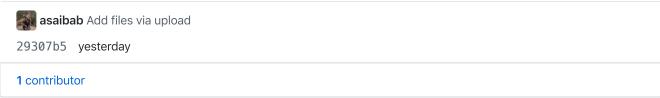
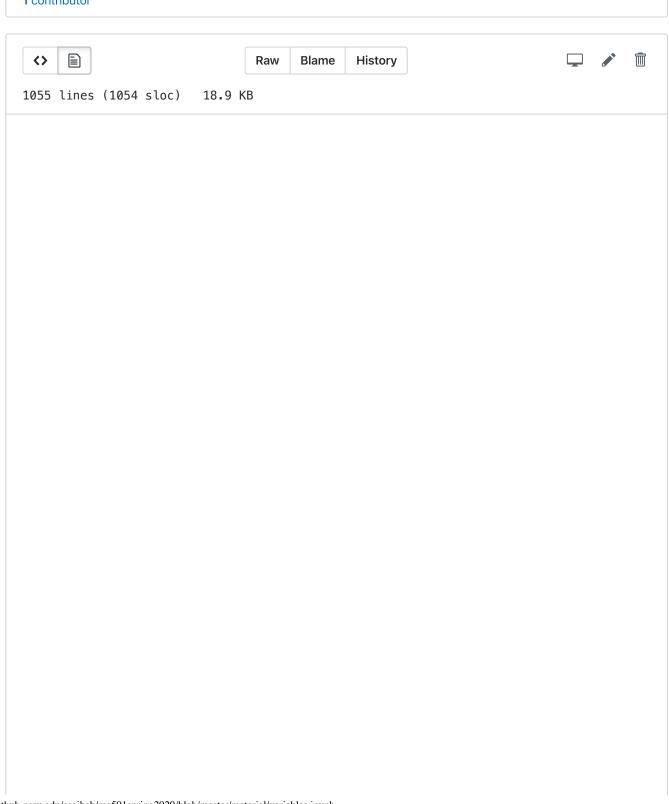
Branch: master ▼ Find file Copy path

ma591spring2020 / material / variables.ipynb





Ways to use Python

- 1. Command line (Use \$ python)
- 2. Self-contained scripts (Use \$ python <run file>)
- 3. Jupyter notebooks (we'll use it throughout this course)
- 4. Integrated development environment (IDE)

About this notebook

Ways to use it:

- 1. Read it on the class github page
- 2. Install <u>Jupyter notebook (https://jupyter.org/)</u> (e.g., Anaconda)
- 3. Use <u>colab (https://colab.research.google.com)</u>. No installation needed.

Bonus: to use the slideshow option, install Rise (https://github.com/damianavila/RISE)

A brief tour of the syntax

Use Python as a calculator

Whitespace

```
In [3]: # Function
    def f(a,b):
        return a + b;

# versus
    def g(a,b):
        return a + b;

print(f(2,3))
print(g(2,3))
```

Two ways to comment

- 1. Using # for inline quotations
- 2. Triple quotes """ "or block quotations

```
In [4]: # Comment 1
y = 1 # Also a comment

"""

This is a multiline
comment. Useful for documenting functions.
"""
```

Semicolons are optional

```
In [5]: x = 1
y = 1;

# Useful for concatenating statements
x = 1; y = 1
```

Print statement

In Python 2, print statements used to look like

```
print 'The value of x is ', 5.0
```

In Python 3, print statements are functions

```
In [6]: print('The value of x is ', 5.0)
The value of x is 5.0
```

Warning 1: Indentation

Message: Indentation is important for readability.

What you cannot do is

Warning 2: Counting

Message: Counting starts at zero.

```
In [8]: a = [1, 2, 3] # This is actually a list
    print(a[0]) # First element
    print(a[-1]) # last element; like a(end) in MATLAB

1
3
```

Variables and Operations

Built-in Variable types

Туре	Example	Description	
int	x = 1	Integer	
float	x = 1.	double precision	
complex	x = 1 + 2j	Complex numbers	
bool	x = True; y = False	Boolean	
str	x = 'abc'	String	

Use 'type()' for data type

Arithmetic operations

Operator	Name	Details
a + b	Addition	
a - b	Subtraction	
a *b	Product	
a/b	Division	Casts the result as double
a//b	Floor division	[a/b]
a % b	Modulus	Gives remainder
a ** b	Exponentiation	
-a	Negation	

Examples

```
In [10]: a = 3
b = 5
c = 5.

print(a + b, a*b, a/b, a//b)
print(a+c, a*c, a/c)

8 15 0.6 0
8.0 15.0 0.6
```

Another difference between Python2 and Python3: In Python2 a/b would have been `0', because the result has to be an integer.

More ways to print

Assignment operations

```
In [14]: a = 5
a += 2 # Same as a = a + 2
print(a)
7
```

More such operations

Operation	Same as	
a -= b	a = a - b	
a *= b	a = a * b	
a /= b	a = a/b	
a //b	a = a//b	
a %= b	a = a %b	
a **= b	a = a**b	

Comparison operations

Operation	Description	
a == b	checks a equal to b	
a != b	not equal	
a < b	less than	
a > b	greater than	
a <= b	less than equal to	
a >=	greater than equal to	

Examples of comparison operations

True False

Be careful of floating point comparisons

Complex numbers

String manipulation

```
In [6]: str1 = 'And now for something ... '
    str2 = "completely different"

    print("Length of string 1 is ", len(str1))

#Capitalize a string
    print("Capitalized string:\n", str2.capitalize())

Length of string 1 is 26
Capitalized string:
    Completely different
```

Other string manipulations

```
In [20]: #Concatenate strings
    print(str1 + str2)

#Repeat strings
    print(3*str1)

# Print the third character in the string
    print(str1[2])

And now for something ... completely different
    And now for something ... And now for something ... And now
    for something ...
```

d

Converting from one format to another

Boolean operations

Bitwise operations

Operation	Alternative	Description	
a & b	a and b	Bitwise and	
а	b	a or b	Bitwise or
~a	not(a)	Bitwise not	

```
In [25]: (x > 2) & (x > 7)
Out[25]: False
```