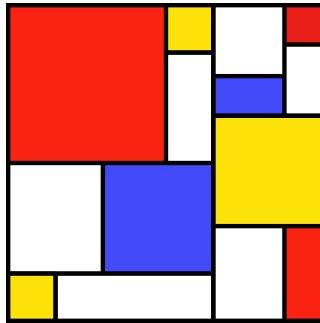


# SLiM

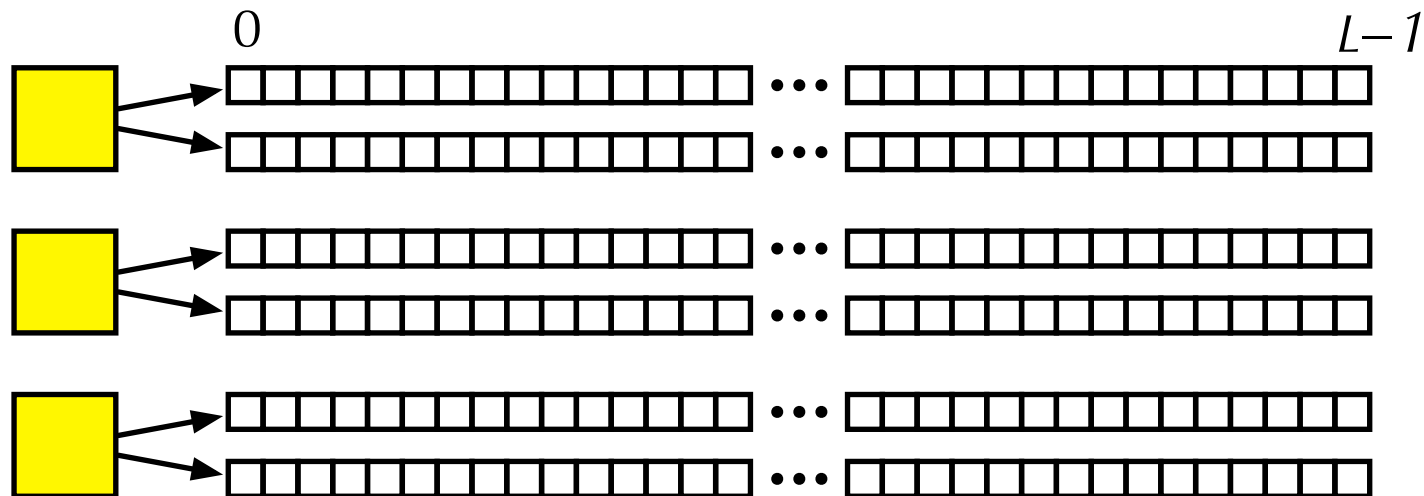
## Workshop Series



### #3: The Population Hierarchy

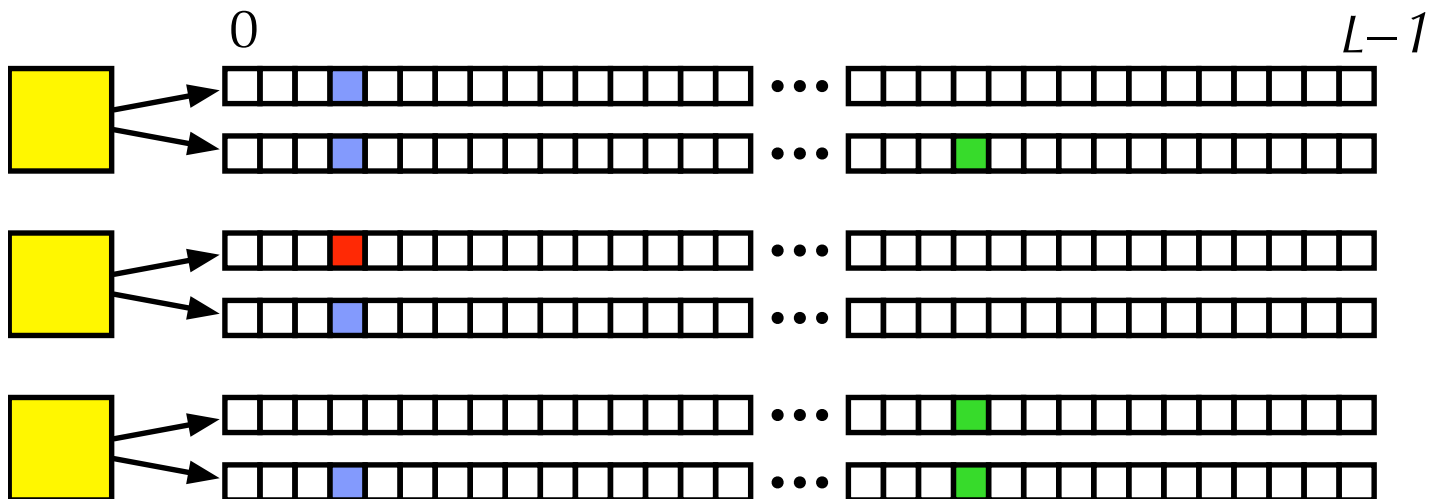
# Individuals and genomes

- Individuals (class `Individual`) are organisms
- Individuals are born, mate, die, ...
- Each individual has two genomes (class `Genome`)
- Each genome has  $L$  discrete base positions



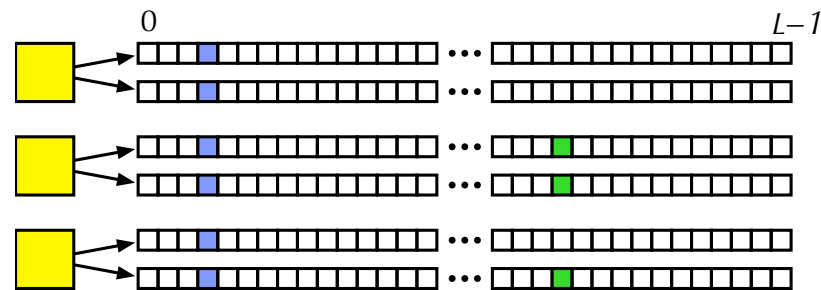
# Mutations

- Mutations (class `Mutation`) live in genomes
- Genomes begin empty (the *ancestral* state)
- Mutations represent a *non-ancestral* allele (SNP)
- Mutations have properties: selection coefficient

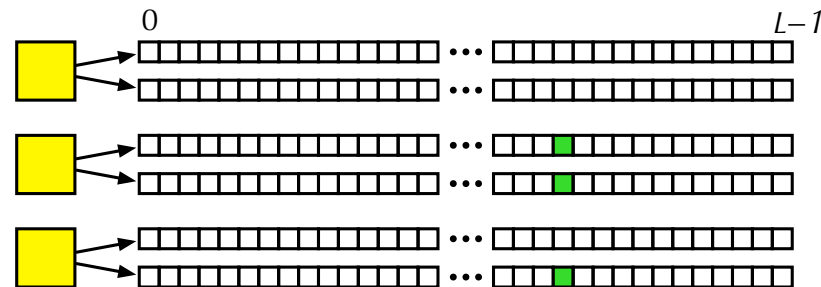


# Substitutions

- Substitutions (class `Substitution`) are fixed
- Fixed mutations become substitutions by default

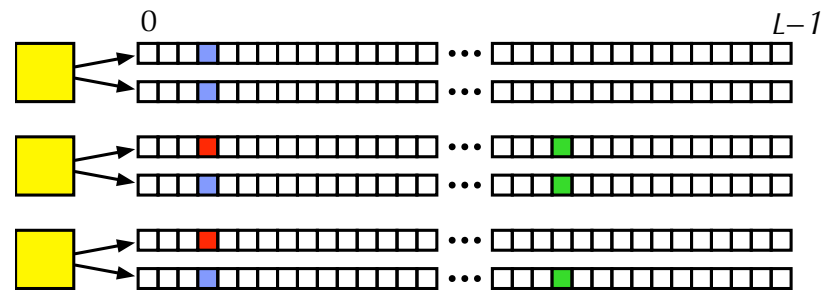


↓ substitution



# Substitutions

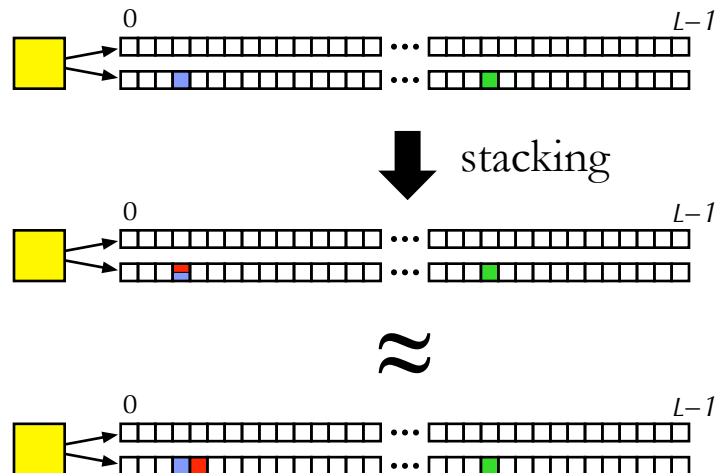
- Every mutational lineage is distinct!
- Even with identical selection coefficients, etc.



no substitution

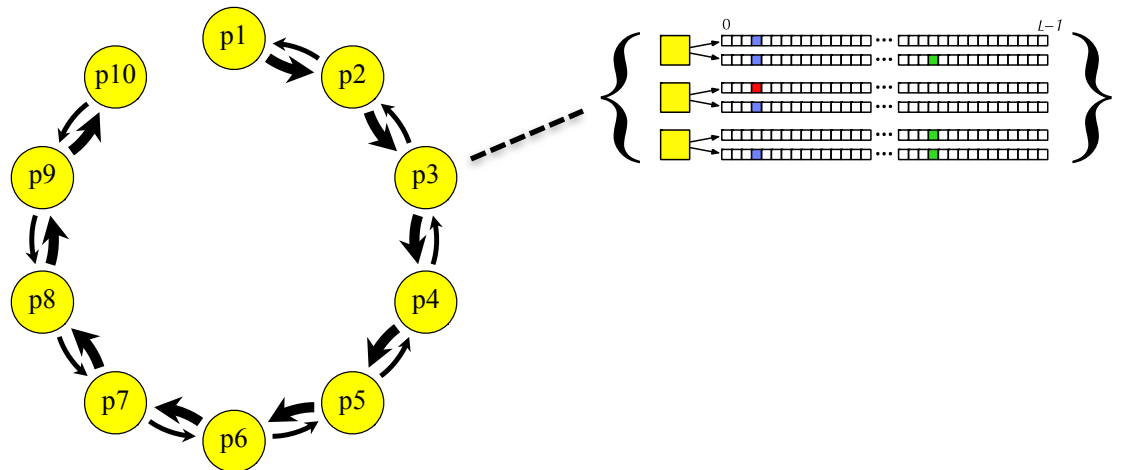
# Mutation stacking

- If mutation A exists in a genome at position  $P$ ...
- And then mutation B occurs at  $P$ ...
- By default, *both* are kept; they “stack”
- Stacking policy can be modified, however



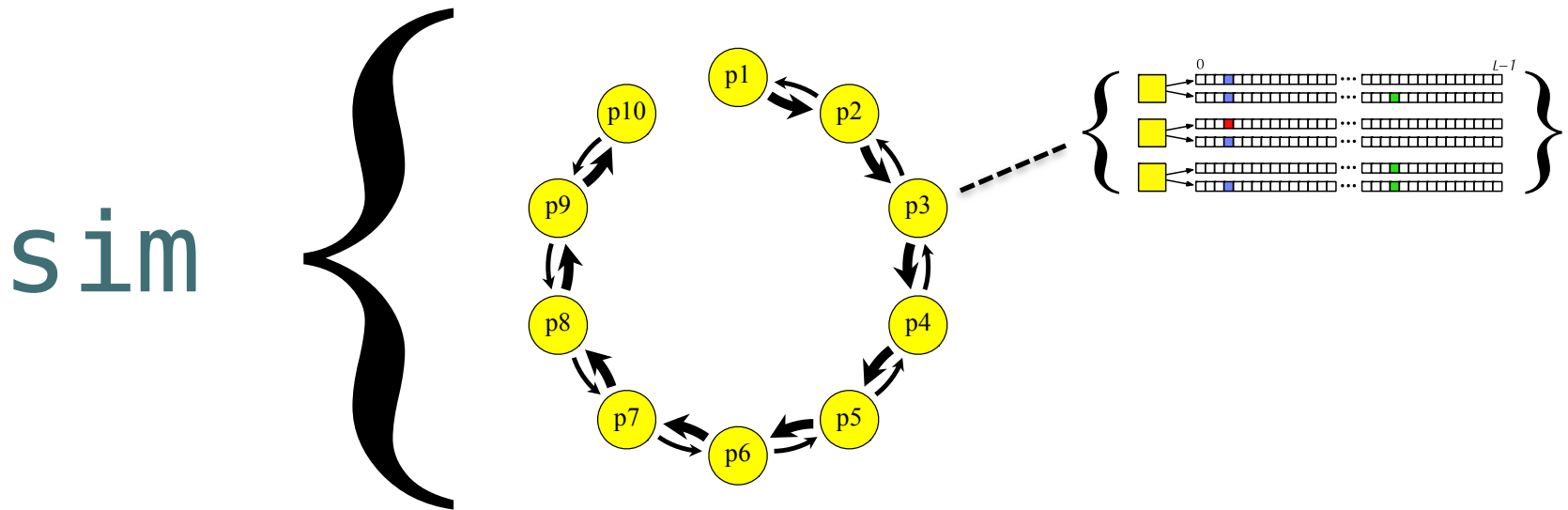
# Subpopulations and migration

- Subpops (Subpopulation) contain individuals
- Any number of subpopulations can exist
- They can be connected by migration in any way
- Population structure can change over time



# The population

- The population contains subpopulations
- Only one population can exist in a model
- It is managed by the simulation object
- The simulation object is a global, `sim` (SLiMSim)





## SLiM reference sheet

[illegible][illegible][illegible][illegible]

```

MutationType (MuType):
color <= (i3)
colorSubstitution <== (i3)
convertSubstitution <== (i3)
distributionPrims <= (f)
distributionType <= (i3)
dominance <== (f)
id <= (i3)
mutatorInitsGroup <== (i3)
mutatorInitsPolicy <== (i3)
tag <= (i3)

= (i3)setValue(i3 key)
= (void)setDistribution(i3 distType, ...)
= (void)setValue(i3 key, value)

GenomicElement (GElement):
endPosition <= (i3)
genomicElementType <= (i3)GenType()
starPosition <= (i3)
tag <= (i3)

= (void)setGenomicElementType(i3)GenType() genomicElementType()

GenomicElementType (GType):
color <= (i3)
id <= (i3)
mutatorInitsPrims <= (f)
mutationType <= (i3)MuType()
tag <= (i3)

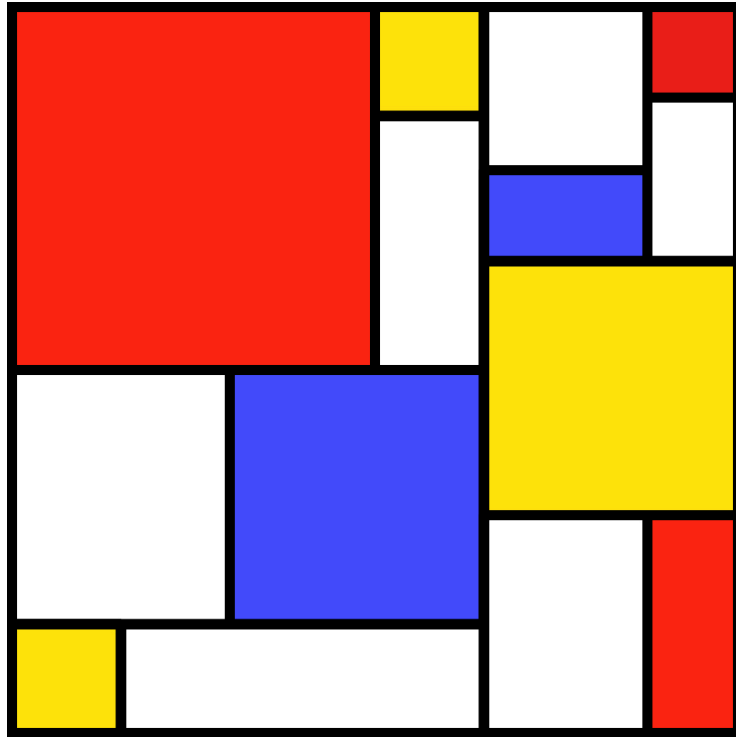
= (i3)setValue(i3 key)
= (void)setMutationInits(i3)MuType() mutationType, n proportions)
= (void)setValue(i3 key, value)

SLIMDistance (SLBlock):
action <== (i3)
end <= (i3)
id <= (i3)
source <= (i3)
start <= (i3)
tag <= (i3)
type <= (i3)

InteractionType (IntType):
id <= (i3)
mutability <== (f)
reciprocal <== (i3)
recognitionType <= (i3)
spatially <== (i3)
tag <= (i3)

= (f)distance(i3)Ind individual1, (NoInd-)individual2))
= (void)distancePoint(i3)Ind individual1, f point)
= (i3)Ind-IndAdjStrength(i3)Ind individual1, i3 count)
= (void)distanceLabel(NoSGroup <sup>sub>, i3 individual)
= (i3)setValue(i3 key, value)
= (i3)interact(NeighborCount(i3)Ind individual1, (NoInd-)individual2))
= (i3)interaction(i3)Ind individual1, (NoInd-)individual2))
= (i3)Ind-nearestInteractionNeighbor(i3)Ind individual1, i3 count)
= (i3)Ind-nearestNeighbor(i3)Ind individual1, i3 count)
= (i3)Ind-nearestInteractionNeighbor(SGroup <sup>sub>, f point, i3 count)
= (i3)InteractInteraction(i3)Ind functionType, ...)
= (void)setValue(i3 key, value)
= (f)strength(i3)Ind receiver, (NoInd-)exerters)
= (i3)totalNeighborStrengths(i3)Ind individual)
= (void)setValue(i3 key, value)

```



### SLiM Workshop Exercise #3