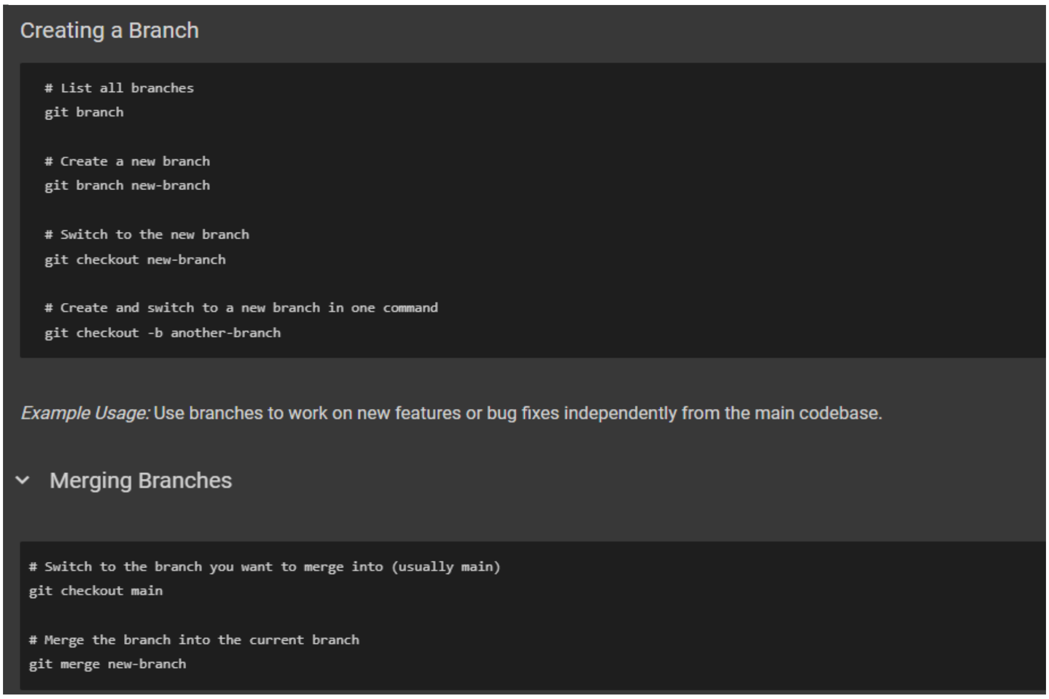


Commands Recap:

```
git branch          # List all branches
git branch new-branch    # Create a new branch
git checkout new-branch  # Switch to a new branch
git checkout -b another-branch  # Create and switch to a new branch
git merge new-branch    # Merge a branch
git branch -d feature-branch  # Delete a branch
```

Picture 10. Recap of commands

3.2. Teach students what branching strategies are, how to manage a merge conflict correctly.



Creating a Branch

```
# List all branches
git branch

# Create a new branch
git branch new-branch

# Switch to the new branch
git checkout new-branch

# Create and switch to a new branch in one command
git checkout -b another-branch
```

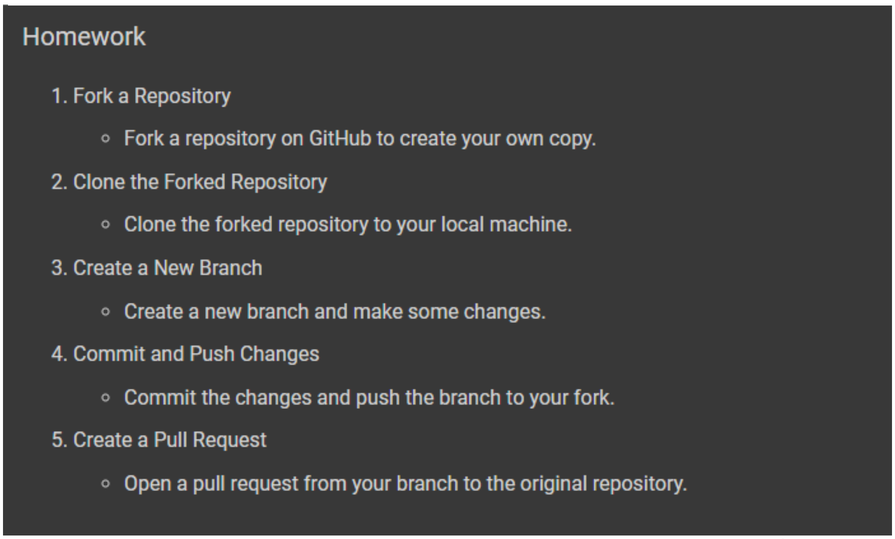
Example Usage: Use branches to work on new features or bug fixes independently from the main codebase.

Merging Branches

```
# Switch to the branch you want to merge into (usually main)
git checkout main

# Merge the branch into the current branch
git merge new-branch
```

Picture 11. Branching commands example

3.3. Create tasks how to explore materials on collaboration with Git.



Homework

1. Fork a Repository
   - Fork a repository on GitHub to create your own copy.
2. Clone the Forked Repository
   - Clone the forked repository to your local machine.
3. Create a New Branch
   - Create a new branch and make some changes.
4. Commit and Push Changes
   - Commit the changes and push the branch to your fork.
5. Create a Pull Request
   - Open a pull request from your branch to the original repository.

Picture 12. Exercises for collaboration

Existing alternatives and analogues: comparative analysis

In order to disclose the project positioning and a potential impact, let me briefly review the current state of education programs in Russia:

1) **Yandex Lyceum:** A 2-year python programming course for high school kids
2) **Kaspersky Cybersecurity for the Next Generation:** Cybersecurity and programming basics.
3) **Moscow Coding School:** Programming courses (Python) for all ages
4) **Codewards:** Deliver educational coder courses for school students in the form of games

While those programs are extensively theoretical and academic, this project is a practical one oriented at modern tooling hands-on to 21st century software development and version control. The materials serve as a guide and are meant to be modified according to the teaching technique and the educational environment.

In conclusion, printable educational materials that will be flexible and adaptable (for other teaching styles and educational needs) are created. As opposed to the very pre-programmed curricula the state or country has only some existing programs have.

Results and conclusion

Results description

A 4 weeks course on Git, 2 weeks course on Docker, and a Python workshop for high school students. This is the whole list (downloadable from shared git repository):

Perspectives for future work

This project has a few different perspectives for future work:

1) To build a website for education, exercise and training videos
2) Develop teacher training programs with the new curriculum
3) Create A Student and Teacher Community to share insights and collaborate on projects.

Conclusion

The purpose of the project “Developing Education Materials for Teaching Python Programming” is to deliver well-structured exhaustive educational materials especially for secondary school students so they can soon become productive. This is done by promoting Python as the simple and strong language of choice for the real world.

Focus on modern delivery — use of development tools like Docker and Git was seen during the project. We ensure students learn to utilise the same type of rigorous development settings that are in use at companies, such as Git for version control workflows, and Docker for creating consistent development environments, and do so at a distributable classroom level and the students walk away with a very marketable set of skills.

These resources are designed for you to be able to do interact, and can be used in a number of learning environments. It offers a hands-on experience with Python, and incorporates self-check questions to help students realize the worth of their learning efforts and ascertaining to what extent they are taking advantage of the course.

This project, which is useful in making a unique contribution to the others in terms of the integration of contemporary tools and the learning steps to be adaptable, hashed against other similar protocols. By offering something as is not currently available, this approach is not just filling gaps in what we offer today, it is also upping what the high school level of computer science can be.

Sources

1. Python Software Foundation. Python Programming Language. Retrieved from https://www.python.org/.
2. Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, 2014(239), 2.
3. Chacon, S., & Straub, B. (2014). *Pro Git*. Apress.
4. Yandex. Yandex.Lyceum. Retrieved from https://lyceum.yandex.ru/.
5. Kaspersky Lab. CyberSecurity for the Next Generation. Retrieved from https://www.kaspersky.com/.
6. Moscow Coding School. Programming Courses. Retrieved from https://codingschool.moscow/.
7. Codewards. Game-based Learning Platform. Retrieved from https://www.codewards.com/.
8. GitHub. The Git Distributed Version Control System. Retrieved from https://github.com/.
9. SlidesGo. Free Google Slides and PowerPoint Templates. Retrieved from https://slidesgo.com/.