CS 106
Introduction to Computer Science I

07 / 02 / 2026

Instructor: Michael Eckmann

Today's Topics

- Questions / Comments?
- Other assignment operators
- Function terminology (parameter, argument, call, return)
- Nested if statements again
- and/or/not boolean operators examples
- Reserved words
- Random
- Loops
- While loops
- Write a guessing game

More assignment operators besides =

- +=, -=, *=, /=, %=

- Here's what they mean. Suppose we have:

  a=7

  a = a + 6 # is equivalent to: a += 6

- These assignment operators are used only if the variable on LHS (left hand side of the assignment) is also the first variable on the RHS.

More assignment operators besides =

- +=, -=, *=, /=, %=

More examples:

```
degrees -= 5
# subtract 5 from degrees and store this new value
# in degrees.

halve_me  /=  2
# divides the value in halve_me by 2 and stores
# this new value in itself */

fine *= 3   #   value in fine is tripled
```

Clarification of some function terminology

- function call
- Parameter / argument
- What a function returns
  - How to “capture/store” what a function returns

function terminology

- **None** – another type in Python which is used when no value is needed.

function terminology

- **Parameter** – functions can have 0 or more parameters. They are a way to allow data INTO a function, when it is called. The data we pass into the function parameters are called **arguments**.
- **Call**ing a function – invoking a function by giving its name in a statement in your program:
  - e.g.
  - print("Hey") # function call for print
  - height = input('enter your height in feet, please')
  - height = int(height)
- Note: A str is being passed in as an argument to print, input and int.

function terminology

- **What a function returns** – This is what the result of a method call gives.
- e.g.

```
print("Hey")              # None is returned

height = input('enter your height in feet, please')
                          # str is returned

height = int(height)    # an int is returned
```

function terminology

- Let's take a look at that projected homeruns and look at each line and use the function terminology to describe them

Nested ifs

- If statements can have other if statements nested within them.
- Pay attention to the indentation to tell which if the elif or else connect to.
- Let's look at a couple of examples.

Logical operators

- **and or not** are three operators that use bools as operands and result in a bool

- Truth table for **and**

False **and** False results in False
False **and** True results in False
True **and** False results in False
True **and** True results in True

Logical operators

- **and or not** are three operators that use bools as operands and result in a bool

- Truth table for **or**

False **or** False results in False
False **or** True results in True
True **or** False results in True
True **or** True results in True

Logical operators

- **and or not** are three operators that use bools as operands and result in a bool

- Truth table for **not**

not True results in False

not False results in True

Logical operators

- Let’s write some ifs using and and or

Let's look at the reserved words list

- Python has a list of reserved words which already have meaning and are not allowed to be used as variable names.
- We've used some of them already: True, False, if, elif, else, and, or, not
- Let's look at the complete list
- Variable names cannot be among the reserved words, and are only allowed to contain letters, digits, _ and they cannot start with a digit.

Random number generation

- To use random numbers in python we must import the random module like so:

```
import random
```

- Then we can call any of the functions defined in the random module by preceding their name with random. (random followed by a dot)
- e.g.

```
random.random()
# returns a random number in [0,1.0)
```

Random number generation

```
random.random()
# returns a random number in [0,1.0)
```

- That is, it will be a float in the range 0 to 1 (not including 1)
- Let's run it a few times and see what we get

Random number generation

```
random.randint(1,5)
# returns a random int among 1,2,3,4,or 5
```

- The first parameter of randint is the low end of the range (doesn't have to be 1)
- The second parameter of randint is the high end of the range (doesn't have to be 5)
- The function returns a random int in that range (inclusive on both ends)

While loop

- A while loop can be used to repeat lines of code as long as some condition (boolean expression) remains true.
- Syntax is

```
while ____:
    # line 1 of body of the loop
    # line 2 of body of the loop
    # …
    # last line of body of the loop
```

- ____ is a boolean expression (evaluates to True or False)

While loop

- Let's read and try to predict what this loop does.

```
count = 0
while count < 10:
    print(count)
    count += 1
```

While loop

- Let's read and try to predict what this loop does.

```
count = 0
while count > 10:
    print(count)
    count += 1
```

While loop

- Let's read and try to predict what this loop does.

```
count = 0
while count >= 0:
    print(count)
    count += 1
```

While loop

- Let's write a guessing game using
  - User input
    - We'll ask for the range they wish to guess among
    - We'll ask for their guess
  - Random number generation
  - While loop to allow repeated guesses if incorrect
  - Modify it to tell the user higher/lower

Time for lab 02

- Please work on lab 2 for the remainder of the class

Int and float

- There is no maximum int in python3 --- it is only restricted to available memory space to store it
- e.g.
- 21 ** 1000
- But float has limits
- e.g.
- 21 ** 1000.0

str methods/operations

- We already saw how to concatenate strings together using +
- Python has some powerful tools that allow us to easily do things with strings (like search them, make them all uppercase, find their length, etc.)
- Do help(str) as well as look at documentation for string methods on the web.
- Let's look at a sampling of the available str methods:

capitalize

endswith

lower

upper

find, rfind

replace

- Note: methods are a lot like functions except we precede them with a variable or a value (in this case a variable or value of type str)

str methods/operations

- Examples:

word = ‘science’

word1 = word.upper()

returns SCIENCE and assigns it to word1,

word remains unchanged

word2 = word.replace(‘s’,’$’)

returns $cience and assigns it to word2

str methods/operations

- in, not in, *, len(), min(), max(), [ ]

- Examples:

word = ‘science’

len(word) # returns 7
word[0]   # returns ‘s’
min(word) # returns ‘c’