CompSci 101
Fall 2021

Lecture 10
Reminders

- **Regrade Requests:** 9/28-9/30 via Gradescope

- **Assignments**
  - Assign 2 due 10/7
  - APT 3 due 10/7

- **Small-group tutoring**
Key instructions

• Input ←
• Output ←
• Assignments* ✔
• Math/Logic ✔
• Conditionals ✔
• Repetition ←

*not listed in book
Python Data Types

• int, float, bool ✔
• Collections
  • Strings ✔
  • Lists ✔
  • Tuples
  • Sets
  • Dictionaries
PFTD

• Pancakes
• Loops
  • While loops
  • Nested for loops

“The mere imparting of information is not education.”
• Dr. Carter G. Woodson
KISS Principle

- Think of the non-computing context for any word/terms
- KISS model
  - Work smarter, not harder!!
- “Good programmers are simply good designers.”
  - Dr. Washington
- Design first and always!
- Importance of reusability
- USE PyCharm/PythonTutor IF YOU HAVE QUESTIONS!
People to Know:
Dr. Patricia Ordóñez

- PhD (CS)-University of Maryland, Baltimore County
- Associate Professor, CS
  - University of Puerto Rico Rio Piedras
- Visual analytics, data mining, machine learning
Pancakes!
APT Pancake

- $\text{minutesNeeded}(\text{numCakes, capacity})$
- How do you solve this (or any) problem?
  - 7 Steps!

- Some APTs are hard problems to solve (step 1-4)
  - Translating to code easy
- Some APTs have easy-to-see algorithms (step 5)
  - Translating to code is hard
Step 1: Solve an instance
Three pancakes in a two-cake pan

- First 5 minutes
  - 2 half cooking
  - 1 uncooked

- Second 5 minutes
  - 2 half cooking
  - 1 almost cooked
Step 1: Solve an instance
Three pancakes in a two-cake pan

- Third 5 minutes
  - 1 done
  - 2 almost cooked
- How many minutes to cook all three pancakes?
Step 1: Solve an instance

• What kind of instances? Simple cases that are quickly solved
  • What are these in Pancake problem?

• Don’t solve for N, solve for 5 (generalize is step 3)
  • What do when there are two parameters?
    • Fix one, vary the other one
  • Helps identify cases
Step 1: Solve an instance

- Pan has capacity 8, vary # pancakes
  - Can you cook 12 in 15 minutes? Why?
  - Can you cook 13 in 15 minutes? Why?

<table>
<thead>
<tr>
<th>cakes</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. 1 2 3 4 5 6 7 8
2. 1 2 3 4 10 11
3. 7 5 6 8
Step 2: What did we just do?

• 13 – 8 = 5
• 8/2 = 4 # Can only take off up to half
• Is 5 <= 4?
  • No, warmer trick won’t work
• 10 minutes for 8 pancakes + 10 minutes for 5 more pancakes = 20 minutes
Step 1: Solve an instance

- Pan capacity 8, vary # pancakes, 17 pancakes?

<table>
<thead>
<tr>
<th>cakes</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

17 pancakes

Pan capacity 8, vary # pancakes, 17 pancakes?
Step 2: What did we just do?

- $17 - 8 = 9$, $9 - 8 = 1$
- $8/2 = 4$
- Is $1 \leq 4$? # Yes, warmer trick will work!
- Total: 25 minutes
  - 10 minutes for 8 pancakes +
  - 5 minutes for 8 pancakes +
  - Take 1 out, start 17th pancake
  - 5 minutes finish pancakes 8 to 15 +
  - 5 minutes finish pancake 16 and 17
Step 3: Generalize

- Pan has capacity 8, Generalize to algorithm?

<table>
<thead>
<tr>
<th>cakes</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

![Diagram showing cake allocation with time in minutes]
Step 3: Generalize

• 13 – 8 = 5
• 8/2 = 4
• Is 5 <= 4?
  • No, warmer trick won’t work
• 10 minutes for 8 pancakes + 10 minutes for 5 more pancakes = 20 minutes

• Remove as many as can with panCapacity
• Will the remainder fit in half the pan?
• Yes, use warmer
  • 5 minutes instead of 10 for last batch
• No, don’t use warmer
  • 10 minutes for all sets of panCapacity
Step 4: Test Steps

- Remove as many as can with `panCapacity`
- Will the remainder fit in half the pan?
  - Yes, use warmer
    - 5 minutes instead of 10 for last batch
  - No, don’t use warmer
    - 10 minutes for all sets of `panCapacity`

- Case 1:
  - `cap 17`, `cook 34`
Step 4: Test Steps

• Remove as many as can with panCapacity
• Will the remainder fit in half the pan?
  • Yes, use warmer
    • 5 minutes instead of 10 for last batch
  • No, don’t use warmer
    • 10 minutes for all sets of panCapacity

• Case 1:
  • cap 17, cook 34
  • remainder = 0
  • Edge case! No need for warmer
  • Total: 20 minutes
• Case 2:
  • cap 17, cook 42
Step 4: Test Steps

- Remove as many as can with panCapacity
- Will the remainder fit in half the pan?
  - Yes, use warmer
    - 5 minutes instead of 10 for last batch
  - No, don’t use warmer
    - 10 minutes for all sets of panCapacity

- Case 1:
  - cap 17, cook 34
  - remainder = 0
  - Edge case! No need for warmer
  - Total: 20 minutes

- Case 2:
  - cap 17, cook 42
  - remainder = 8
  - Yes, use warmer
  - Total: 25 minutes
Step 5: Code

• Remove as many as can with panCapacity
• Will the remainder fit in half the pan?
  • Yes, use warmer
    • 5 minutes instead of 10 for last batch
  • No, don’t use warmer
    • 10 minutes for all sets of panCapacity

• N pancakes
• How many panCapacity can remove?
  • N // panCapacity
  • remainder
    • N % panCapacity
• Half of pan?
  • panCapacity / 2
In Summary

- When it comes to planning the algorithm (Steps 1-4) and coding (Step 5), each part can be easy or hard.
Why Solve This? In Python?

- 3n+1 sequence
  - Start with n
  - Next number:
    - n/2 → n is even
    - 3n+1 → n is odd
  - End sequence when n==1

- How many numbers are generated before it reaches 1?

- Mathematics is foundational in computer science, but
  - Not everyone enjoys logic/math puzzles, but …
Developing and Reasoning about While Loops

• Don't know: *how many times* loop executes
  • we'll know after finishes

• Do know: condition that should be true to enter/repeat (*BOOL_CONDITION*-loop guard)
  • When *BOOL_CONDITION* NOT TRUE → ends

```python
while *BOOL_CONDITION*:
    LOOP_BODY
    # modify variables, affect expression
```
History: From while to for loops

while loop (sum list)

lst = [4,1,8,9]
s = 0
i = 0

while i < len(lst):
    s += lst[i]
    i += 1

print(s)

for loop (sum list)

lst = [4,1,8,9]
s = 0

for n in lst:
    s += n

print(s)
Concrete Example: Collatz

• Don't know: *how many times* loop executes
  • some numbers: long sequences, others short

• Do know: condition that should be true to enter/repeat (**BOOL_CONDITION**-loop guard)
  • What is true after loop below finishes?

```python
while value != 1:
    loop body
    # modify value somehow
```
Activity 2: Collatz and While
Parallel Lists

• Case Study: FileFrequency.py

• We'd like to analyze word occurrences
  • Google N-Gram, it's easy to do, but …
  • What about Rotten Tomatoes?

• This code is built using the tools that we have
  • In the future, learn of more efficient structures

• We'll use an API for opening files
High Level View

• We will use parallel lists to track data
  • Each word is stored in a list named \texttt{words}
  • Word’s count is stored in a list named \texttt{counts}
  • \# occurrences of \texttt{words[k]} is in \texttt{counts[k]}

["apple", "fox", "vacuum", "lime"]
[5,2,25,15]

• What happens when we read a word?
Pseudo-code for getFileData

• Let user choose a file to open
• Read each line of the file
  • Process each word on the line
    • If word never seen before? Add to words and counts
    • Update # occurrences using .index and location

• TPS: What would we do for each color when doing step 5 (translate to code) of the 7 steps?
for line in f:
    data = line.strip().split()

for word in data:
    word = word.lower()

if word not in words:
    words.append(word)
    counts.append(0)

location = words.index(word)
counts[location] += 1

TPS: What is guaranteed about words and counts?
Comparing Two Approaches

- Why do we have a loop in a loop?
  - Code mirrors structure:
    - file has lines, lines have words

- Notice:
  - `strip`
  - `split`
  - `lower`
  - `not in`
  - `append`
  - `index`
  - `+=`

```python
for line in f:
    data = line.strip().split()
    for word in data:
        word = word.lower()
        if word not in words:
            words.append(word)
            counts.append(0)
            location = words.index(word)
            counts[location] += 1
```

Outer loop

Inner loop
Comparing Two Approaches

• Why do we have only one loop?
  • Code mirrors structure, which is better?
  • File is a sequence of characters!!

```python
for word in f.read().lower().split():
    if word not in words:
        words.append(word)
        counts.append(0)
    location = words.index(word)
    counts[location] += 1
```
Activity 3: File Frequency
Reminders

• Work smarter, not harder
• Design first
• Try to identify where you are stuck
  • Identify resources to help solve problem
• Leverage your design and PythonTutor to understand program flow of control
  • [http://pythontutor.com](http://pythontutor.com)