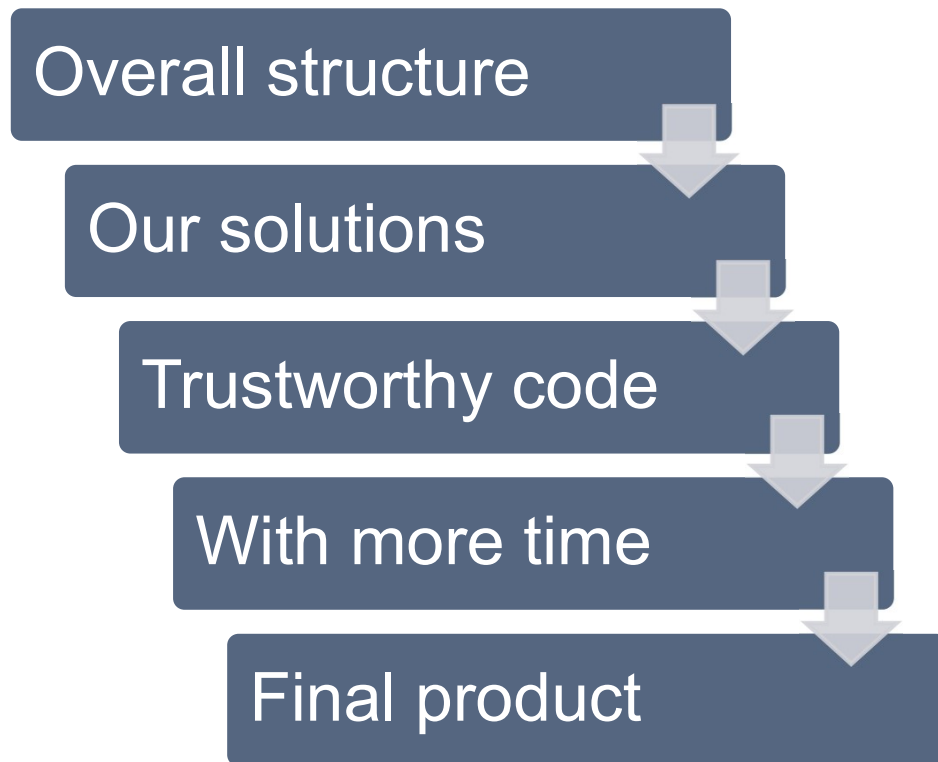


Rossumøya Island

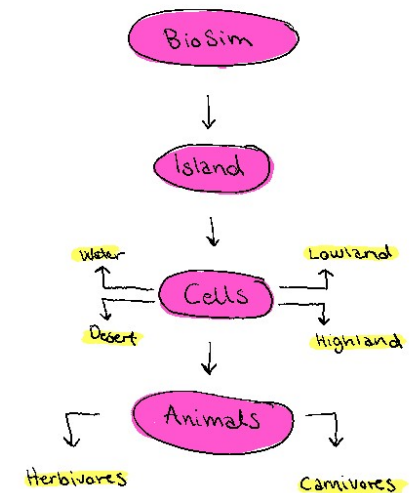
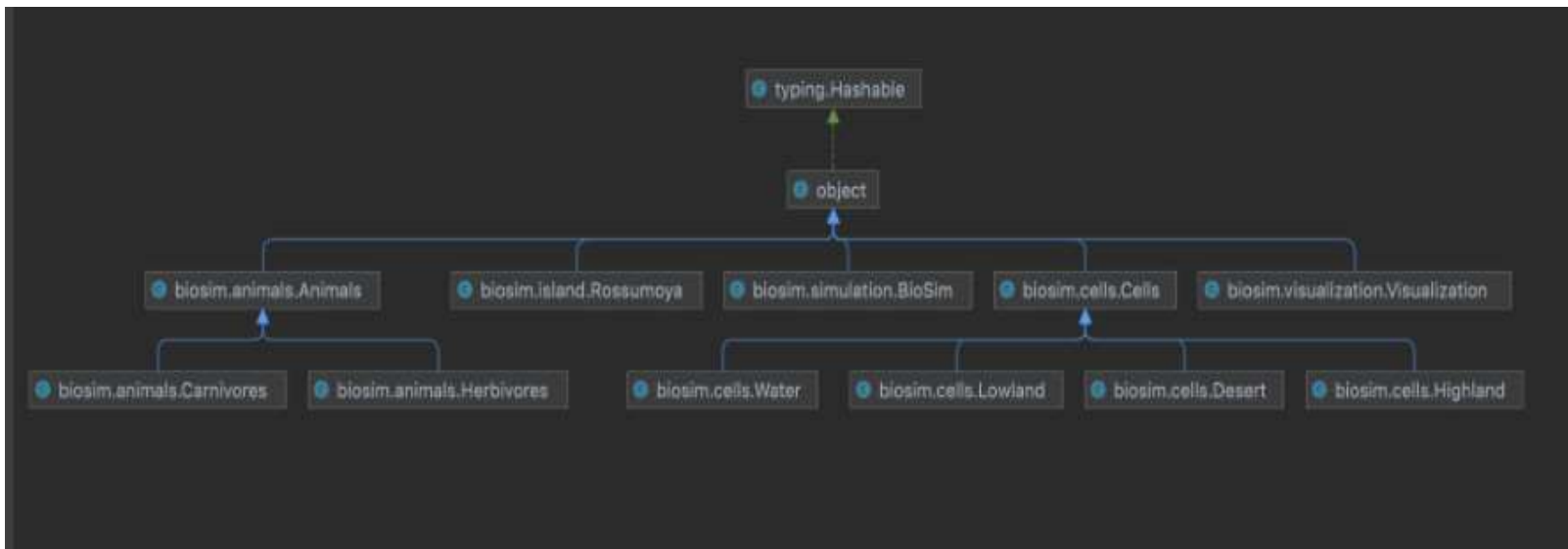
Vilde Riiber Dale & Nada Salah Mahammed

21.01.22

TABLE OF CONTENTS



OVERALL STRUCTURE



```

def animal_aging(self):
    """
    """
    self.age += 1

def calc_birth_weight(self):
    """Calculates weight of an animal at birth..."""
    birth_weight = random.gauss(self.parameters['w_birth'], self.parameters['sigma_birth'])
    return birth_weight

def weight_gain(self, food):
    """When eating, the animal gains weight..."""
    self.weight += self.parameters["beta"] * food
    self.calc_fitness()

def weight_loss(self):
    """Each cycle the animal lose weight..."""
    self.weight -= (self.parameters['eta'] * self.weight)
    self.calc_fitness()

def calc_fitness(self):
    """Fitness is the overall health of the animal..."""

    q_age = 1 / (1 + math.exp(self.parameters['phi_age'] * (self.age - self.parameters['a_half'])))
    q_weight = 1 / (1 + math.exp(-self.parameters['phi_weight'] * (self.weight - self.parameters['w_half'])))

    if self.weight <= 0:
        self.fitness = 0
    else:
        self.fitness = q_age * q_weight

```

OUR SOLUTIONS

TRUSTWORTHY CODE

- Safe code = well tested code
- Our projects biggest flaw



```

mono_hc.py not covered
mono_ho.py not covered
sample_sim.py not covered
✓ src 85% files, 88% lines covered
  ✓ biosim 100% files, 88% lines covered
    __init__.py 100% lines covered
    animals.py 96% lines covered
    cells.py 88% lines covered
    island.py 82% lines covered
    simulation.py 68% lines covered
    visualization.py 98% lines covered
  > biosim.egg-info

```

With the given biosim_interface test we got good coverage

WITH MORE TIME

- Testing
 - Should have over 80% coverage
- Optimization
 - Faster code
- Thorough documentation
 - Easier for the user to use
 - Code examples

```

class BigFile:
    def __init__(self, datadir, ndims):
        idfile = os.path.join(datadir, "id.txt")
        self.names = [x.strip() for x in str.split(open(idfile).read()) if x.strip()]
        self.name2index = dict(zip(self.names, range(len(self.names))))
        self.ndims = ndims
        self.featurefile = os.path.join(datadir, "feature.bin")
        print "[BigFile] %d features, %d dimensions" % (len(self.names), self.ndims)
        print "      binary: %s" % self.featurefile
        print "      txt: %s" % idfile

    def read(self, requested, isname=True):
        if isname:
            index_name_array = [(self.name2index[x], x) for x in requested]
        else:
            assert(min(requested) >= 0)
            assert(max(requested) < len(self.names))
            index_name_array = [(x, self.names[x]) for x in requested]
            index_name_array.sort()

        vecs = seq_read(self.featurefile, self.ndims, [x[0] for x in index_name_array])
        return [x[1] for x in index_name_array], vecs

    def shape(self):
        return [len(self.names), self.ndims]

```

FINAL PRODUCT

```

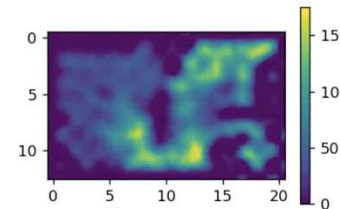
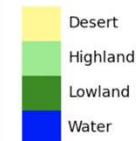
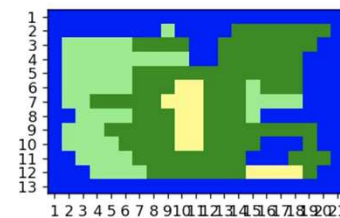
        'pop': [{'species': 'Herbivore',
                  'age': 5,
                  'weight': 20}
                for _ in range(150)]]
ini_carns = [{'loc': (9, 6),
                'pop': [{'species': 'Carnivore',
                          'age': 2,
                          'weight': 45}
                        for _ in range(300)]]]
new_carns = [{'loc': (4, 13),
               'pop': [{'species': 'Carnivore',
                         'age': 5,
                         'weight': 30}
                       for _ in range(10)]]]

sim = BioSim(island_map=geogr, ini_pop=ini_herbs+ini_carns,
             seed=123456,
             hist_specs={'fitness': {'max': 1.0, 'delta': 0.05},
                        'age': {'max': 60.0, 'delta': 2},
                        'weight': {'max': 60, 'delta': 2}},
             vis_years=1)

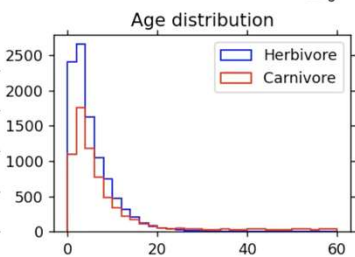
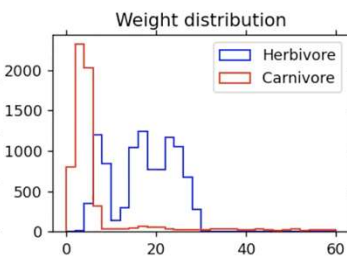
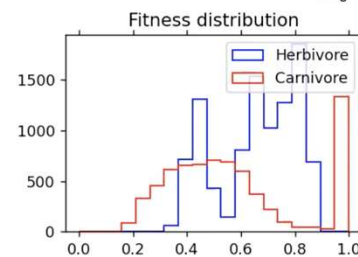
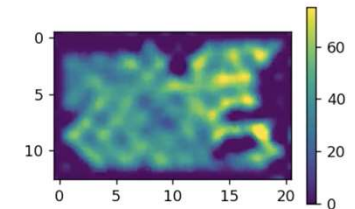
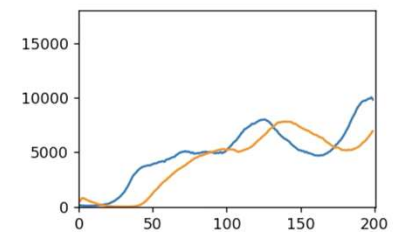
sim.set_animal_parameters('Herbivore', {'zeta': 3.2, 'xi': 1.8})
sim.set_animal_parameters('Carnivore', {'a_half': 60, 'phi_age': 0.5,
                                         'omega': 0.3, 'F': 65,
                                         'DeltaPhiMax': 9.,
                                         "mu": 0.9})
sim.set_landscape_parameters('L', {'f_max': 1200})

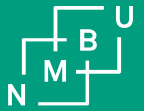
sim.simulate(num_years=100)
sim.add_population(population=new_carns)
sim.simulate(num_years=100)
sim.make_movie()

```



Simulation Results After
Year 199





Thank you!

