DAN vs HSMA eat off
Rule 1. Take it in turns to take 1-3 sweets from the bag (HSMA starts)

Module 3 : Modelling Pathway and Queuing Problems

Session 3B : SimPy for Discrete Event Simulation Part 1

Dr Daniel Chalk

"SimPy the Best"

NIHR | Applied Research Collaboration South West Peninsula

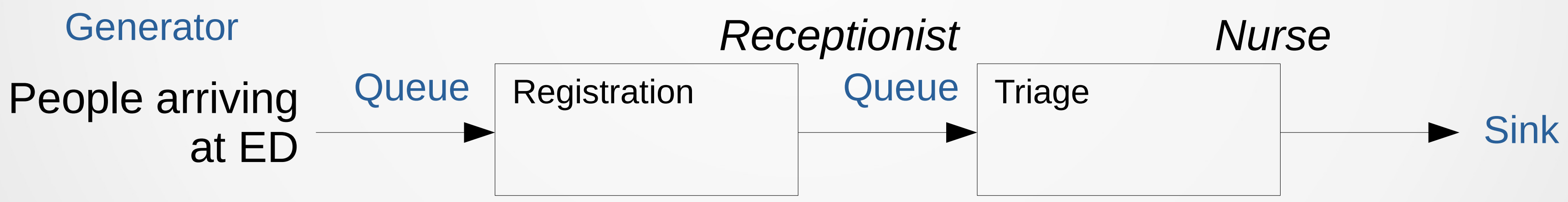
@peninsula_ARC #HSMA4

HSMA 4 Everyone

More than one sequential activity - Linear

In most DES models, we'd have our more than one activity in our system.

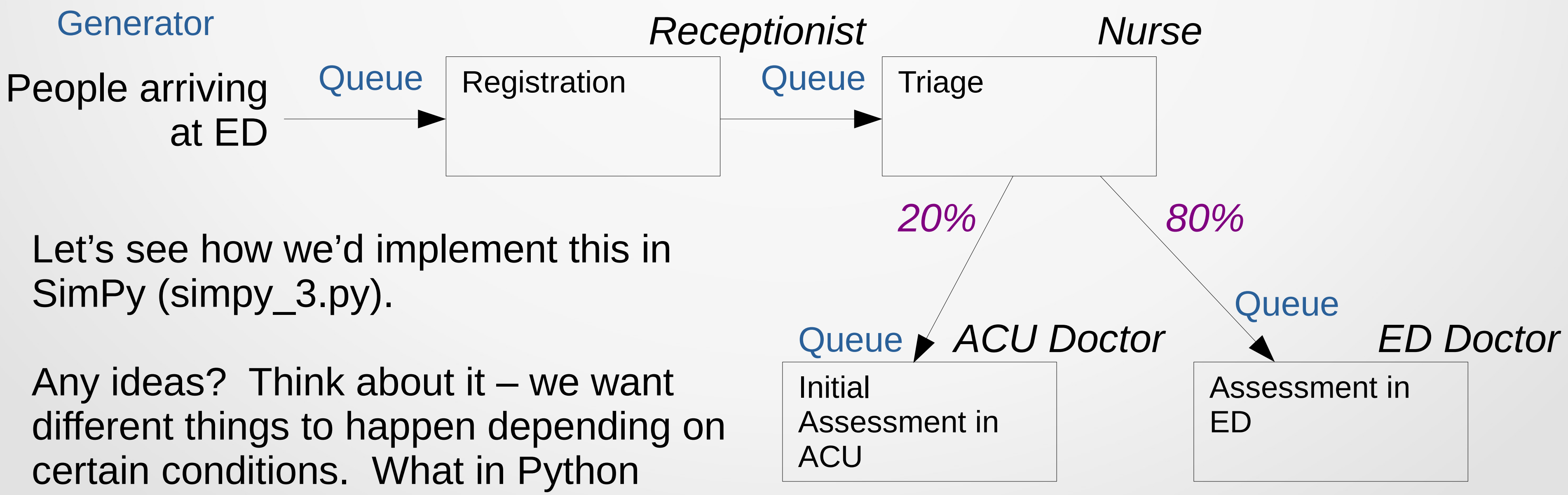
Let's first consider a simple *linear* example, where patients queue for one activity, and then queue for a second activity.



In the above example, we have two activities that patients queue for sequentially, and two types of resource – receptionists and nurses. Let's see how we'd implement that in SimPy (simpy_2.py). (simpy_1.py contains a stripped back version of the example you saw in the last session)

More than one sequential activity - Non-Linear

Often in real world systems, not everything is linear. Different things might happen depending on certain conditions. For example, after triage a patient might be referred to a Ambulatory Care to diagnose and treat the patient without being admitted to the hospital overnight.



Let’s see how we’d implement this in SimPy (simpy_3.py).

Any ideas? Think about it – we want different things to happen depending on certain conditions. What in Python could we use for that?

Storing Results

Let’s say we want to store results over the simulation run so we can, for example, calculate the average queuing time across patients.

One way we could do this would be to set up a list to store the results we’re interested in, and then add each patient’s results to the list.

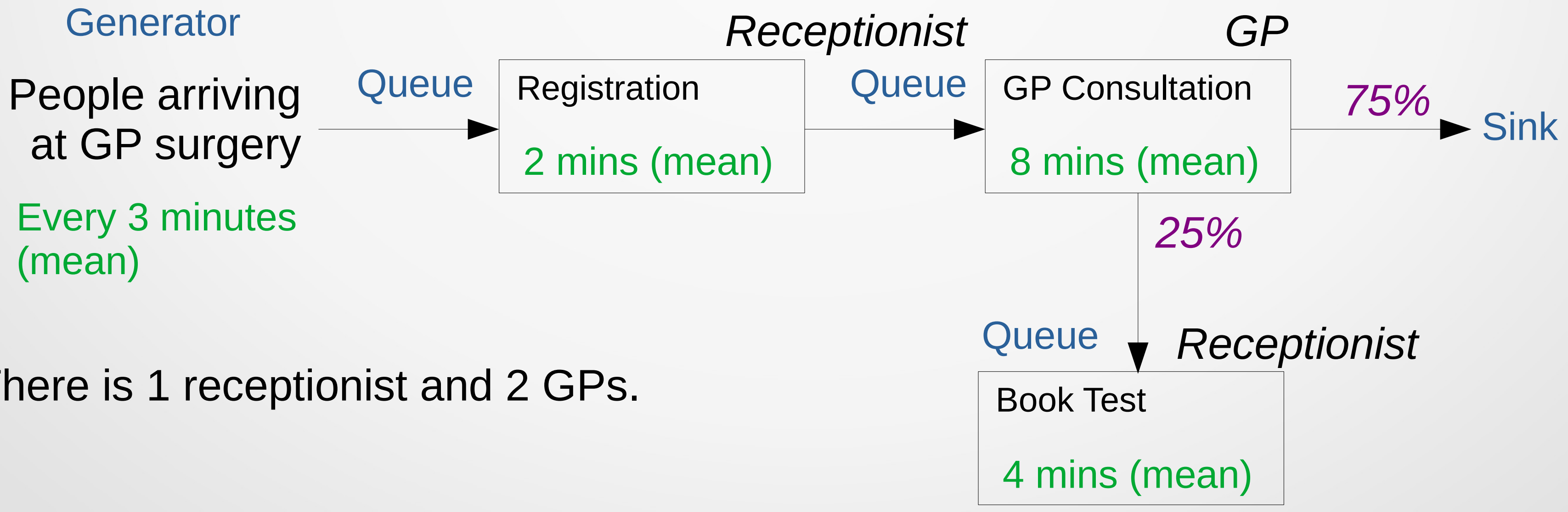
If we do that, we need to make use of the *global* keyword.

Let’s look at an example (simpy_4.py).

Exercise 1

Build a DES model in SimPy of the below system. Run for 8 hours of simulated time. Add functionality to the model to calculate and print the mean queuing times for each of the three queues, along with the mean time in system (from point of arrival to the surgery, to point of departure). For extra points, additionally plot a bar chart showing these times.

You have 1 hour (+ 5 minute comfort break). You may choose to work as a group, or individually.



There is 1 receptionist and 2 GPs.

Multiple Generators

It’s quite possible we’ll have more than one way for our entities to enter our system in our model. They may or may not go through the same processes.

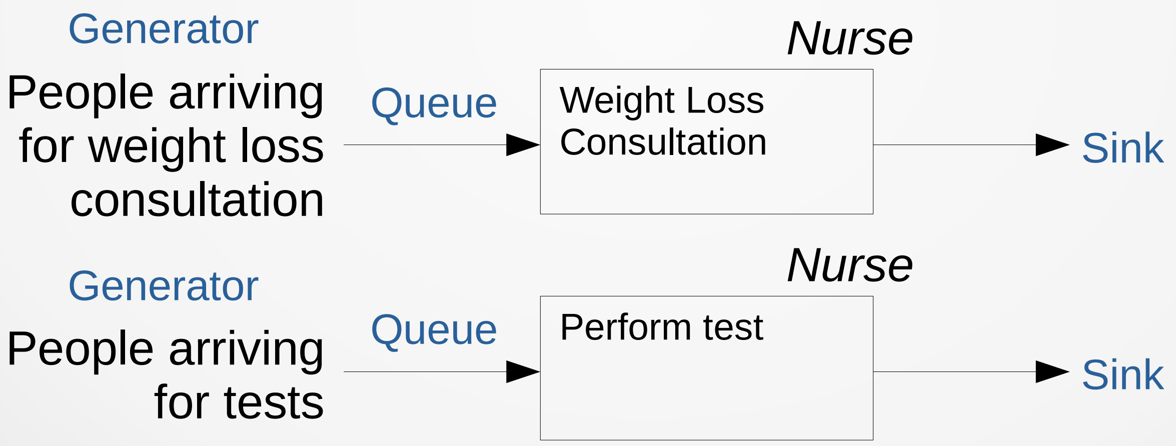
For example, patients might arrive at an ED by self-presenting or by ambulance, but the same things might happen to them regardless.

Or, in our GP surgery example from the first session, people may be coming in to the same place and using the same resources but for different reasons (e.g. coming to see nurse for test, or for weight loss clinic).

Multiple Generators in SimPy

Either way, it’s easy to capture multiple entry points into our system in SimPy – we just set up more than one entity generator!

Let’s look at an example (simpy_5.py).



Multiple Runs

In a stochastic model, it is important that we do not just run a model once if we’re looking to draw insights from our results.

This is because every run of the simulation will have different random samples for inter-arrival times, activity times etc.

What if you had a run with unusually long activity times sampled (a run of “bad luck”)? Or unusually long inter-arrival times (a run of “good luck”)?

We need to run a stochastic simulation many times and take summary statistics over the results from each run to get more representative results from the model.

Multiple Runs in SimPy

In SimPy, to run a simulation multiple times, we put *all* of the **setup** (including the establishment of the environment) in a **for loop**, with the range specified as the number of runs we want.

This means that the simulation resets and starts anew after every run, so if we want to store individual simulation run results (which we need to calculate averages) we either need to :

- have a global list outside of the loop that stores the results from each run

OR

- write the results of each run to file

Let’s look at an example (simpy_6.py), where we’ve added multiple runs to our previous simulation model.

Warm Up

A potential problem in our DES models is that, by default, we assume that the system is empty of entities (e.g. patients) when the system starts.

In some cases, that might be fine – for example, if we’re simulating a day of Minor Injury Unit activity, where patients turn up from when it opens and there’s nobody left when it closes.

But imagine we’re modelling an ED or a hospital more generally. These systems are never empty, and so if we run our model starting from empty, then our results are going to be skewed by the fact that early on in the simulation, there were no or very few people in the model.

Fortunately, there’s a simple solution – we can use a *warm up period*.

Warm Up Period

During a warm up period, our model runs, but it doesn’t collect any results. The warm-up period allows the simulation to get to a more representative state before we start collecting results.

The length of our warm up period will vary depending on the system we’re modelling. We might choose to just set a sufficiently lengthy warm up period to ensure the system is more representative once results start to be taken, or we might run the simulation multiple times and track the results over time to see when a steadier state appears to be reached.

Either way, we can easily add a warm up period to our SimPy models by simply using conditional logic to ensure that results are not collected until the simulation time has passed the warm up period. Let’s look at an example (simpy_7.py).

Containers

So far, we’ve considered resources as being discrete identifiable entities. But what if our resource was continuous, or if we wanted to model a homegenous mass of resource.

For example, imagine we were modelling the fuelling of a car, and our resource was petrol.

Or, maybe rather than model number of nurses, we want to model simultaneous nurse time in use, from which we take from and add back to a pool of time as incoming activity fluctuates.

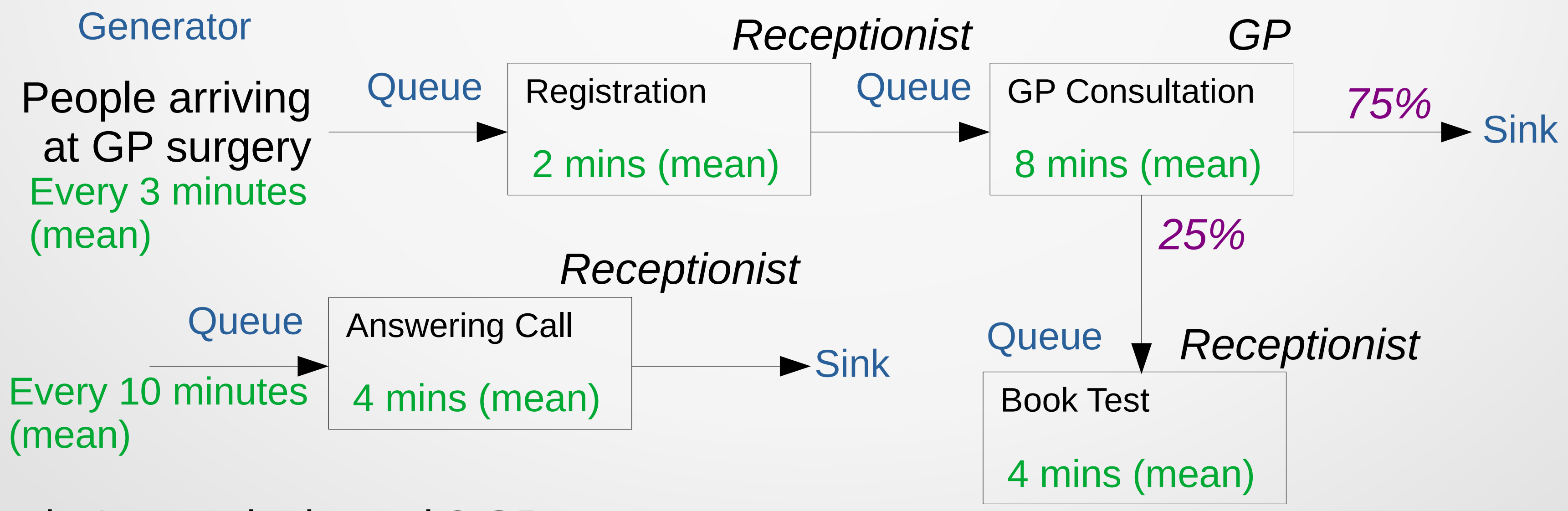
In SimPy, rather than use the Resource class, we can use the *Container* class for these kinds of problems.

Let’s look at an example (simpy_8.py).

Exercise 2

Extend the DES model you built in Exercise 1 to include :
- calls coming in to reception on average every 10 minutes, and which last an average of 4 minutes, answered by the receptionist. Don't worry about adding any results for call queues, or adding these calls to times in system results. We just want to model the drain on the receptionist resource for other activities (registering patients and booking tests)
- a warm up period of 3 hours before the 8 hours results collection period
- functionality to run the simulation multiple times – have the simulation run 100 times, with average results taken over these runs

You have 40 minutes. We'll share the solution at the end of this session.



There is 1 receptionist and 2 GPs.