import_module(name):
    `sys.modules[name]`*? return it*
    submodules:
        load parent module
        `sys.modules[name]`*? return it*
    `spec = find_spec(name, path)`
    `load(spec)`
    `module = sys.modules[name]`
    submodules:
        set `module` *as attribute of* `parent`
    return `module`

find_spec(name, path):
    `for finder in sys.meta_path:`
        call its `find_spec`*(name, path)*
        return spec if successful

PathFinder.find_spec(name, path):
    `for directory in path:`
        get `sys.path_hooks` *entry*
        call its `find_spec`*(name)*
        return spec if successful

load(spec):
    module = `spec.loader.create_module(spec)`
    if module is None:
        module = `types.ModuleType(spec.name)`
    set initial module attributes
    `sys.modules[spec.name]` *= module*
    `spec.loader.exec_module`*(module, spec)*

get_code(module, spec):
    if `spec.cached` *exists, and matches* `origin` *stats,*
        return it!
    load source from `origin` *and compile it*
    write bytecode to `spec.cached`