```julia
"""
    struct ComplexSpace <: EuclideanSpace{ℂ}

A standard complex vector space ℂ^d with Euclidean inner product and no additional
structure. It is completely characterised by its dimension and whether its the
normal space
or its dual (which is canonically isomorphic to the conjugate space).
"""
struct ComplexSpace <: EuclideanSpace{ℂ}
  d::Int
  dual::Bool
end
ComplexSpace(d::Integer = 0; dual = false) = ComplexSpace(Int(d), dual)
function ComplexSpace(dim::Pair; dual = false)
    if dim.first === Trivial()
        return ComplexSpace(dim.second; dual = dual)
    else
        msg = "$(dim) is not a valid dimension for ComplexSpace"
        throw(SectorMismatch(msg))
    end
end
ComplexSpace(dims::AbstractDict; dual = false) = ComplexSpace(dims...; dual = dual)

# convenience constructor
Base.:^(::ComplexNumbers, d::Int) = ComplexSpace(d)
Base.getindex(::ComplexNumbers) = ComplexSpace
Base.getindex(::ComplexNumbers, d::Int) = ComplexSpace(d)

# Corresponding methods:
#------------------------
dim(V::ComplexSpace) = V.d
isdual(V::ComplexSpace) = V.dual
Base.axes(V::ComplexSpace) = Base.OneTo(dim(V))

Base.conj(V::ComplexSpace) = ComplexSpace(dim(V), !isdual(V))

Base.oneunit(::Type{ComplexSpace}) = ComplexSpace(1)
⊕(V1::ComplexSpace, V2::ComplexSpace) = isdual(V1) == isdual(V2) ?
    ComplexSpace(dim(V1)+dim(V2), isdual(V1)) :
    throw(SpaceMismatch("Direct sum of a vector space and its dual does not
exist"))
fuse(V1::ComplexSpace, V2::ComplexSpace) = ComplexSpace(V1.d*V2.d)
flip(V::ComplexSpace) = dual(V)

infinum(V1::ComplexSpace, V2::ComplexSpace) = isdual(V1) == isdual(V2) ?
    ComplexSpace(min(dim(V1), dim(V2)), isdual(V1)) :
    throw(SpaceMismatch("Infinum of space and dual space does not exist"))
supremum(V1::ComplexSpace, V2::ComplexSpace) = isdual(V1) == isdual(V2) ?
    ComplexSpace(max(dim(V1), dim(V2)), isdual(V1)) :
    throw(SpaceMismatch("Supremum of space and dual space does not exist"))

Base.show(io::IO, V::ComplexSpace) = print(io, isdual(V) ? "(ℂ^$(V.d))'" :
"ℂ^$(V.d)")
Base.show(io::IO, ::Type{ComplexSpace}) = print(io, "ComplexSpace")
```