Principles of Software Engineering and Data Bases

Davide Yi Xian Hu
Email: davideyi.hu@polimi.it
Date: 17 December 2024
Exercise Lecture: 10 - Concurrency and Django

POLITECNICO
MILANO 1863

Concurrency

Exercise 1 - Threads

```
sources = ["url1", "url2", "url3", "url4"]

def fetch_data(source):
    print(f"Starting fetch from {source}")
    time.sleep(random.uniform(0.5, 1.0))  # Simulate I/O delay
    data = f"data_from_{source}"
    print(f"Finished fetch from {source}")
    return data

for source in sources:
    fetch_data(source)
```

Exercise 2 - Processes

```
sources = ["url1", "url2", "url3", "url4"]

def fetch_data(source):
    print(f"Starting fetch from {source}")
    time.sleep(random.uniform(0.5, 1.0))  # Simulate I/O delay
    data = f"data_from_{source}"
    print(f"Finished fetch from {source}")
    return data

for source in sources:
    fetch_data(source)
```

Exercise 3 - Search for an element

```
import time
import random

def generate_large_array(size, search_element):
   arr = [random.randint(0, 1_000_000) for _ in range(size)]
   if search_element not in arr:
       arr[random.randint(0, size - 1)] = search_element
   return arr

def sequential_search(arr, target):
   start = time.time()
   for i, value in enumerate(arr):
       if value == target:
           end = time.time()
           print(f"Element {target} found at index {i}")
           print(f"Search took {end - start:.2f} seconds")
           return i
   end = time.time()
   print(f"Element {target} not found")
   print(f"Search took {end - start:.2f} seconds")
   return -1
```

Exercise 4 - Maximum of a list

```python
import random
import time

def generate_large_array(size):
    return [random.randint(0, 1_000_000) for _ in range(size)]

def find_max_sequential(arr):
    start = time.time()
    max_value = max(arr)  # Sequentially find the maximum value
    end = time.time()
    print(f"Sequential max value: {max_value}")
    print(f"Time taken (sequential): {end - start:.2f} seconds")
    return max_value
```

Django