Lecture Notes
CSC111
Week 2 — Sept 14, 2015

Dominique Thiébaut
dthiebaut@smith.edu
• The Programming Process
• Memory: RAM
• Variables revisited
  • Literals: numbers, strings, lists
  • Types: type( )
  • Multiple assignments
  • Operators. Overloaded operators.
• Loops
  • range( ); list( )
• input( )
  • eval( ): string of digits => number
• Programming exercises
The Programming Process
The Programming Process

- Analyze the **Problem**
The Programming Process

- Analyze the Problem
- Determine Specifications
The Programming Process

- Analyze the **Problem**
- Determine **Specifications**
- Create a **Design**
The Programming Process

- Analyze the **Problem**
- Determine **Specifications**
- Create a **Design**
- **Implement**
The Programming Process

• Analyze the **Problem**
• Determine **Specifications**
• Create a **Design**
• **Implement**
• Test & Debug
The Programming Process

- Analyze the **Problem**
- Determine **Specifications**
- Create a **Design**
- **Implement**
- Test & Debug

*iterate many times*
The Programming Process
The Programming Process
The Programming Process
The Programming Process
The Programming Process
The Programming Process

Variables

Definite Loops

Input
Memory (last lecture)
What does memory really look like?
What does memory really look like?
What does memory really look like?
What does memory really look like?

Random Access Memory (RAM)
What does memory really look like?

Random Access Memory (RAM)

Single In-line Memory Module (SIMM)
What does memory really look like?

- RAM: 4, 8, 12, 16 GigaBytes
- **Giga** = billion: $10^9$ bytes
- In RAM: room for approximately 2 billion integers
- 1 number takes **4 bytes**
- 1 character takes **2 bytes**
2 Billion Something?

2 billion integers

2 billion quarters?
2 Billion Something?

2 billion integers

2 miles, or 3.2 km!

2 billion quarters?
Variables and Assignment

age = 3
Variables and Assignment

age = 3
name = "Smith"
rate = 0.06
Variables and Assignment

age = 3
name = "Smith"
rate = 0.06

literals
Variables and Assignment

age = 3
name = "Smith"
rate = 0.06

age = age * 2       # double the age
age = age + 1       # increment the age
Variables and Assignment

```python
age = 3
name = "Smith"
rate = 0.06

age = age * 2  # double the age
age = age + 1   # increment the age

name = name + " College"  # name will contain # "Smith College"
```
Variables and Assignment

age = 3
name = "Smith"
rate = 0.06

age = age * 2  # double the age
age = age + 1  # increment the age

name = name + " College"  # name will contain "Smith College"

In a programming language operators may have different meanings depending on the context
Variables and Assignment

age = 3
name = "Smith"
rate = 0.06

age = age * 2  # double the age
age = age + 1  # increment the age

name = name + " College"  # name will contain "Smith College"

Overloaded operators
Exercises

age = 3
name = "Smith"
rate = 0.06

age = age * rate  
name2 = "his + hers"
Exercises

age = 3
name = "Smith"
rate = 0.06

age = age * rate  # age will contain 0.18
name2 = "his + hers"  # name will contain "his + hers"
rate = name * rate  #
Exercises

age = 3
name = "Smith"
rate = 0.06

age = age * rate  # age will contain 0.18
name2 = "his + hers"  # name will contain "his + hers"
rate = name * rate  # TypeError: can't multiply
# sequence by 'float'

col = name + "College" * 2
#
Exercises

age = 3
name = "Smith"
rate = 0.06

age = age * rate  # age will contain 0.18
name = "his + hers"  # name will contain "his + hers"
rate = name * rate  # TypeError: can't multiply
# sequence by 'float'

col = name + "College" * 2
# this will depends on which
# operator is executed first
# operator priority: * / + -
Exercises

age = 3
name = "Smith"
rate = 0.06

age = age * rate  # age will contain 0.18
name = "his + hers"  # name will contain "his + hers"
rate = name * rate  # TypeError: can't multiply
    # sequence by 'float'

col = ( name + "College" ) * 2
    # "SmithCollegeSmithCollege"
Exercises

age = 3
name = "Smith"
rate = 0.06

age = age * rate  # age will contain 0.18
name = "his + hers"  # name will contain "his + hers"
rate = name * rate  # TypeError: can't multiply
                # sequence by 'float'

col = name + ( "College"  * 2 )
       # "SmithCollegeCollege"
Simultaneous Assignments

\[
\begin{align*}
& a, b, c = 10, 20, 30 & \quad \# a = 10, b = 20, c = 30 \\
& a, b = b, a & \quad \# a = 20, b = 10 \text{ (swap!)} \\
& a, b, c = 10, 20, 30 \\
& a, b, c = a+1, b*2, a-b & \quad \# a = 11, b = 40, c = -10
\end{align*}
\]
a, b, c = 10, 20, 30
# a = 10, b = 20, c = 30

trio = a, b, c  #
More...

```python
a, b, c = 10, 20, 30  # a = 10, b = 20, c = 30
trio = a, b, c        # trio contains (10, 20, 30)
x, y, z = trio         #
```
a, b, c = 10, 20, 30
# a = 10, b = 20, c = 30

trio = a, b, c
# trio = (10, 20, 30)

x, y, z = trio
# x = 10
# y = 20
# z = 30
Trick Question

a, b = 10, 20

a, b = b, a  # what happens?
The Programming Process

Variables

Definite Loops

Input
for <var> in <sequence>:
<body>
for <var> in <sequence>:
<body>

for can in [Is, Coca, Crush, Dr Pepper, Mist]:
  open( can )
  drink( can )
  throwAway( can )
for <var> in <sequence>:
<body>

for can in [Pepsi, Coca-Cola, Crush, Dr Pepper, 7UP, Sprite, Snapple, Mist]:
open( can )
drink( can )
throwAway( can )

Sequence

Many actions repeated, each group for each can
for `<var>` in `<sequence>`:
`<body>`

for `name` in [ "Alex", "Max", "Rui" ]:
print( "Hello", name + "!" )
print( "— — — — — — — — — — — " )
for `<var>` in `<sequence>`:
`<body>`

for `x` in `range(10)`:  
`print(x)`
http://docs.python.org/3.0/
2. Built-in Functions — Python 3.4.2 documentation
https://docs.python.org/3/library/functions.html

The Python interpreter has a number of functions and types built into it that ... The bytearray class is a mutable sequence of integers in the range $0 \leq x < 256$. 

2. Built-in Functions — Python 2.7.9 documentation
https://docs.python.org/2/library/functions.html

The argument must be in the range $[0..255]$, inclusive; ValueError will be raised if $i$ is outside that range. See also unichr(). classmethod(function)¶. Return a ...
Examples to Try Out:

```python
for x in range( ... ): # replace ... with
    print( x ) # range expression
      # below:

# range( 10 )
# range( 2, 10 )
# range( -5, 5 )
# range( 0, 10, 2 )
# range( 0, 10, 3 )
# range( 9, 0, -1 )
```
Exercise

Generate an equivalency table of temperatures in Fahrenheit and Celsius. 100 F should be on the first line, and -30F on the last line. Show only Fahrenheit temperatures that are multiples of 10.

Celsius = (Farhenheit - 32) * 5 / 9
The Programming Process

Variables

Definite Loops

Input
```python
name = input("What is your name?")
```

```

"Alex"
```

```
name
```
Demo Time!

```python
20 >>> c
30 >>> trio = a, b, c >>> trio
(10, 20, 30) >>> x, y, z = trio >>> x
10 >>> y 20 >>> z 30 >>> i, j = trio
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    i, j = trio
ValueError: too many values to unpack
```
review lab2. eval(input) vs input(eval)

Programming Practice: Example 1

Programming Practice: Example 2
\texttt{eval}(input(\ldots)) \ vs \ input(\texttt{eval}(\ldots))
\texttt{eval}(\texttt{input}(\ldots)) \text{ vs. } \texttt{input}(\texttt{eval}(\ldots))
\texttt{eval} (\textit{input}(\ldots)) \text{ vs. } \textit{input} (\texttt{eval}(\ldots))
\texttt{eval(input(\ldots))} vs \texttt{input(eval(\ldots))}

\begin{verbatim}
a = input( eval( "Enter a number" ) )
\end{verbatim}
\texttt{eval(input(\ldots) \ vs \ input( eval(\ldots) ) )}

\begin{align*}
\texttt{a} &= \texttt{input( eval( "Enter a number" ) )}
\end{align*}
\texttt{eval(input(...))} \ vs \ \texttt{input(eval(...))}

\begin{center}
\begin{itemize}
\item \texttt{a = input(eval("Enter a number")})
\end{itemize}
\end{center}

\textbf{PYTHON ERROR!}
\texttt{eval(\textit{input(\ldots)}) \textit{vs input( eval(\ldots))}}
\texttt{eval(input(...)) vs input(eval(...))}

\begin{verbatim}
a = eval(input("Enter a number"))
\end{verbatim}
\texttt{eval(\textit{input}(\ldots) \ )} vs \texttt{input( \texttt{eval}(\ldots) \ )}

\begin{align*}
a &= \texttt{eval( \texttt{input( "Enter a number" )} \ )}
\end{align*}
\texttt{eval}(\texttt{input}(\ldots)) \ vs \ \texttt{input}(\texttt{eval}(\ldots))

\begin{center}
\begin{verbatim}
a = eval( input( "Enter a number" ) )
\end{verbatim}
\end{center}

Enter a number 32
\texttt{eval(input(...))} vs \texttt{input( eval(...))}

\begin{align*}
a &= \texttt{eval(input("Enter a number") )}
\end{align*}
**eval**(input(...) ) vs input( eval(...) )

a = eval("32")

"32"

Enter a number 32
$\text{eval}(\text{input}(\ldots)~)~\text{vs}~\text{input}(\text{eval}(\ldots)~)$

$$a = \text{eval}(\text{"32"})$$

Enter a number 32
The Programming Process
Example 1
The Programming Process
Example 1
Problem 1

Get first name, last name, Id from student, and final grade, as a number (0-100).

Also known is class average, as a number (0-100).

Display student information in a box, and horizontal bar-graph of 2 grades.
Problem 1

First name?  Dominique
Last name?  Thiebaut
Id?          990123456
Final grade? 90

+--------------------------------------------------------+  +--------------------------------------------------------+
|Dominique Thiebaut                                         | 990123456 |
+--------------------------------------------------------+  +--------------------------------------------------------+

00...10...20...30...40...50...60...70...80...90...100

grade: #################################################################
class: #################################################################
Problem 2
(if time permits)

Write a python program that displays an 8x8 chessboard. Black cells should be 3x3 with #-signs in them, and white cells should be 3x3 with spaces inside.
Problem 2

Output

(and if you are ambitious, when you are done, make the program ask the user for the number of cells wanted, horizontally and vertically.)
Some Apps Written in Python

- Faculty Elections vote counting (Hare method)
- **DropBox**: https://www.dropbox.com/en/
- **Reddit**: https://www.reddit.com/
- (First version of) **Google** search engine