Midterm Review

Rob Thompson

UW CSE 160

Winter 2021
Midterm Setup

• Releases 8:00pm tonight
• Due 11:00pm Wednesday
• It will not take you 51 hours to finish

• Open-book, open-note, open-lecture, open-class-website
• Can ask questions on Ed and in Office Hours
• Can share anything within your group
ARE WE REALLY READY FOR THIS?

Yes! You’ve learned a lot, and you’ve been building a knowledge base this whole time.
You type *expressions*. Python computes their *values*.

- 5
- 3 + 4
- 44 / 2
- 2 ** 3
- 3 * 4 + 5 * 6
  - If precedence is unclear, use parentheses
- (72 – 32) / 9.0 * 5
Variables hold values

• Recall variables from algebra:
  – Let $x = 2$ ...
  – Let $y = x$ ...

• In Python assign a variable: “varname = expression”

  ```python
  pi = 3.14
  pi
  avogadro = 6 * 10 ** 23
  avogadro
  22 = x  # Error!
  ```

• Not all variable names are permitted
Types of values

- Integers (int): -22, 0, 44
  - Arithmetic is exact

- Real numbers (float): 2.718, 3.1415
  - float, for “floating point”
  - Arithmetic is approximate

- Strings (str): "I love Python", ""

- Truth values (bool): True, False
  - bool, for “Boolean”
for Loop Explained

A better way to repeat yourself:

```python
for fahr in [30, 40, 50, 60, 70]:
    cent = (fahr - 32) / 9.0 * 5
    print(fahr, cent)
print("All done")
```

Output:
30 -1.11
40 4.44
50 10.0
60 15.56
70 21.11
All done
The range function

A typical for loop does not use an explicit list:

```python
for i in range(5):
    # body ...
```

- `range(5)` → will loop through \[0, 1, 2, 3, 4\]
- `range(1, 5)` → will loop through \[1, 2, 3, 4\]
- `range(1, 10, 2)` → will loop through \[1, 3, 5, 7, 9\]
Nested Loops

for i in [1, 2, 3]:
    print("Before j loop i is", i)
    for j in [50, 100]:
        print("j is", j)

What is the output?
Using If to find absolute value

**If** the value is negative, negate it.

**Otherwise**, use the original value.

```python
val = -10

# calculate absolute value of val
if val < 0:
    result = -val
else:
    result = val

print(result)
```

In this example, `result` will always be assigned a value.
if height > 100:
    print("space")
elif height > 50:
    print("mesosphere")
elif height > 20:
    print("stratosphere")
else:
    print("troposphere")

ONE of the print statements is guaranteed to execute: whichever condition it encounters first that is true

See in python tutor
What Happens Here? (bad example)

```python
# height is in km
if height > 100:
    print("space")
if height > 50:
    print("mesosphere")
if height > 20:
    print("stratosphere")
else:
    print("troposphere")
```

Try height = 72
Creating a function

Define the machine, including the input and the result

```
def dbl_plus(x):
    return 2 * x + 1
```

Keyword that means: I am defining a function

Name of the function. Like “y = 5” for a variable

Input variable name, or “formal parameter”

Keyword that means: This is the result

Return expression (part of the return statement)
How to look up a variable

Idea: find the nearest variable of the given name

1. Check whether the variable is defined in the local scope
2. ... check any intermediate scopes (none in CSE 160!) ...
3. Check whether the variable is defined in the global scope

If a local and a global variable have the same name, the global variable is inaccessible ("shadowed")

This is confusing; try to avoid such shadowing

```python
x = 22
stored = 100
def lookup():
    x = 42
    return stored + x
val = lookup()
x = 5
stored = 200
val = lookup()
```
Local variables exist only while the function is executing

```python
def cent_to_fahr(cent):
    result = cent / 5.0 * 9 + 32
    return result

tempf = cent_to_fahr(15)
print(result)
```

ALL of your variables will be local for the midterm
How to design a function

1. Wishful thinking: Write the program as if the function already exists

2. Write a specification: Describe the inputs and output, including their types
   - No implementation yet!

3. Write tests: Example inputs and outputs

4. Write the function body (the implementation)
   - First, write your plan in English, then translate to Python

---

```python
def fahr_to_cent(fahr):
    """Input: a number representing degrees Farenheit
    Return value: a number representing degrees centigrade
    """
    result = (fahr - 32) / 9.0 * 5
    return result

assert fahr_to_cent(32) == 0
assert fahr_to_cent(212) == 100
assert fahr_to_cent(98.6) == 37
assert fahr_to_cent(-40) == -40

# Main program
tempf = 32
print("Temperature in Fahrenheit: ", tempf)

tempc = fahr_to_cent(tempf)
print("Temperature in Celsius: ", tempc)
```

---

Not relevant for the midterm

Already done for you!

We did some of these for you, you should make more to be sure

This is the main part of the midterm
What is a list?

• A list is an ordered sequence of values
  – A list of integers:
    
    [3, 1, 4, 4, 5, 9]

  – A list of strings:
    
    ["Four", "score", "and", "seven", "years"]

• Each value has an index
  – Indexing is zero-based (counting starts with zero)

• len([3, 1, 4, 4, 5, 9]) returns 6
List Creation

\[ a = [ 3, 1, 2 \times 2, 1, 10 \div 2, 10 - 1 ] \]

\[ b = [ 5, 3.0, 'hi' ] \]

\[ c = [ 4, 'a', a ] \]

\[ d = [ [1, 2], [3, 4], [5, 6] ] \]
List Querying

Expressions that return parts of lists:

• Single element: \texttt{mylist[index]}
  – The single element stored at that location

• Sublist (“slicing”): \texttt{mylist[start:end]}
  – the sublist that starts at index \texttt{start} and ends at index \texttt{end} – 1
  – If \texttt{start} is omitted: defaults to 0
  – If \texttt{end} is omitted: defaults to \texttt{len(mylist)}
  – \texttt{mylist[:]} evaluates to the whole list
  – \texttt{mylist[0:len(mylist)]} also does
More List Querying

• Find/lookup in a list
  \[ x \text{ in } \text{mylist} \]
  • Returns True if \( x \) is found in \text{mylist}

\text{mylist.index}(x)
  • Return the integer index in the list of the \textit{first item} whose value is \( x \).
  • It is an error if there is no such item.

\text{mylist.count}(x)
  • Return the number of times \( x \) appears in the list.
List Insertion

• **mylist.append**(x)
  – Extend *mylist* by inserting *x* at the end

• **mylist.extend**(L)
  – Extend *mylist* by appending all the items in the argument list *L* to the end of *mylist*

• **mylist.insert**(i, x)
  – Insert item *x* *before* position *i*.
  – *a.insert*(0, x) inserts at the front of the list
  – *a.insert*(len(a), x) is equivalent to

\[
\text{a.append}(x)
\]

**Note:** *append*, *extend* and *insert* all return *None*
List Removal

• `mylist.remove(x)`
  – Remove the first item from the list whose value is `x`
  – It is an error if there is no such item
  – Returns `None`

• `mylist.pop([i])`
  – Remove the item at the given position in the list, and return it.
  – If no index is specified, `a.pop()` removes and returns the last item in the list.

**Note:** `remove` returns `None`
List Replacement

• `mylist[index] = newvalue`
• `mylist[start:end] = newsublist`
  – Replaces `mylist[start]... mylist[end - 1]` with `news sublist`
  – Can change the length of the list
• `mylist[start:end] = []`
  – removes `mylist[start]... mylist[end - 1]`
• `mylist[len(mylist):] = L`
  – is equivalent to `a.extend(L)`
List expression examples

What does this mean (or is it an error)?

["four", "score", "and", "seven", "years"][2]

["four", "score", "and", "seven", "years"][0,2,3]

["four", "score", "and", "seven", "years"][[0,2,3]]

["four", "score", "and", "seven", "years"][[0,2,3][1]]
Reading a file in python

# Open takes a filename and returns a file object.
# This fails if the file cannot be found & opened.
myfile = open("datafile.dat")

# Approach 1: Process one line at a time
for line_of_text in myfile:
    ... process line_of_text

# Approach 2: Process entire file at once
all_data_as_a_big_string = myfile.read()

myfile.close() # close the file when done reading

Assumption: file is a sequence of lines
Where does Python expect to find this file (note the relative pathname)?
Writing to a file in python

# Replaces any existing file of this name
myfile = open("output.dat", "w")

# Just like print-ing output
myfile.write("a bunch of data")
myfile.write("a line of text\n")

myfile.write(4)
myfile.write(str(4))

myfile.close()