



Recap of Webster's problem

Comp 140

Fall 2008



The "word problem"

- The devil made a proposition to Daniel Webster. The devil proposed paying Daniel for services in the following way: "On the first day, I will pay you \$1,000 early in the morning. At the end of the day, you must pay me a commission of \$100. At the end of the day, we will both determine your next day's salary and my commission. I will double what you have earned at the end of the day, but you must double the amount that you pay me. Will you work for me for a month?"



Recipe construction

- Only Webster's salary and the devil's commission are important for calculating Webster's decision.
- The key thing is how Webster's salary and the devil's commission evolve from one day to the next.
→ we use an algebraic equation to represent this relationship succinctly.
- We compute only what we need to make a decision (Goal: compute w_{30})
- We apply decision rule ($w_{30} < 0$) to solve problem

Abstraction

30 August 2008

$$w_0 = 1000$$

$$d_0 = 100$$

for $t = 0, 1, \dots, 29$

$$w_{t+1} = 2(w_t - d_t)$$

$$d_{t+1} = 2d_t$$

Reject deal if $w_{30} < 0$,
else accept deal

Automation

(c) Devika Subramanian, 2008



The final recipe in Python

```
w = [1000]
d = [100]
```

← Seed the two sequences
w and d

```
for t in range(30):
```

← Do the following for t=0
through t=29, in order

```
    w.append(2*(w[t]-d[t]))
    d.append(2*d[t])
```

← Extend the
sequences w
and d for
each value of t

```
    if w[30]<0:
        print "Sorry, no deal!"
    else:
        print "Yeah, take the deal!"
```

← make the decision

PYTHON



Decision rules

- Current rule: Accept deal if the salary on the 30th day is greater than or equal to zero, reject otherwise.
- New rule: Accept deal if the sum of the takes up to the the 30th day is greater than or equal to 0, reject otherwise
 - Recall take on day t is salary on day t - commission on day t
 - If $v = [v_0, \dots, v_n]$, $\text{sum}(v) = v_0 + v_1 + v_2 + \dots + v_n$



A different decision rule

```
w = [1000]
d = [100]
```

Seed the two sequences
w and d

```
for t in range(30):
```

Do the following for t=0
through t=29, in order

```
    w.append(2*(w[t]-d[t]))
    d.append(2*d[t])
```

Extend the
sequences w
and d for
each value of t

```
    if (sum(w)-sum(d))<0:
        print "Sorry, no deal!"
    else:
        print "Yeah, take the deal!"
```

make the decision



Variation #1

- Suppose Webster wants out of the deal on the day before his salary for that day is less than zero.
- Compute when Webster should quit working for the devil, and can make the following counteroffer:
 - I will take your starting salary of \$1000 and give you a starting commission of \$100, and follow your other rules, but I will quit on the t -th day.
- Now make the decision based on his net take (sum of takes up to day t)



Homework problem 1a

- Identify the key abstraction choices, then write the recipe for Variation 1 using mathematics, and translate it into a Python program after choosing an appropriate computational mapping.
- Run the Python program interactively, and report the answer.
- Try the second decision rule (quitting based on net take) and see if the answer is different.



Variation #2

- Suppose Webster wants out of the deal before his salary on a day is less than zero.
- Calculate the **starting salary** that Webster should negotiate so he can work for the devil according to the devil's terms in the original problem (the full 30 days).
- Identify the abstraction choices, then write the recipe in mathematics, and finally in Python.
- Run your Python program and report the answer.
- Replace "salary on specific day" by "net take up that day" in the decision rule above, and see if your answer is different.



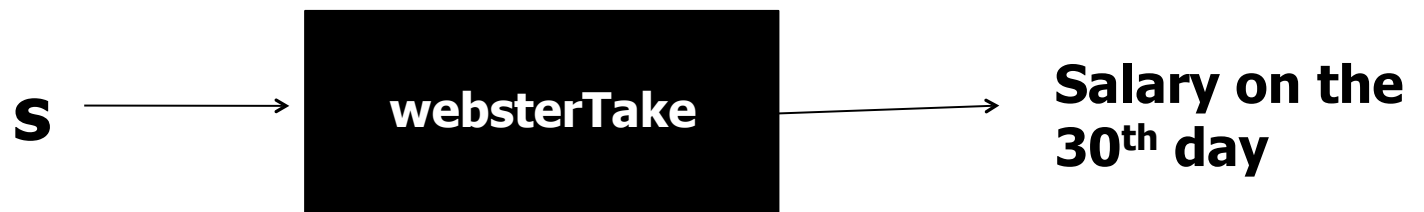
Homework problem 1b

- Identify the key abstraction choices, then write the recipe for Variation 2 using mathematics, and translate it into a Python program after choosing an appropriate computational mapping.
- Run the Python program interactively, and report the answer.
- Try the second decision rule (quitting based on net take) and see if the answer is different.



A useful construct

- A function that takes a start salary as input and computes Webster's salary on the 30th day





Functions in Python

```
def double(x):  
    return 2*x
```

```
double(2)  
double(double(2))  
double('hello')
```



websterTake function

```
def websterTake(s):  
    """ Calculates Webster's salary on the 31st day starting with  
    a salary of s on day 0 and devil's commission that starts at 100  
    and doubles each day """  
  
    w = [s]  
    d = [100]  
    for t in range(30):  
        w.append(2*(w[t]-d[t]))  
        d.append(2*d[t])  
    return w[30]
```



Starter hint for Problem 1b

- Armed with function, `websterTake`, how can you find the start salary s for which Webster's salary on the 31st day is exactly zero?



Problem 2

- Read the essay on Computational Thinking by J. Wing.
 - Contrast Wing's views on computer science with what your view of computer science was before you read the article, and before the first comp140 lecture.



Format of answer

- **Structure**

- Paragraph 1: What I thought computer science was about (e.g., in high school), and why you held that view (point to supporting evidence)
- Paragraph 2: My new view of computer science (after lecture 0 of comp140 and Wing's article). Identify the most compelling argument in Wing's article for you.
- Paragraph 3: If your "old" and "new" views are the same, explain why, with supporting arguments and evidence.