Stack-of-Tasks Cheat sheet

==== General commands ===

entity.help()
# displays the help for an entity, namely the commands with the description

entity.commands()
# displays the commands of an entity, without description

entity.displaySignals()
# lists the signals of an entity, indicates whether they are plugged or not.

entity.signal.value
display the current value of a signal

entity.signal.time
display the computation time of the signal.

entity.signal.recompute(T)
# force the recomputation of the signal for time T (only effective it the given time is greater than the current time).

=== Signal connection ===

plug (e1.signal1, e2.signal2)
plug signals.

entity.signal.unplug()
# unplug signal.

entity.signal.displayDependencies(T)
# displays the signals linked to ent.sign

=== Entity creation ===

name = EntityType('name')
# create an EntityType called name

=== SoT entity ===

Entities: SOT/ SoTKine / SoTDyn

Possible commands:
sot # display the sot

sot.clear()
# remove all tasks, (but leave the contacts)

sot.rm(task_name)
# remove given task.

sot.push(task_name)
# insert task.

sot.pop()
# remove last task.

sot.up(task_name)
# lower the priority of the task

sot.down (task_name)
increase the priority of the task.

sot.addContact(contact.name)
sot.rmContact(contact.name)

solver is a wrapper for the SOT, defining the methods remove /add (feature)
solver.sot
corresponds to the sot entity.

=== Feature entity ===

Entities: Feature1D, Feature3D, Feature6D, FeatureGeneric, FeatureVisualPoint

feature.setReference(featureDes.name)
# defines the desired value signal

feature.selec.value = ’10…’
# limits the dof considered by the feature (1 activated / 0 not activated), *in reverse polish notation*

=== Task entity ===

task.add(feature.name)
# associate a feature to the task.

=== Tracer entity ===

tracer.setBufferSize(tracerSize)

tracer.open('folder','prefix','.extension')

tracer.start()
#start saving

tracer.dump()
#write the data

tracer.stop()
tracer.close()