### Pytransitions Quick Reference Sheet

*Note: For quick reference only. Mostly code demo. Full guide w/ more explanations found here:* [*https://github.com/pytransitions/transitions*](https://github.com/pytransitions/transitions)

**Quick setup**

class Matter(object):  
 pass

lump = Matter()

states=['solid', 'liquid', 'gas', 'plasma']

transitions = [ { 'trigger': 'melt', 'source': 'solid', 'dest': 'liquid' } ]

**machine = Machine(lump, states=states, transitions=transitions, initial='liquid')**

**lump.melt()**

**lump.triger('melt')**

*States*

|  |  |  |  |
| --- | --- | --- | --- |
| *Callbacks*   |  | | --- | | ***on\_enter****: callbacks declared on the destination state* | | ***on\_exit****: callbacks declared on the source state)* | | class Matter(object):  def say\_hello(self): print("hello, new state!")  def say\_goodbye(self): print("goodbye, old state!")    states = [  State(name='solid', **on\_exit=['say\_goodbye'])**,  State('liquid')  'gas',  { 'name': 'plasma'}  ]    **machine.on\_enter\_gas('say\_hello')** |
| *Check state* | lump.**state**  lump.**is\_gas()**  machine.**get\_state(lump.state)**.name |

**Transitions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Add trans* | |  | | --- | | transitions = [   **{ 'trigger': 'melt', 'source': 'solid', 'dest': 'liquid' },  { 'trigger': 'evaporate', 'source': 'liquid', 'dest': 'gas' }**  ] | | transitions = [  **['melt', 'solid', 'liquid'],  ['evaporate', 'liquid', 'gas']**  ] | | machine.**add\_transition('melt', source='solid', dest='liquid')** | |
| *Hide trigger error* | m = Machine(lump, states, initial='solid', **ignore\_invalid\_triggers=True**) |
| *Get its triggers* | |  | | --- | | m.**get\_triggers('solid')** | | m.**get\_triggers('solid', 'liquid', 'gas', 'plasma')** | |
| *Forced trans* | lump.**to\_liquid()** |
| *Triggers from many states* | |  | | --- | | machine.add\_transition('transmogrify', **['solid', 'liquid', 'gas']**, 'plasma') | | machine.add\_transition('to\_liquid', **'\*'**, 'liquid') | |
| *Reflexive trans* | machine.add\_transition('touch', ['liquid', 'gas', 'plasma'], **'=', after='change\_shape'**) |
| *Ordered trans* | states = ['A', 'B', 'C'] machine = Machine(states=states, initial='A') **machine.add\_ordered\_transitions() machine.next\_state()** |
| *Queued trans* | machine = Machine(states=states, **queued=True**) |
| *Conditional trans* | class Matter(object):  def is\_flammable(self): return False  def is\_really\_hot(self): return True   |  |  | | --- | --- | | machine.add\_transition('heat', 'solid', 'gas', **conditions='is\_flammable'**) machine.add\_transition('heat', 'solid', 'liquid', **conditions='is\_really\_hot'**) | *if is\_flammable true, 'solid' -> 'gas'; otherwise, if is\_really\_hot true, 'solid' -> 'liquid'* | | machine.add\_transition('heat', 'solid', 'gas', **unless=['is\_flammable', 'is\_really\_hot']**) | *run heat() unless both is\_flammable and is\_really\_hot return False* | | lump.heat(**temp=74**)  # = lump.trigger('heat', temp=74) | *passed to is\_flamable as optional kwarg* | |
| *Callbacks* | |  |  | | --- | --- | | ***before, after***  *(before / after transition)* | class Matter(object):  def make\_noises(self): print("HISSSSSSSSSSSSSSSS")  def disappear(self): print("where'd all the liquid go?")    transitions = [  {'trigger': 'melt', 'source': 'solid', 'dest': 'liquid', **'before': 'make\_noises'**},  {'trigger': 'evaporate', 'source': 'liquid', 'dest': 'gas', **'after': 'disappear'** } ] | | ***prepare***  *(executed as soon as trans starts, before conditions or callbacks)* | class Matter(object):  heat = False  attempts = 0  def heat\_up(self): self.heat = random.random() < 0.25  def count\_attempts(self): self.attempts += 1  def is\_really\_hot(self): return self.heat  def stats(self): print('#Attempts: %i' %self.attempts)    transitions = [ { 'trigger': 'melt', 'source': 'solid', 'dest': 'liquid', **'prepare': ['heat\_up', 'count\_attempts']**, 'conditions': 'is\_really\_hot', 'after': 'stats'}] | | ***default callbacks***  *(declared on Machine, before / after trans)* | m = Machine(lump, states, **before\_state\_change='make\_noises', after\_state\_change='disappear'**) | | |  | | --- | | ***prepare\_event***  *(executed once before trans are processed)* | | ***finalize\_event***  *(executed even if no trans occur or exception is raised)* | | ***send\_event***  *(error is retrievable)* | | class Matter(object):  def prepare(self, event): print("I am ready!")  def raise\_error(self, event): raise ValueError("Oh no")  def finalize(self, event): print("Result: ", type(event.error), event.error)    m = Machine(lump, states, **prepare\_event='prepare',** before\_state\_change='raise\_error', **finalize\_event='finalize', send\_event=True**) | |

**Alternative initialization patterns**

|  |  |
| --- | --- |
| *Model inherited from Machine class* | class Matter(Machine):  def \_\_init\_\_(self):  states = ['solid', 'liquid', 'gas']  **Machine.\_\_init\_\_(self, states=states, initial='solid')**  self.add\_transition('melt', 'solid', 'liquid') |

**Add/remove multiple models in single machine**

lump1 = Matter()  
lump2 = Matter()

machine = Machine(states=states, transitions=transitions, initial='solid', add\_self=False)

**machine.add\_model(lump1)  
machine.add\_model(lump2, initial='liquid')**

**machine.remove\_model([lump1, lump2])  
del lump1  
del lump2**