Ask me anything

0 questions
0 upvotes
Should a Square inherit from Rectangle or vice versa? Why?

- square from rectangle, because it's a specialization?
- square should inherit from rectangle because it is a special case of a rectangle
- depending on the usage, usually shouldn't because they should inherit from shape

According to Subclassing, rectangle inherit from square, in is_a, a square inherit from rectangle

because rectangles may have further attributes and methods
<table>
<thead>
<tr>
<th>Subtyping in a way is a conceptual specialization</th>
<th>Subtyping: substitutability; specialization: says that one thing is a special case of another</th>
<th>Subtype can substitute a type; specialization is an extension of something else</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtyping can also be used to just ensure compatibility of multiple(!) different types, whereas is-a vertical chain of is-a relations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Can a dynamically-typed language be polymorphic, or only a statically-typed one?

A dynamically typed language is always polymorphic, because the variables accept objects of various (all) types.

I think one is independent of other. At least if you consider ad-hoc polymorphism.
How is an object like a closure?

- As long as attributes are private, they are hidden to the outside, just like in closures.
- Both give a way to hide (make private) certain parts.
- Both are environments that bind free names within it.
- Closure gives you access to an outer function’s scope from an inner function.
- It is a variable namespace that may overwrite variables with greater scope, thus why in C++ for example "this" exists. Also self in Python is explicitly the closure.
Should an “intersect” methods for a Shape class and subclasses be covariant in its arguments? Contravariant? Neither?

What does intersect do?

Probably contravariant, because if it was covariant, you could only intersect two shapes of the same type.

The issue is that you can create new shape by intersecting known shape

The output is a new shape so it should be invariant.

Neither because the initial set is smaller than the final set

Neither, as the return type is not known and probably a complex shape
Can a dynamically typed language support overloading? Why or why not?

Yes.

It could by having no strict types for the function signature and then do type checking of the parameters it got and call different methods.

It is independent again so yes.

Yes, haskell does it for example (pattern matching).

no - if i overload methods i have to say what static type they have - but there is no static typing here.

Python does it with type hints.

Yes, even if it's dynamically typed, it still can have different types, that require the same overloaded method.
In what way does a class represent a family of types?

- Because a class is a generator of type
- That a class can have subclasses which again represent a subtype of the type of the original (super) class
Last chance for questions