**OOP- dry part**

**Basics**

Question 1

Read about the following terms and explain them to your instructor:

a) Polymorphism

b) OOP

c) Functional Programming

Answer 1

1. Polymorphism is a programming idea in OOP where instead of having different names for the same act done in different entities (classes for example), the same terminology is used to describe the act across entities.

Example:

Class Animal is a parent class, and each of its children has a “make sound” method. Instead of using “bark” for dog and “meow” for cat, the multiple use of “make sound” uses the polymorphism idea to tolerate different entities (animals) with the same act (making a sound).

1. OOP- Object Oriented Programming. A programming technique that relies on the usage of objects and classes. Objects are classified under classes and have attributes and methods. This method is helpful for real-world modeling, where entities and types of entities exist.
2. Functional programming is a different type of programing technique, which focuses mainly on the solution world and less on the structure and relations between entities. Code that is written with a functional programming technique contains functions that usually relate to each other, can get other functions as input, and usually do not create side-effects and effects only output (immutable and pure).

Question 2

Read about the following topics and explain them to your instructor:

a) Composition VS Inheritance.

i) What are they?

ii) What are the pros/cons of each?

b) Mutability VS Immutability.

i) What are they?

ii) What are the pros/cons of each?

c) Static Typing VS Dynamic Typing in programming languages

i) What are they?

ii) What are the pros/cons of each?

iii) Give examples of 2 programming languages of every type

Answer 2

1. i) Composition and inheritance both are methods to let an object from a certain class access attributes and/or methods from a different class. Composition is the case where an object uses an entire object as an attribute. For example, a class “car” uses a class “engine” as an attribute. Thus the engine is replaceable and serves a role in the sentence “The car **has an** engine”). In inheritance, the object inherits an entire class (and its attributes and methods). For example, a class “dog“ inherits the class “animal”. A dog **is an** animal, and thus inherits all of the animal’s attributes and methods, and in addition can have another, or override existing ones.

ii) Inheritance: Pros are that it is possible to build hierarchies with that method, where a change of parent class affects children directly. This lets us use polymorphism. Cons are that a single error in the parents-children chain can break the whole objects. A child can only have a single parent.

Composition: Pros are that multiple classes can be used. This is flexible since the objects if the class can be modified. Errors in a used class does not error the class. Cons are that there are less direct hierarchies, with weaker polymorphism.

1. i) Mutability is the ability of a subject to change its attributes in-place. For example an integer value cannot be modified after creation, only replaced (immutable), while a list can be modified after creation with appending a value (mutable).

ii) Pros of mutability: In nature some variables need to be changed over time. Cheaper for memory allocation. Cons: side effects and unintended changes to variables, might be problematic in runs that contain parallelism. Cannot be used as dictionary keys.

1. i) Static typing refers to variables that their types are set before compilation and cannot be changed afterwards. Dynamic typing refers to variables that their types are changeable during runtime.

ii) Static typing is better for runtime and error handling, since types are checked at compilation time (and not during runtime). Dynamic typing is obviously more dynamic and gives the programmer more freedom.

iii) Static typing- Java and C

Dynamic typing- Python and JavaScript.

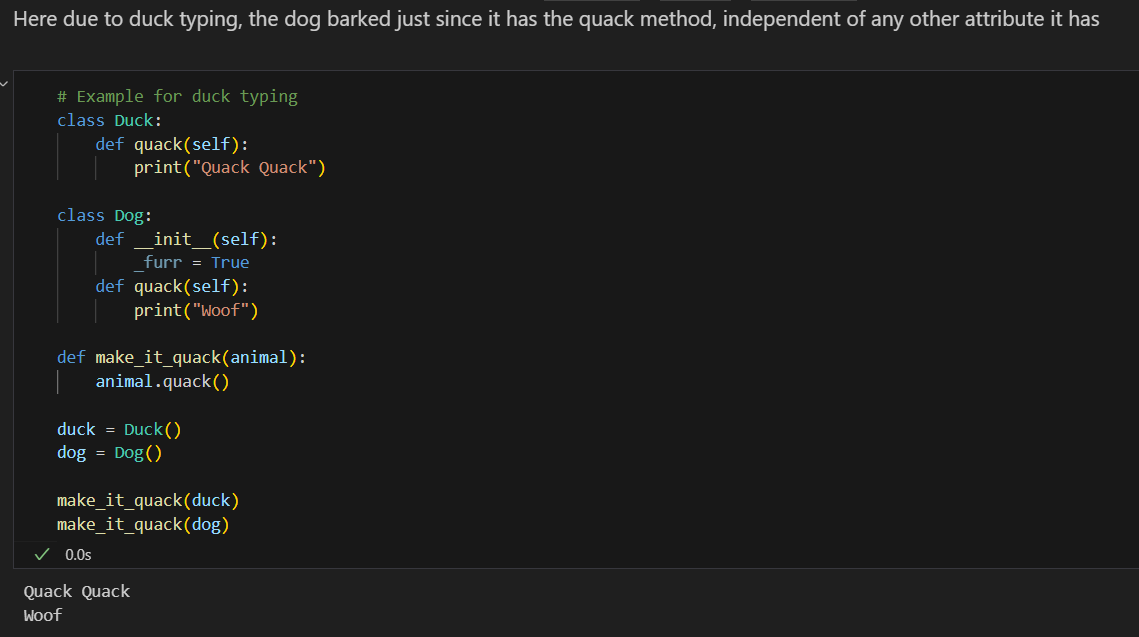
Question 3

What’s duck typing? Show a basic code example in python demonstrating this concept

Answer 3

Duck typing in dynamically typed languages practically means that what really matters is what the subject does and not really what is its type. “If it walks like a duck and quacks like a duck- it’s a duck”. From Wikipedia: “With duck typing, an object is of a given type if it has all methods and properties required by that type”.

Example:



Question 4

What’s the difference between the various accessibility levels of fields/methods (private/protected/public)? How does it work in python?

Answer 4

In programming languages like C and Java there is a way to strictly enforce the accessibility of fields and methods- where can they be accessed - only inside the class (private), only inside the class and its subclasses (protected), or accessible everywhere (public).

In python, there isn’t a way to strictly enforce accessibility, though there are conventions of variable naming:

Public: : can be accessed everywhere

Protected: : can actually still be accessed everywhere but not recommended.

Private: : cannot be accessed everywhere as is, uses name changing at compilation time (to ) to enable private access.

Question 5

What’s the difference between instance/class/static methods in python? What’s the use case of each type?

Answer 5

These are the 3 kinds of methods possible inside a class.

are the regular methods- their first argument is and they can access both instance and class attributes.

are methods that can access only class attributes (rather than instance attributes). They are declared with and their first argument is .

are methods that do not access class or instance attributes, and are inside the class since they relate to it logically. Thy are declared with and do not get as input.

**Design Concepts**

Question 1

Read about SOLID design principles. Explain them to your instructor.

Answer 1

SOLID principles are 5 OOP design principles for better OOP.

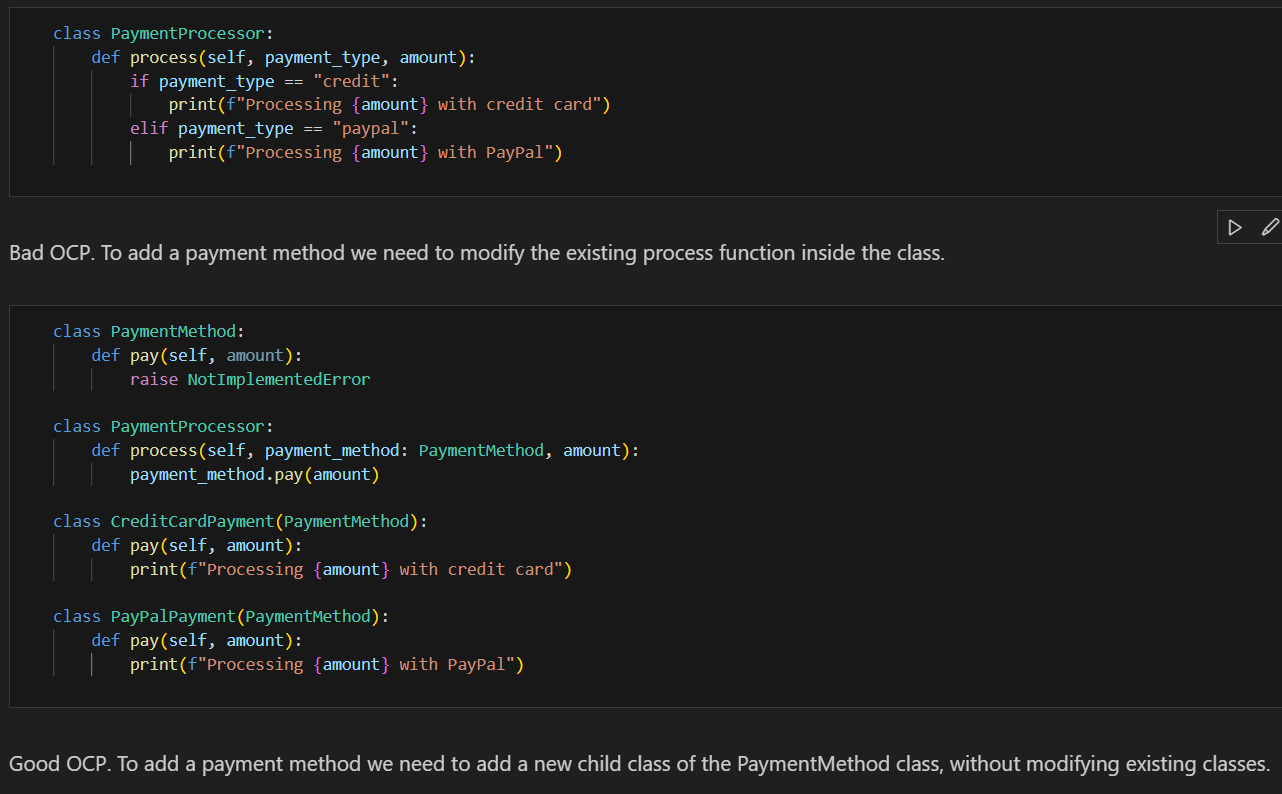
**S**RP-

Single Responsibility Principle. Means each class should have only one responsibility- only one reason to change. Example:



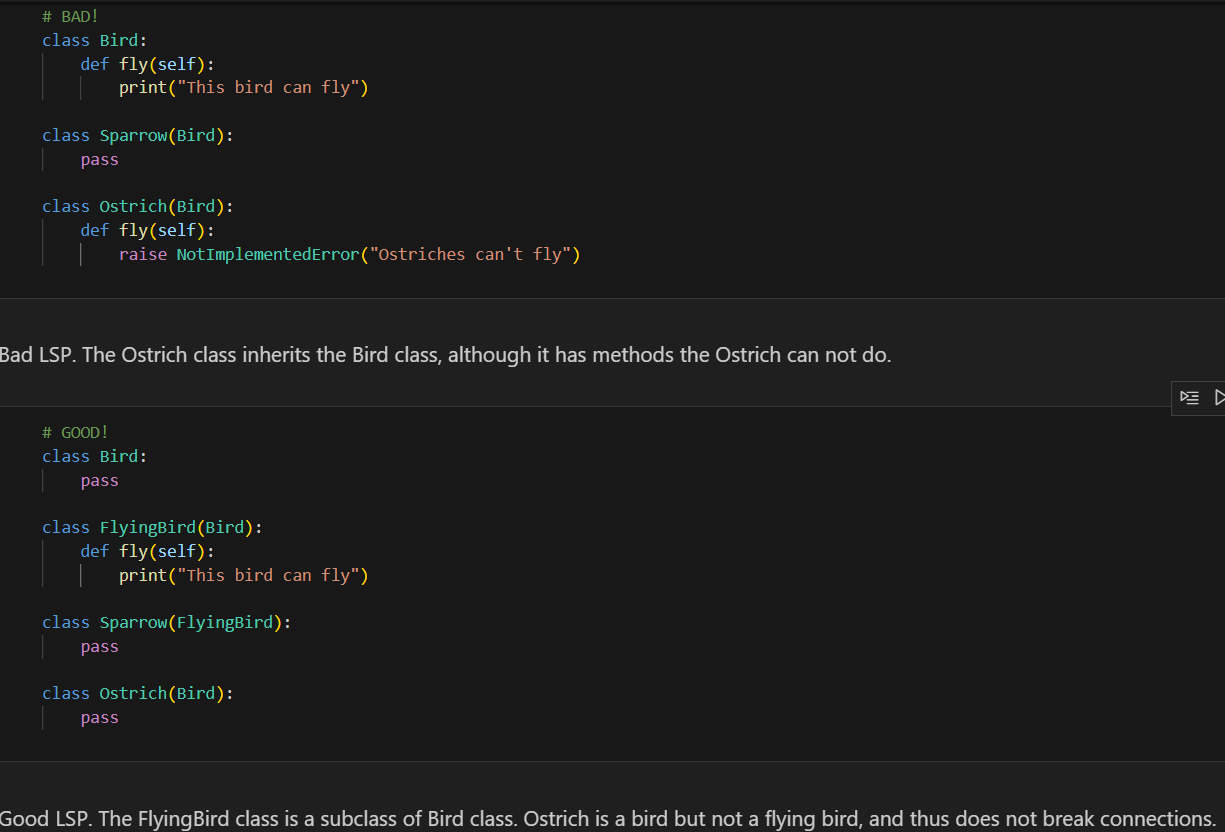
**O**CP-

Open/closed principle. Software entities (classes, modules, functions) should be open for extension but closed for modification.

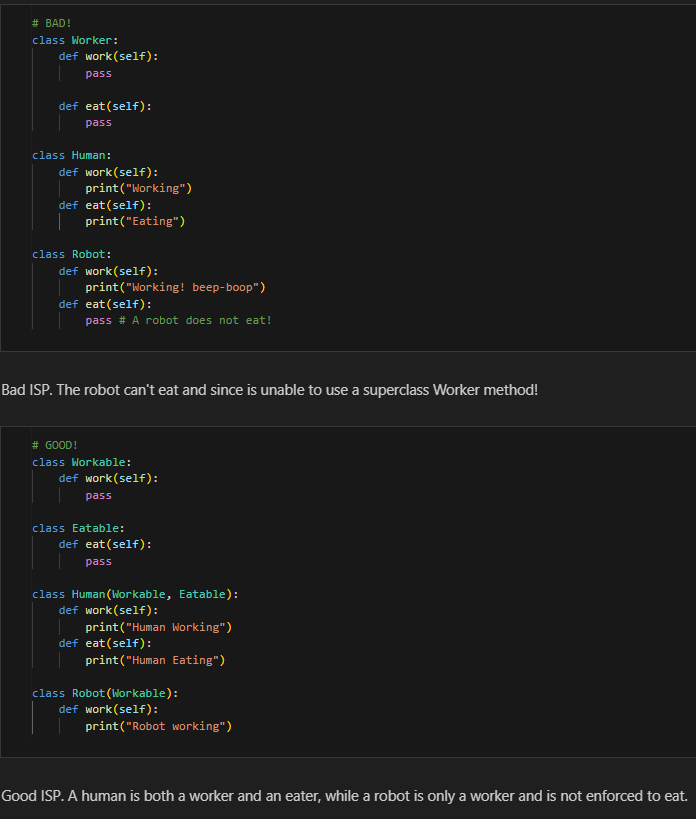


**L**SP –

Liskov Substitution Principle. Objects of a subclass should be replaceable with objects of their superclass without breaking the program’s behavior.



**I**SP –

Interface Separation Principle. A subclass should not be required to implement a method it does not need.

**Did not do D!**

Question 2

What’s the advantage of having multiple constructors? In python that feature is not supported unfortunately. How can you overcome that?

Answer 2

Having multiple constructors is helpful for better flexibility in creation of instances with different argument set / input types. This can be overcome in python by creating a class method that returns a new instance and gets input as chosen.

Question 3