CS 111: Program Design I

Lecture 6: slicing strings; if

Robert H. Sloan & Richard Warner
University of Illinois at Chicago
September 12, 2019

UIC

SNOWDEN, 4TH AM., ETC.

Glen Greenwald is the investigative journalist who first broke the story of Edward Snowden's collection of classified documents. Greenwald had a large number of those documents on his hard drive when he was meeting with Snowden before breaking the story. The US government has, as of about 2014 controversially taken the position that possession of such documents violates the Espionage Act (which can carry the death penalty).

Suppose (this did not happen) that the FBI legally seizes Greenwald's laptop. The FBI strongly suspects classified documents are on the hard drive, but it cannot read the files because the hard drive is encrypted. *Should the FBI be able, through a court order (a subpoena), to compel Greenwald to provide them with the encryption key?*

A. Yes **B. No**

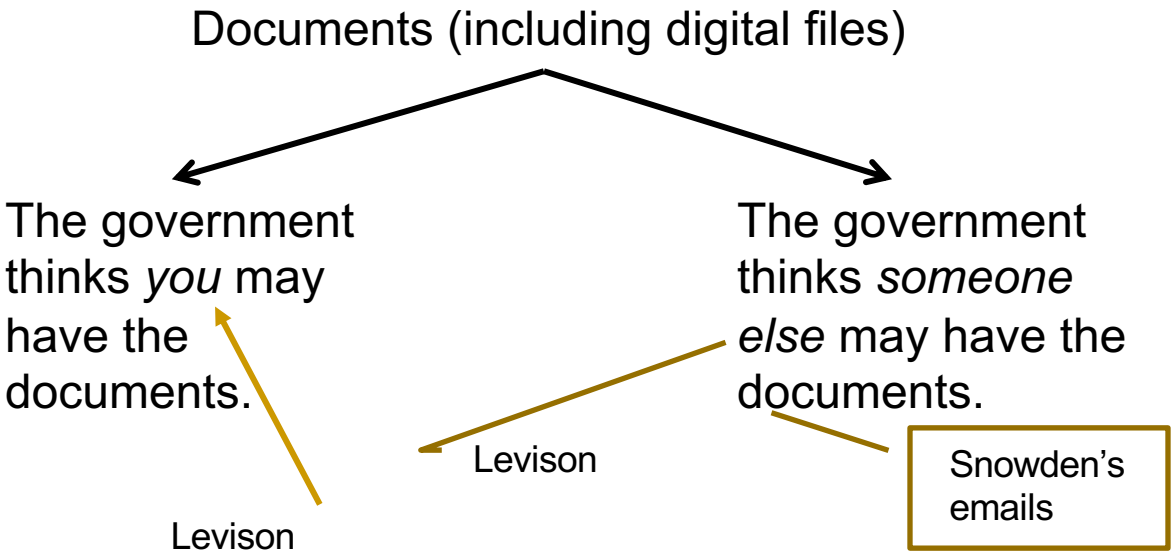
Lavabit and Snowden

- Lavabit was an encrypted webmail service between 2004 – 2013 owned and operated by Ladar Levinson.
- Edward Snowden used the service.
- “In 2013, the United States sought to obtain certain information about a target [Snowden] in a criminal investigation. The Government obtained court orders . . . requiring Lavabit to turn over particular information related to the target. When Lavabit and Levison failed to comply with those orders, the district court held them in contempt and imposed monetary sanctions.”

4th Not 5th Amendment

- The orders raise 4th not 5th Amendment issues.
- The 5th Amendment provides protection against *self*-incrimination.
- The orders direct a *third party*—Levison—to provide information.
- Levison can raise 4th Amendment but not 5th Amendment issues.

Our Diagram


Documents (including digital files)
The government thinks *you* may have the documents.
The government thinks *someone else* may have the documents.
Levison
Levison
Snowden's emails

What Did the Government Want?

- If the government installs the pen register without the encryption keys, it scoops up everything, and that everything would be encrypted with TLS with 100% of major email providers, much less Lavabit. So just tapping the “out” and “in” line with a device won’t get the government what it wants.
- But it would be relatively easy for an expert to write some software that pulls out the incoming and outgoing email headers in real time, because the Email provider has to have them to process every piece of email, and the likes of Gmail must have done this so that they can comply with the occasional pen-trap order. Levison surely could have done this.

The 4th Amendment

- Does the 4th Amendment apply?
- The 4th Amendment: “The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no Warrants shall issue, but upon probable cause, supported by Oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized.”

What It Means

- There is a zone of privacy—"secure in their persons, houses, papers, and effects"—that cannot be invaded without a warrant.
- The point:
  - to prevent the government from seeing too deeply into your life without a warrant.

The 3rd Party Doctrine

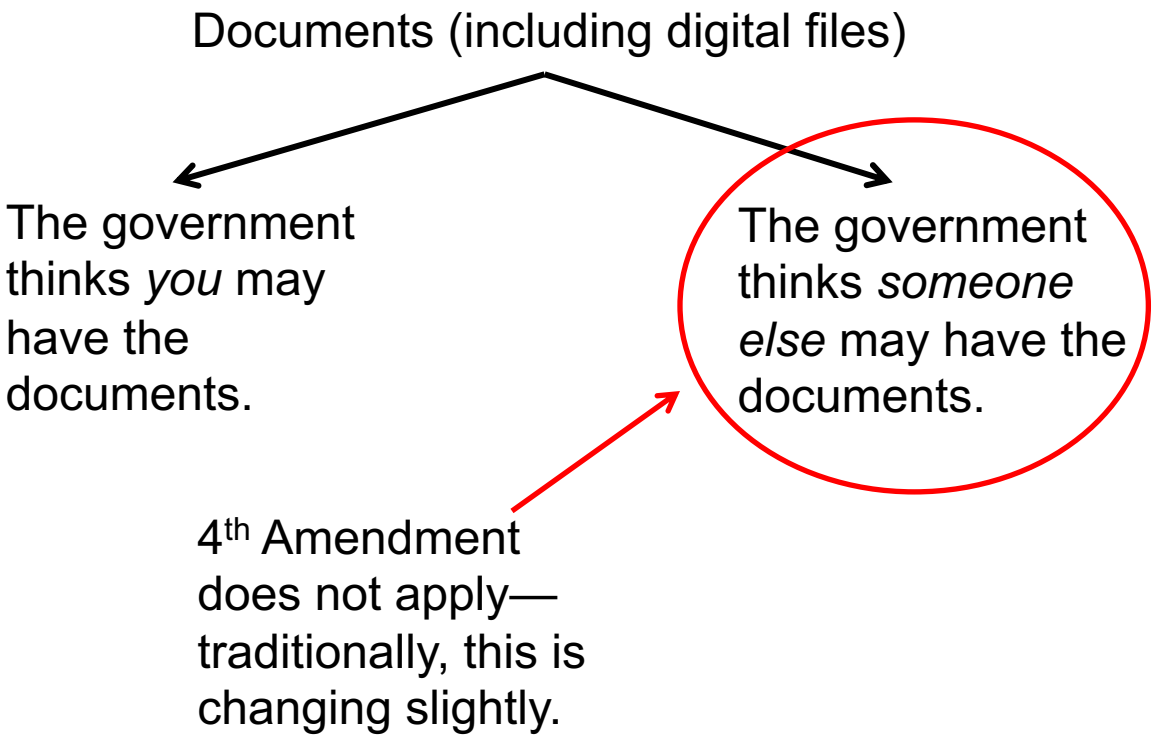
"The Fourth Amendment does not prohibit the obtaining of information revealed to a third party and conveyed by him to Government authorities, even if the information is revealed on the assumption that it will be used only for a limited purpose and the confidence placed in the third party will not be betrayed."

- United States v. Miller, 425 U.S. 435, 445 (1976).

Information Others Have About Us

- How much would I know about you if I examined all the information that you have stored online?
- Would you let me look at all of it?
- What about all of your emails? How much would I know about you if I examined all of them?
- If the government can see deeply enough into your life, shouldn't 4th Amendment apply?

The Consequence



STRING SLICING

Slicing: Getting part of a string

- Substring of string is called slice
- s[m:n] returns characters from index m (counting from 0 as always) up to *but not including* character n

```
cj =  'Chief Justice Roberts'
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
```

```
cj[0:5]  → 'Chief'
cj[6:10] → 'Just'
```

Slicing: omission = start/end

```
cj =   'Chief Justice Roberts'
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
```

```
cj[:13] → 'Chief Justice'
cj[14:] → 'Roberts'
```

and sort of silly one:

```
cj[:]   → 'Chief Justice Roberts'
```

Another slice example

```
                0 1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 1
                              0 1 2 3 4 5 6
>>> s = 'Register to vote!'

>>> s[9:11]
'to'
```

Slice can also use negative indices

```
                0 1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 1
                              0 1 2 3 4 5 6
>>> s = 'Register to vote!'
>>> s[len(s) - 1]
'!'
>>> s[-1]
'!'
>>> s[1:-1]
'egister to vote'
```

Trivia: Over end treated as end in slice

```
                0 1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 1
                              0 1 2 3 4 5 6
>>> s = 'Register to vote!'

In [1]: s[9:11]
Out[1]: 'to'
In [2]: s[12:20]
Out[2]: 'vote!'
```

Using strings: slicing (Recap & check)

```
                                0 1 2 3 4 5 6 7 8 9 10 11
>>> s = "AATGCCGTGCTT"          A A T G C C G T G C T T
```

```
>>> s[0:4]
    'AATG'
>>> s[3:7]
    'GCCG'
>>> s[1:]
    'ATGCCGTGCTT'
>>> s[:4]
    'AATG'
```

TGCTT ?

A. s[5]

B. s[7:11]

C. s[7:12]

D. s[7:len(s)]

E. Both C and D

Fancy slicing you'll never use

- You are allowed to give 3 indices, which are interpreted as start:end:step

```
cj = 'Chief Justice Roberts'
      01234567890123456789 0
cj[4:17:3]
```

Fancy slicing you'll never use

- You are allowed to give 3 indices, which are interpreted as start:end:step

```
cj = "Chief Justice Roberts"
      0123456789012345678 90
```

`cj[4:17:3]` → `'fui b'`

- Well, you'll never use it in practice. It does make nice exam questions

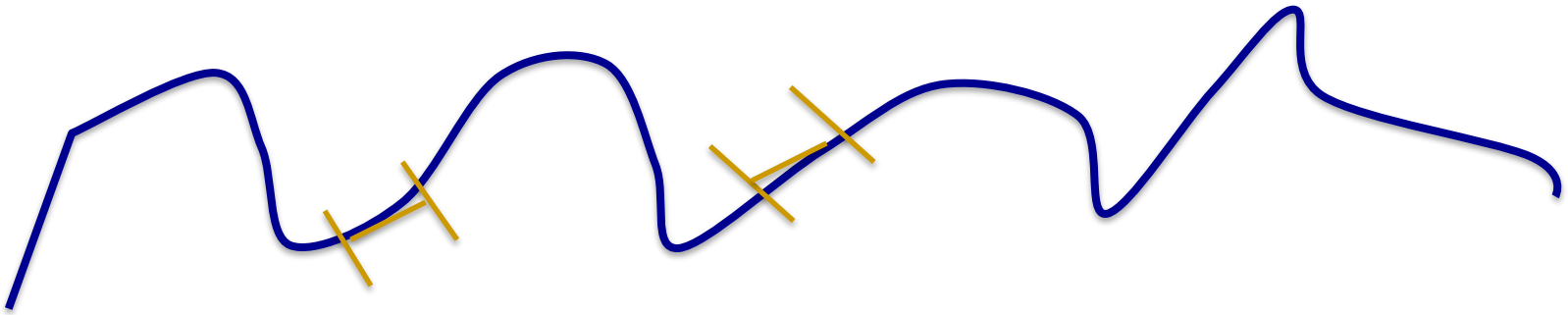
Fancy slicing you'll never use except for this one Python *idiom*

- Weird, yet standard, way to reverse a string s is s[::-1]
  - Give me the whole string, stepping 1 *backwards* each time!

```
In [1]: first_chief = 'Jay'
In [2]: first_chief[::-1]
Out[2]: 'yaJ'
```

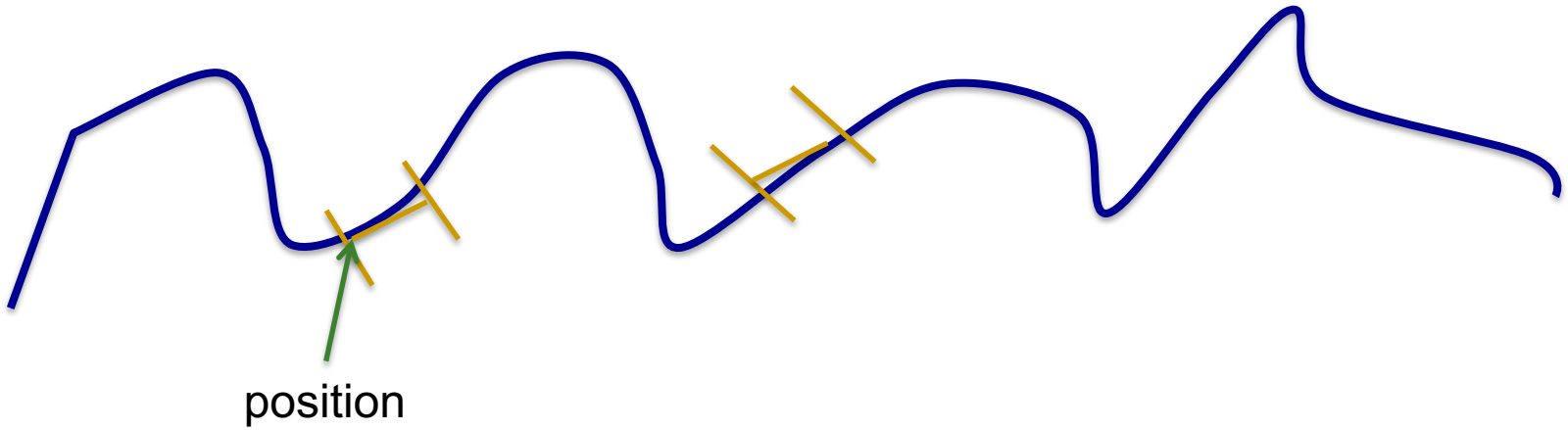
Finding strings in strings

- Suppose we want to find *second* occurrence of substring target in string s



Finding strings in strings

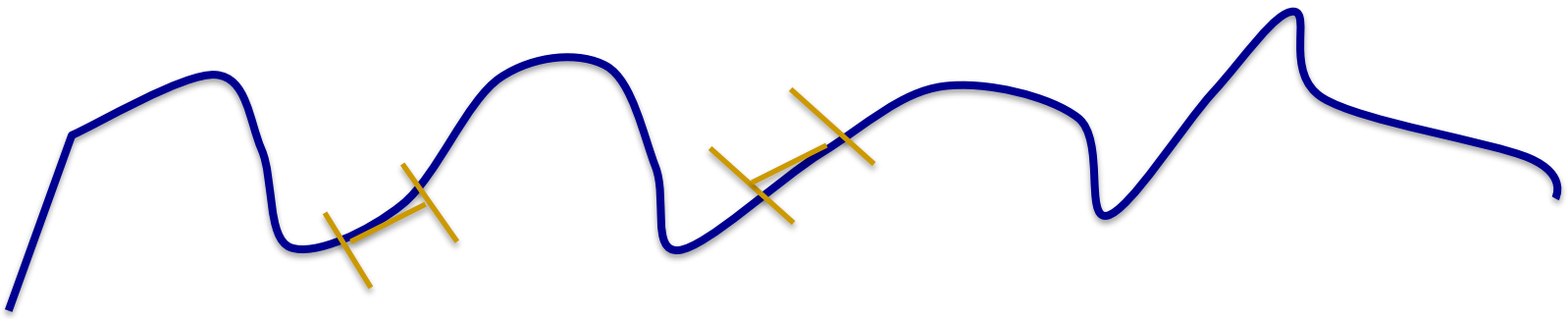
- Suppose we want to find *second* occurrence of substring target in string s



- `s.find(target)` gives position of *first* occurrence in s

Finding strings in strings

- Suppose we want to find *second* occurrence of substring target in string s



- `s.find(target)` gives position of *first* occurrence in s
- `s.find(target, start)` gives position of first occurrence of target *but now in slice* `s[start:]`

Strings: lots of other thing too!

- Strings have many useful built-into-Python methods besides .find(). Will cover in detail in next few weeks. One more example for now:
- Counting occurrence of letter (or any substring in a string): with .count() string method:
  - `sentence = 'Mary had a little lamb'`
  - `sentence.count('a')` → `4`
  - `sentence.count('had')` → `1`
  - `sentence.count('T')` → `0`

Some ~~people~~ objects just never change

- In Python, integers are immutable
  - Cannot assign to *integer object*
  - I.e., cannot write 1 = 0
    - Probably not a surprise

- Important: strings are immutable too!

```
first_justice = "Jay"
first_justice[0] = "H" Illegal; ERROR!
```

But Professor

- Can write

```
n = 1
n = 0
```

- Is that changing an integer?

But Professor

- Can write

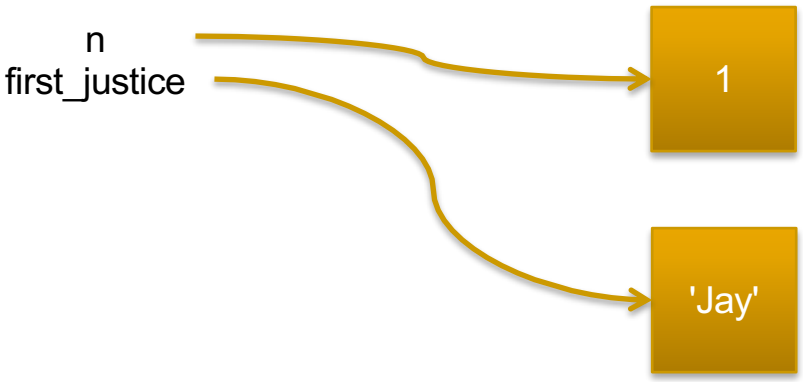
```
n = 1
```

```
n = 0
```

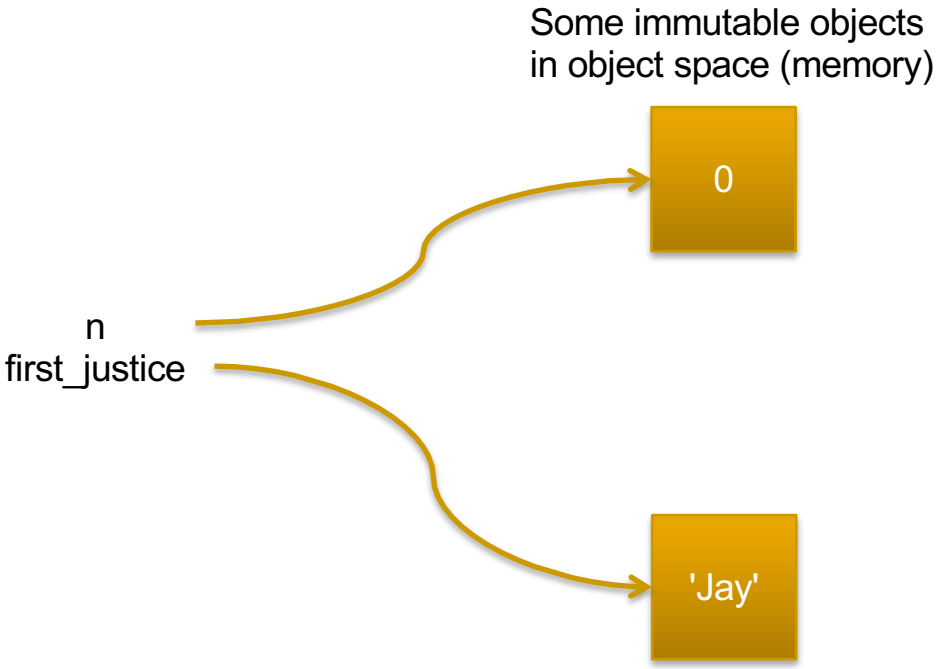
- Is that changing an integer?
- NO! Changing which object variable name n is assigned to
- Could reassign (entire object of) `first_justice` too

Pictures

Some immutable objects
in memory

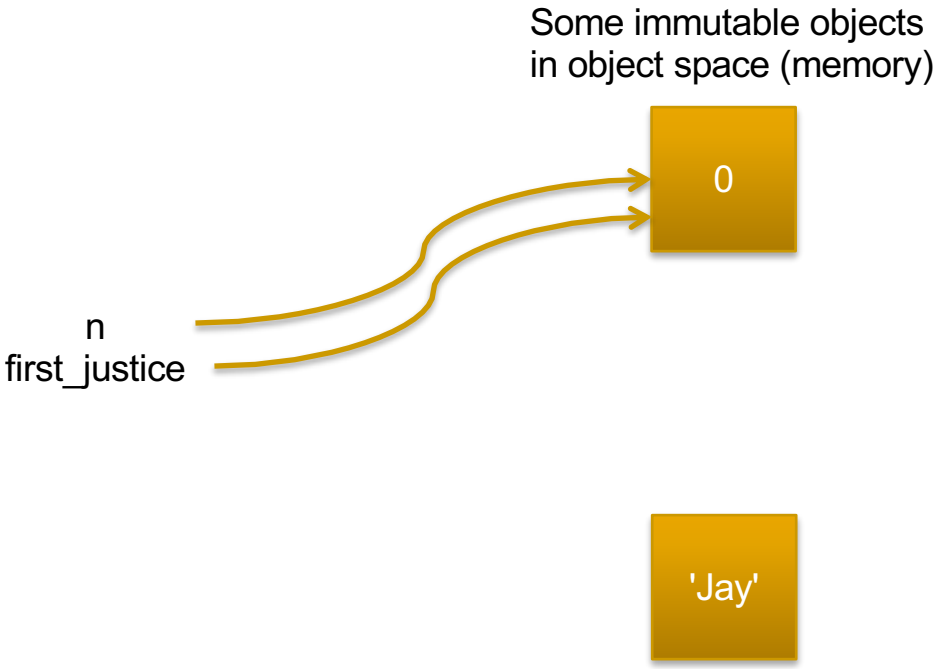


Pictures



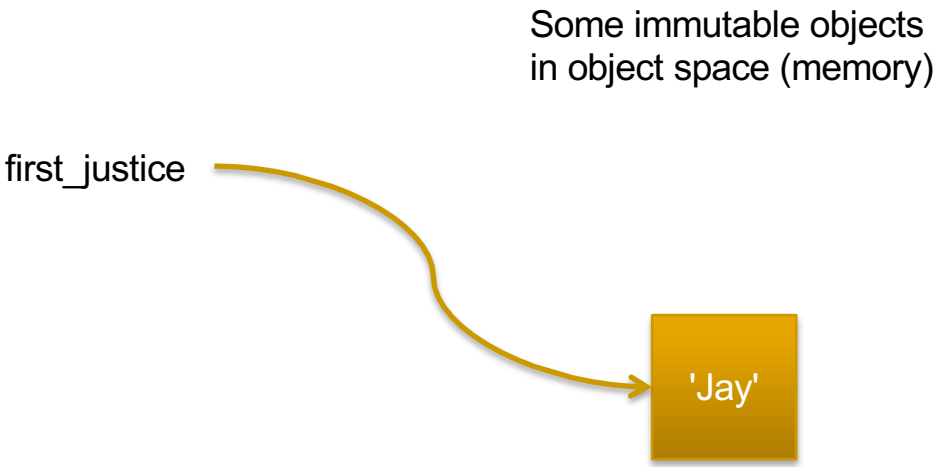
n = 0 doesn't change immutable integer 1,
just the assignment of a variable to an object

Pictures



first_justice = 0 doesn't change immutable string 'Jay',
just the assignment of variable to an object

Problem with first_justice[0] = "H" after first_justice = "Jay"



first_justice[0] = "H"
is illegal because we cannot change the ***contents***
of *any* of the yellow boxes. Those objects are ***immutable***

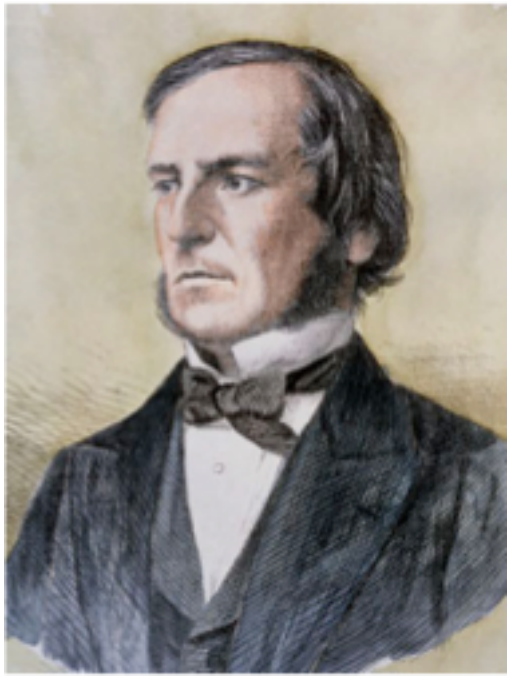
Coming attractions: Mutable types

- Numbers, Booleans, and strings all immutable
- Python has some mutable types
  - Two very important ones we will see:
  - Lists and dictionaries

GETTING READY FOR CAESAR'S INVASION NEXT WEEK

Types: Booleans

```
>>> 2 + 2 == 4
True
>>> 'cat' == 'cat'
True
>>> 2 == 4
False
>>> 2 != 4
True
```



Note operator is == (*two* = symbols)

String Boolean Operators

2 symmetric arguments:

==	equals
!=	is not equal to

2 arguments; order matters:

in	tells if lhs is substring of rhs
	e.g., "c" in 'cat' → true

Unary operator:

not	negates (flips True/False)

Just FYI: Later in the semester

Also

>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Which of the following will evaluate to True after first line?

```
x = "ATTACK AT DAWN"
```

A. `"AT" == x`
B. `"AT" in x`
C. `x == 6`
D. `x != 3`
E. More than one of the above

Why Booleans?

- To take one action **if** some condition holds.

- E.g., **if** *message m to encrypt has length 1* we can use function that just shifts 1 letter to encrypt it with Caesar
- otherwise we can do what we will learn to do for longer messages

THE ZEN OF LEARNING PYTHON & PROGRAMMING

Learning programming…

1. Expect it to be different!
2. *Don't feel you need to memorize it*
3. Immersion == Experimentation

Problem: Shifting a whole string

- Lab (with hints last lecture): shift/rotate one character: rotate()
- So, know how to shift strings of length 1
- And we saw == operator, so can tell if string s has length one:
  - len(s) == 1

- Notice: rotate() returned; *did not print*

return vs. print

- Must choose the one that does what we need on case-by-case basis to match specifications of job we need to do
- But: return often right choice, to make our work usable later in new/bigger/related/etc. problem

return vs. print example

```
def trip(x):
    return 3 * x

def works(input):
    return trip(input)
```

```
def times3(x):
    print (3 * x)

def whoops(input):
    return times3(input)
```

Which function correctly returns 3 times its input?

A. works()

B. whoops()

C. Both works() & whoops()

D. Neither of them

CONDITIONAL STATEMENTS

RUN CODE ONLY IF CERTAIN CONDITION IS MET

if statements

```
if <condition>:
    <body>
```

- `<condition>` is Boolean expression
  - E.g., `len(s) == 1`
- Code in `<body>` runs only if `<condition>` is `True`
- Colon and indentation of body mandatory!

Example

```
x = "ATTACK"

if len(x) > 1:
    print("x is longer than 1 letter!")
```

What will this print?

```
x = 5
if x == 3:
    print("Hi!")
print("Bye!")
```

A. Nothing
B. "Hi!"
C. "Bye!"
D. "Hi!" and "Bye!"

What will this print?

```
x = 5
if x == 5:
    print("Hi!")
print("Bye!")
```

A. Nothing
B. "Hi!"
C. "Bye!"
D. "Hi!" and "Bye!"