add_JonasSchweisthal_s4535561

November 15, 2020

0.0.1 Homework #1

Name: Jonas Schweisthal
Student number: s4535561

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

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

[3]:
```
numbers = ["zero","one","two","three","four","five","six", "seven", "eight",␣
 ↪"nine", "ten", "eleven",
           "twelve", "thirteen", "fourteen", "fiveteen","sixteen", "seventeen",␣
 ↪"eighteen", "nineteen", "twenty"]
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)
```

[4]:
```
## old count function:

# def count_from(start, end):
#     g = Chunk(name = "goal", slots = {"isa":"count-goal","start":start,"end":
 ↪end})
#     m.goal = g
#     done = False
#     while not done:
#         try:
#             if not "current" in g.slots:
#                 g.slots["current"] = g.slots["start"]
#                 request = Chunk(name = "request", slots = {"isa":
 ↪"count-fact", "num1":g.slots["current"]})
#                 m.time += .05
#                 chunk, latency = m.retrieve(request)
#                 m.add_encounter(chunk)
#                 m.time += latency
#                 # print(m.time)
#                 print(g.slots["current"])
#                 g.slots["current"] = chunk.slots["num2"]
#                 m.time += 0.3
#             elif g.slots["current"] != g.slots["end"]:
#                 request = Chunk(name = "request", slots = {"isa":
↪"count-fact", "num1":g.slots["current"]})
#                 m.time += .05
#                 chunk, latency = m.retrieve(request)
#                 m.add_encounter(chunk)
#                 m.time += latency
#                 # print(m.time)
#                 print(g.slots["current"])
#                 g.slots["current"] = chunk.slots["num2"]
#                 m.time += 0.3
#             else:
#                  #print(m.time)
#                 print(g.slots["current"])
#                 done = True
#         except:
#             print("Error")
```

[5]:
```
## pragmatic solution:

# def pragmatic_add(start, end):
#     count_end = numbers[numbers.index(start) + numbers.index(end)]
#     count_from(start, count_end)
```

[6]:
```
# solution Homework 1:
# adding tow numbers by counting

def add(start, end):
    g = Chunk(name = "goal", slots = {"isa":"count-goal","start":start,"end":
↪end})
    m.goal = g
    done = False
    while not done:
        try:
            if not "current_sum" in g.slots:
                # initialize current sum and counter
                g.slots["current_sum"] = g.slots["start"]
                g.slots["current_count"] = "zero"
                request = Chunk(name = "request", slots = {"isa":"count-fact",␣
↪"num1":g.slots["current_sum"]})
                m.time += .05
                chunk, latency = m.retrieve(request)
                m.add_encounter(chunk)
                m.time += latency
                # checking if "zero" is added or if a counter is needed
                if end != "zero":
                # print(m.time)
                    print(g.slots["current_sum"])
                    g.slots["current_sum"] = chunk.slots["num2"]
                    request = Chunk(name = "request", slots = {"isa":
↪"count-fact", "num1":g.slots["current_count"]})
                    m.time += .05
                    chunk, latency = m.retrieve(request)
                    m.add_encounter(chunk)
                    m.time += latency
                    g.slots["current_count"] = chunk.slots["num2"]
                m.time += 0.3

            # checking if "zero" is added or if a counter is needed
            elif g.slots["current_count"] != g.slots["end"] and end != "zero":
                # adding/counting till end number is reached
                request = Chunk(name = "request", slots = {"isa":"count-fact",␣
↪"num1":g.slots["current_count"]})
                m.time += .05
                chunk, latency = m.retrieve(request)
                m.add_encounter(chunk)
                m.time += latency
                g.slots["current_count"] = chunk.slots["num2"]
                request = Chunk(name = "request", slots = {"isa":"count-fact",␣
↪"num1":g.slots["current_sum"]})
                m.time += .05
                chunk, latency = m.retrieve(request)
                m.add_encounter(chunk)
                m.time += latency
                # print(m.time)
                print(g.slots["current_sum"])
                g.slots["current_sum"] = chunk.slots["num2"]
                m.time += 0.3
            else:
                 #print(m.time)
                print(g.slots["current_sum"])
                done = True
        except:
            print("Error: Number could not be retrieved")
```

[7]:
```
# Testing with example:
add("two", "four")
```

```
two
three
four
```

```
five
six
```

[8]:
```
# print(m)
```