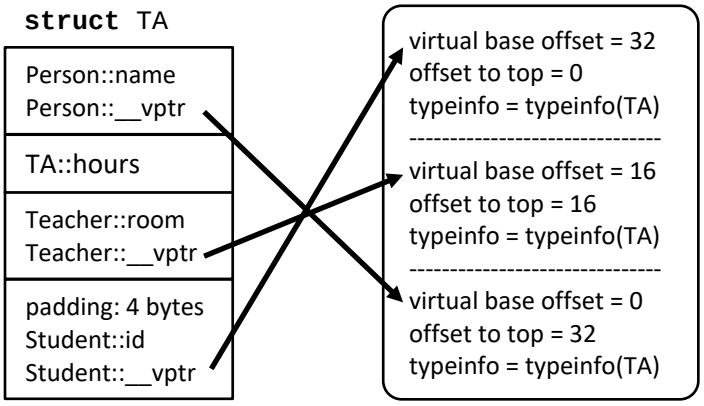
Off-side rule: blocks are expressed by indentation (Python)
Scope: region of code where name binding is valid
Block scope: name is valid within the block it is declared in
Function scope: local variable valid until end of function
`global`: Allows reassignment of global variable inside a function. Otherwise by default a local variable is created instead.
Sequence: an ordered collection of values (repetition allowed). Python strings and tuples are immutable. Changing → copying.
`strip(), endswith(s), s.join(lst), split(c)`
Python lists are mutable – can be updated.
`insert(v), remove(v), count(v), append(v), pop(idx), extend(lst), index(v), sort()`
`map(P(x), Iter) == [P(x) for x in Iter]`
Alias: different names referring to same memory location
Dictionary: collection of key value pairs. Key must be unique and hashable (immutable object with no reference to mutables).
`update(d), {K(x):V(x) for x in Iter if P(x)}`
Memoisation: top-down approach, uses recursion
Dynamic programming: bottom up-approach, uses iteration
Pure Function: output only determined by input, no side effects
Referential transparency: Can substitute expression with value
Prototype-based Programming: inheritance occurs by copying existing objects (a prototype) and adding fields/methods to it. Add `__slots__` to prevent instance from adding attributes.
Ad-hoc Polymorphism: function or operator overloading.
Subtyping: interface inheritance, forms "is-a" relationship
Runtime Polymorphism: choose behaviour through one interface
Dynamic Dispatch: determines which implementation to call at runtime based on context (caller's type and maybe input type).
Single Dispatch: Context is the instance type of the caller.
Multiple Dispatch: Context also includes type of parameter.
Late Binding: Associates name with operation at runtime.
Virtual Base Class and Virtual Tables for Derived Class:

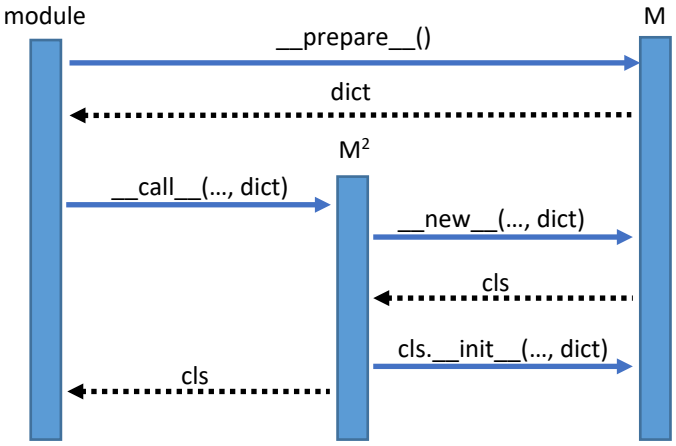


Fragile Base Class: in implementation inheritance, changing base class can silently break derived class due to dependency.
Composition: instead of subclassing, make it a field (has-a).
Forwarding: forward method to composed class.
Delegation: Automatic forwarding of method to composed class.
Mixin: Implementation inheritance not expected to be used as a super class, but rather provides functionality to other classes.
Dependency Inversion Principle: 1) modules should depend on each other's interfaces, 2) interface should not depend on implementation, 3) can swap out module and maintain behavior.
Method Resolution Order: the order attributes are looked up.
Local Precedence Order: the order parent classes are inherited.
Monotonicity: MRO for a class should always be the same.

C3 Linearization: L[C] = (C, merge(L[B_1], …, L[B_N], B_1 … B_N)
Lvalue: value stored in memory, has an address.
Rvalue: temporary value, may not be in memory.
Rvalue Reference: reference to temporary object to enable move semantic (instead of doing a deep copy, which can be expensive).
Copy Elision: compiler optimization to avoid copying of objects.
Return Value Optimization: building return value at final location.
Generic Programming: writing program with minimal assumption about the structure and/or type of data.
Parametric Polymorphism: ability to handle values without depending on their types (e.g. join lists of the same element type)
Overload Resolution: (if two candidates are equally suitable)

1. Non-template function overload
2. Template specialization
3. More specific and specialized template
4. Base template

Introspection: Ability to examine compiler internal knowledge.
Reflection: Ability for a process to introspect and change itself.
Reification: turns abstract representation to concrete objects.
Type Erasure: removal of type information/check at runtime.
Descriptor: class which customizes attribute access of another object (through `__get__`, `__set__`, and `__delete__`)
Decorator: augment existing function or class (returns callable)
Functor: A class that implements `__call__` (a function object)
Closure: inner function that is returned by other function, and it uses outer variables (retain value at time of closure definition)
Class Creation: M=Metaclass, M^2=Metametaclass (usually `type`)



Coercion: implicit type conversion
Type punning: changes type but not in-memory representation.
Strict Aliasing: pointer of different types assumed to not alias.
Type checking: process of verifying and enforcing type safety.
Nominal Typing: variable have same type if types are same name
Duck Typing: suitability based only on presence of attribute.
Structural Typing: suitable if they have same structure/interface.
Type Inference: can infer type information from variable usage.
Enum Class: disallows coercion to integer and other enums.
Union: stores different data types at same memory location.
Tagged Union: union managed by a tag to say which field is in use
Covariance: relationship allows use of more derived type.
Contravariance: allows use of more generic type than specified.
Contract Programming: support for specifying precondition, postcondition, errors, and invariants of functions and structures.
Metaprogramming: writing code that will generate more code.
Macro Systems: maps input sequence into replacement output.