Assignment #0 - Ayça Avcı (s4505972)

November 21, 2021

Addition model

[1]:
```
from model import Model
from dmchunk import Chunk
```

[2]:
```
m = Model()

numbers = ["zero", "one", "two", "three", "four", "five", "six"]
for i in range(0, len(numbers) - 1):
    fact = Chunk(name = "cf" + numbers[i], slots = {"isa": "count-fact", "num1":
 ↪ numbers[i], "num2": numbers[i+1]})
    m.add_encounter(fact)
```

[3]:
```
def add(num1, num2):
    g = Chunk(name = "goal", slots = {"isa": "sum-goal", "start": num1, "count":
 ↪ num2})
    m.goal = g
    done = False
    while not done:
        if not "current" in g.slots:
            # setting counter to a zero
            g.slots["counter"] = m.get_chunk("cfzero").slots["num1"]
            counter = Chunk(name = "counter", slots = {"isa": "count-fact",␣
 ↪"num1": g.slots["counter"]})
            m.time += .05
            chunk2, latency2 = m.retrieve(counter)
            m.time += latency2
            # setting current sum to a start, which means the num1 given in the␣
 ↪function arguments
            g.slots["current"] = g.slots["start"]
            request = Chunk(name = "request", slots = {"isa": "count-fact",␣
 ↪"num1": g.slots["current"]})
            m.time += .05
            chunk1, latency1 = m.retrieve(request)
            m.time += latency1
            # if we have a case like 0 addition (add("two", "zero")), it just␣
 ↪prints out the num1.
```

```
            # it does not work where the num1="six" since there is not a chunk␣
↪created initially that contains "six"
            # as num2.
            if g.slots["count"] == g.slots["counter"]:
                print(g.slots["current"])
                done = True
            # else it prints out the current, and sets it to the next number in␣
↪the chunk. Also sets the counter
            # to the next number, which means increasing it by one, in the␣
↪chunk that counter's initial value set.
            else:
                print(g.slots["current"])
                g.slots["current"] = chunk1.slots["num2"]
                m.time += 0.3
                g.slots["counter"] = chunk2.slots["num2"]
                m.time += 0.3
        elif g.slots["count"] != g.slots["counter"] and not done:
            counter = Chunk(name = "counter", slots = {"isa": "count-fact",␣
↪"num1": g.slots["counter"]})
            m.time += .05
            chunk2, latency2 = m.retrieve(counter)
            m.time += latency2
            g.slots["counter"] = chunk2.slots["num2"]
            m.time += 0.3

            request = Chunk(name = "request", slots = {"isa": "count-fact",␣
↪"num1": g.slots["current"]})
            m.time += .05
            chunk1, latency1 = m.retrieve(request)
            m.time += latency1
            print(g.slots["current"])
            g.slots["current"] = chunk1.slots["num2"]
            m.time += 0.3
        else:
            print(g.slots["current"])
            done = True
```

[4]:
```
add("one", "five")
```

```
one
two
three
four
five
six
```