CSCI 303

Introduction to Data Science

2 - Python Basics

Preview

In [1]:
```python
n = 1024
logn = 0
while n > 1:
    logn = logn + 1
    n = n / 2
print(logn)
```

```
10
```

About Python

- Created c. 1991 (by Guido van Rossum)
- General purpose, interpreted programming language
- Multiparadigm (OO, functional, procedural)
- Cross-platform and open source
- Widely used in data science:
  - Easy to learn and use
  - Huge library, including interfaces to just about everything

Python 2 vs 3

- Python 2
  - Still very popular
  - End of life in 2020
  - Older data science texts preferred 2.7 due to package availability
- Python 3
  - Recommended for anyone starting today
  - Has caught up with 2.7 in package availability
  - **Used in this class** - use `python3` on jupyterhub

Usage

- batch - programs or "scripts"

- interactive - *REPL** shells
    - `python3` interactive mode
    - IDLE
    - many others
- Jupyter (IPython) notebook

**REPL = Read - Eval - Print Loop*

Getting Started

Highly recommended:

https://docs.python.org/3/tutorial/index.html (https://docs.python.org/3/tutorial/index.html)

Everything is an Object

You can get help on pretty much any object with `help(obj)` :

In [2]:
```
help('math')
```

```
Help on module math:

NAME
    math

MODULE REFERENCE
    https://docs.python.org/3.6/library/math (https://docs.python.org/3.6/librar
y/math)

    The following documentation is automatically generated from the Python
    source files.  It may be incomplete, incorrect or include features that
    are considered implementation detail and may vary between Python
    implementations.  When in doubt, consult the module reference at the
    location listed above.

DESCRIPTION
    This module is always available.  It provides access to the
    mathematical functions defined by the C standard.
```

Jupyter Notebook Help

In the Jupyter notebook, there are some additional helpers.

- Type ? after any module, class name, or function to get help on it
- Tab completion after '.' operator for methods, properties
- Can even see source code with ??

In [3]:
```
import math
math?
```

In [4]:
```python
math.
```

```
  File "<ipython-input-4-6994845579ab>", line 1
    math.
         ^
SyntaxError: invalid syntax
```

In [ ]:
```python
def foo():
    '''
    This function prints a greeting.
    '''
    print('Hello, world!')
```

In [ ]:
```python
foo??
```

Scalar Base Types

- integers: `4` , `-20`
- floating point numbers: `1.234` , `1e-15`
- boolean: `True` , `False`
- `None`

Operators

operator	meaning
+, -, *	addition, subtraction, multiplication
/	real division
//	integer (floor) division
%	modulus
**	exponentiation

In [ ]:
```python
print((1 + 4) / 2)
print((1 + 4) // 2)
```

Variables

- Dynamic ("duck") typing
- Type not associated with variable, but with object
  - variables can be assigned different objects sequentially

In [ ]:
```python
a = 'Hello'
print('a =', a)
x = 7
y = 3.14
a = y * x
print('a =', a)
```

Control Flow

Consider the following code to generate Hailstone sequences:

In [ ]:
```python
n = 1292340923509522304582
while n != 1:
    print(n, end = ' ')
    if n % 2 == 0:
        n = n // 2
    else:
        n = n * 3 + 1
print(n)
```

While

While loop syntax:

```
while <condition>:
    <statements>
```

Notes:

- Condition is a Boolean expression
- Colon (:) ends the line (mostly for readability)
- Inside of loop is set off by mandatory *indentation*
- End of loop indicated by end of indentation

If-Else

If statement syntax:

```
if <condition 1>:
    <statements 1>
elif <condition 2>:  # optional, can repeat
    <statements 2>
else:                # optional
    <statements 3>
```

Note the indentation, the use of colons, and the Boolean conditional expressions.

For

Python has powerful `for` loops, but they work in conjunction with *iterable* objects, like sequences.

We'll study them after looking more into lists.

L02: Practice

1. Create a new Python 3 Notebook
2. Add a comment at the top of the cell with your name and the date
3. Solve this problem: Given an integer number `n`, print the number of times the digit `5` occurs in the number (without converting to a string). For example, n = 5555055555, should outpt 9 (hint: use modulo and floor division to work with one digit at a time).
4. Download the Notebook (.ipynb) file and submit to Canvas.