Session 2

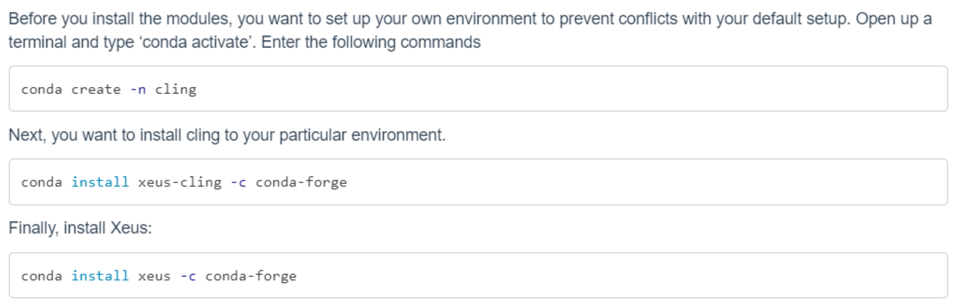
Task 1

Python library that hide the password while typing it

```
import getpass

password= print("please enter your password: ")

password=getpass.getpass()
```

Task 2

Package in jupyter to allow using C++



Before you install the modules, you want to set up your own environment to prevent conflicts with your default setup. Open up a terminal and type ‘conda activate’. Enter the following commands

```
conda create -n cling
```

Next, you want to install cling to your particular environment.

```
conda install xeus-cling -c conda-forge
```

Finally, install Xeus:

```
conda install xeus -c conda-forge
```

Task 3

Code that have the same performance of do while

```
secret_word = "python"

counter = 0


while True:

   word = input("Enter the secret word: ").lower()

   counter = counter + 1

   if word == secret_word:

      break

   if word != secret_word and counter > 7:

      break
```

Task 4

A python code that pass by power 2 every loop

```
for x in (2**p for p in range(10)):   print(x)
```

result:

```
1
2
4
8
16
32
64
128
256
512
```

Task 5

How to create infinite loop using for loop

```
from itertools import *

a= [100]

for i in cycle(a):

        print(i)
```

Task 6

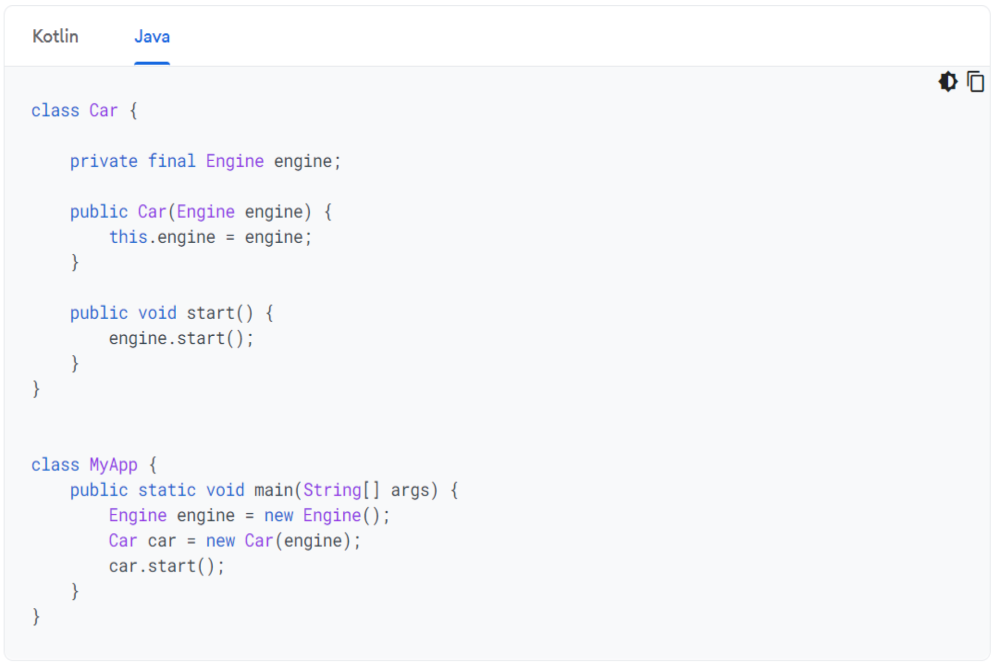
What is dependency injection?

is a technique in which one object supplies the dependencies of another object.

A **dependency** is an object that can be used in the class. It can be a Network service, Database service, Location service

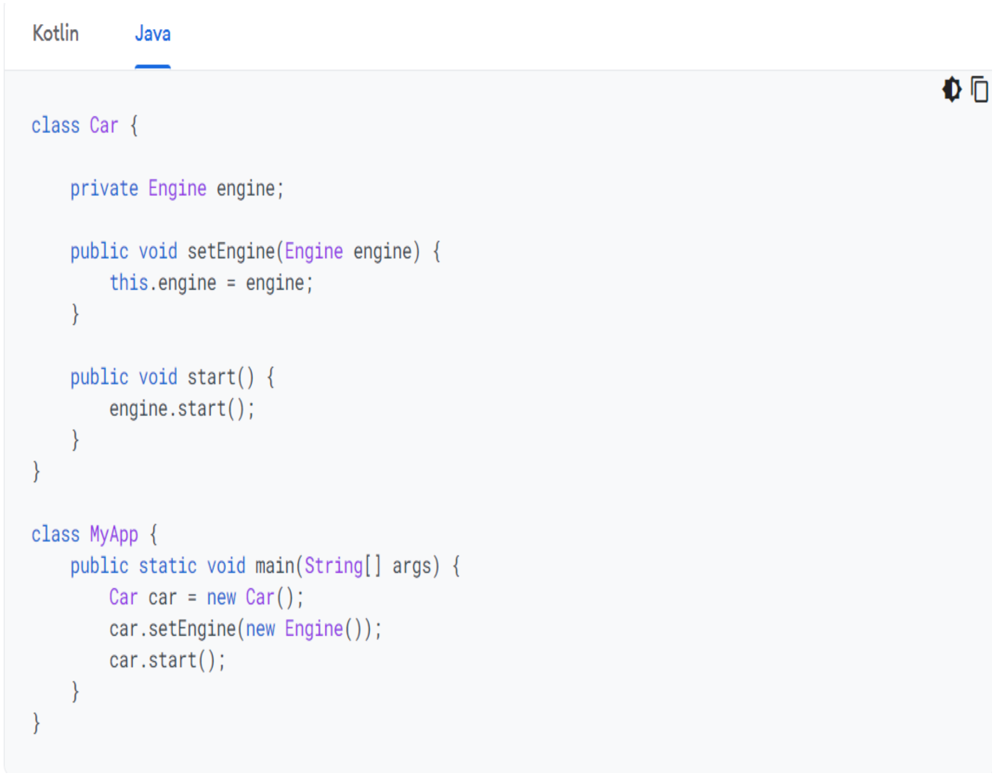
There are two major ways to do dependency injection:

- **Constructor Injection** You pass the dependencies of a class to its constructor.



```java
class Car {

    private final Engine engine;

    public Car(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        engine.start();
    }
}


class MyApp {
    public static void main(String[] args) {
        Engine engine = new Engine();
        Car car = new Car(engine);
        car.start();
    }
}
```

- **Field Injection (or Setter Injection)**. Certain classes are instantiated by the system, so constructor injection is not possible. With field injection, dependencies are instantiated after the class is created.



```java
class Car {

    private Engine engine;

    public void setEngine(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        engine.start();
    }
}

class MyApp {
    public static void main(String[] args) {
        Car car = new Car();
        car.setEngine(new Engine());
        car.start();
    }
}
```

Task 7

What are the clean code rules?

- **Design rules**

1. Keep configurable data at high levels.
2. Prefer polymorphism to if/else or switch/case.
3. Separate multi-threading code.
4. Prevent over-configurability.
5. Use dependency injection.
6. Follow Law of Demeter. A class should know only its direct dependencies

- **Names rules**

1. Choose descriptive and unambiguous names.
2. Make meaningful distinction.
3. Use pronounceable names.
4. Use searchable names.
5. Replace magic numbers with named constants.
6. Avoid encodings. Don't append prefixes or type information.

- **Functions rules**

1. Small.
2. Do one thing.
3. Use descriptive names.
4. Prefer fewer arguments.
5. Have no side effects.
6. Don't use flag arguments. Split method into several independent methods that can be called from the client without the flag.

Comments rules

1. Always try to explain yourself in code.
2. Don't be redundant.
3. Don't add obvious noise.
4. Don't use closing brace comments.
5. Don't comment out code. Just remove.

6. Use as explanation of intent.
7. Use as clarification of code.
8. Use as warning of consequences.