Branch: master

Find file Copy path

ma591spring2020 / material / variables.ipynb

asaibab Add files via upload

29307b5 yesterday

1 contributor

Raw Blame History

1055 lines (1054 sloc) 18.9 KB

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 Jupyter notebook (https://jupyter.org/) (e.g., Anaconda)
3. Use colab (https://colab.research.google.com). 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

In [2]:
```
1+1
```

Out[2]: 2

In [2]:
```
x = 5
x*3
```

Out[2]: 15

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))
```

```
5
5
```

Two ways to comment

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

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

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

Out[4]:
```
'\nThis is a multiline\ncomment. Useful for documenting func
tions.\n'
```

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.

In [7]:
```
# If-else
if 4 > 3:
        print("It is true!")     #The spacing can be arbitra
ry as long as it is consistent
else:
    print("It is not true.")
    print("false")
```

```
It is true!
```

What you cannot do is

```
if 4 > 3:
        print("It is true!")
else:
    print("It is not true.")
        print("false")  #Inconsistent font spacing
```

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

Type	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

In [5]:
```
x = 1
y = 1.
z = 1 + 2j
b = True
s = 'abc'
print(type(y), type(z), type(b), type(x), type(s))
```

```
<class 'float'> <class 'complex'> <class 'bool'> <class 'in
t'> <class 'str'>
```

In [ ]:
```

```

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	$\lfloor a / b \rfloor$
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

In [1]:
```
x = 23.56
y = 5e-6
print('The value of x is ', x, ' and the value of y is ', y)
```

```
The value of x is  23.56  and the value of y is  5e-06
```

In [2]:
```
print('The value of x is {0} and the value of y is {1}'.form
at(x,y) )
```

```
The value of x is 23.56 and the value of y is 5e-06
```

In [12]:
```
print('The value of x is %g and value of y is %g' %(x, y))
```

```
The value of x is 23.56 and value of y is 5e-06
```

In [13]:
```
print('The value of x is %2.3f and value of y is %.15f' %(x,
y))
```

```
The value of x is 23.560 and value of y is 0.000005000000000
```

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

In [15]:
```
x = 5; y = 3

print(x == y)
print(x != y)
print(x > y)
print(x <= y)
```

```
False
True
True
False
```

Be careful of floating point comparisons

In [16]:
```
print(0.1 + 0.2 == 0.3)
```

```
False
```

In [3]:
```
print(abs(0.1+0.2 -0.3))
```

```
5.551115123125783e-17
```

Complex numbers

In [17]:
```
x = 2 + 3j

print('The real part is %g and imaginary part is %g' %\
      (x.real, x.imag ))
```

```
The real part is 2 and imaginary part is 3
```

In [18]:
```
print('The conjugate is ', x.conjugate())
print('The magnitude is ', abs(x))
```

```
The conjugate is  (2-3j)
The magnitude is  3.605551275463989
```

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

In [21]:
```
print(float(1))
print(int(23.6))
print(str(23.6))
print(complex(23.6))
print(float(str(23.6)))
```

```
1.0
23
23.6
(23.6+0j)
23.6
```

Boolean operations

In [22]:
```
x = 5
(x > 2) and (x > 7)
```

Out[22]: False

In [23]:
```
(x > 2) or (x > 7)
```

Out[23]: True

In [24]:
```
not(x > 2)
```

Out[24]: False

Bitwise operations

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

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

Out[25]: False