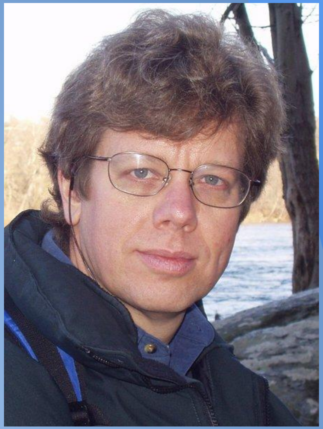
Introduction to Python

Break through the barrier of entry and begin programming in Python!

Guido van Rossum (BDFL)



What is Python?

- Interpreted Language
- Very Clear, Readable Syntax
- Extensive Libraries ("Batteries Included")
- Intuitive Object Orientation
- Strong Introspection Capabilities

Interpreted vs. Compiled

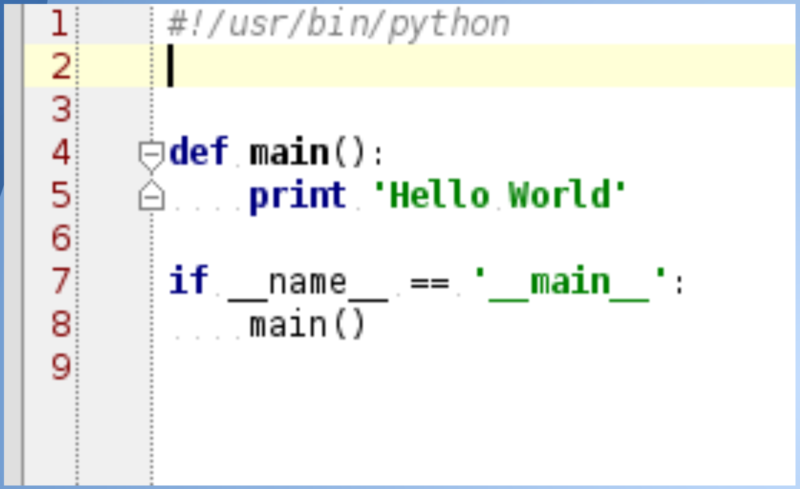
Interpreted Languages:

- Python - Perl - PHP - Ruby
- JavaScript - Lua

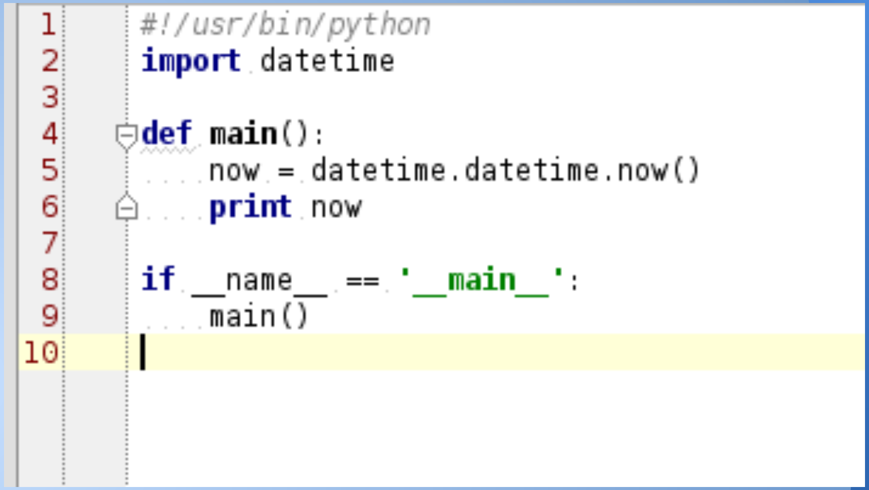
Compiled Languages:

- C/C++ - Java - Objective C
- Visual Basic - Pascal

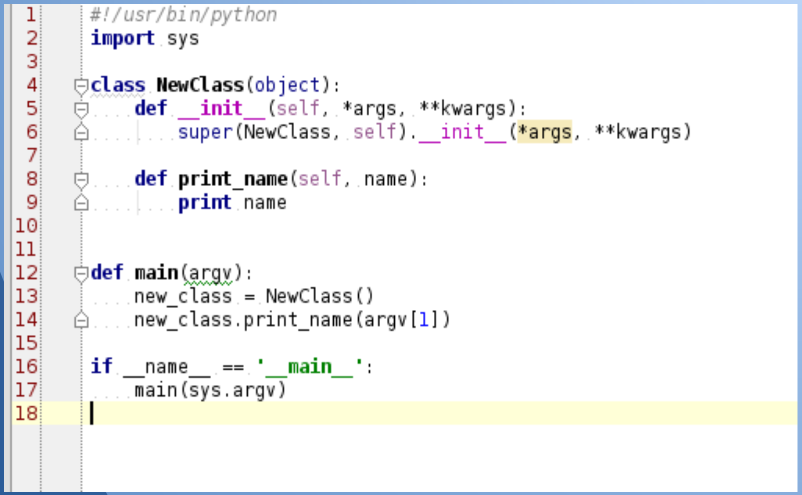
Very Clear, Readable Syntax



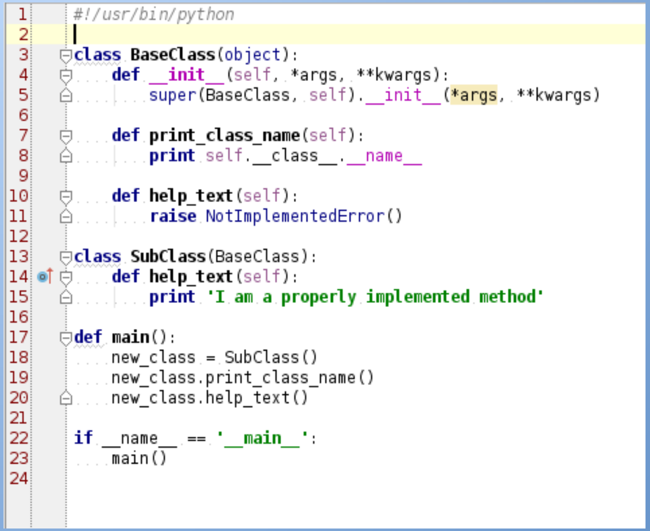
```python
#!/usr/bin/python


def main():
    print 'Hello World'

if __name__ == '__main__':
    main()

```



```python
#!/usr/bin/python
import datetime

def main():
    now = datetime.datetime.now()
    print now

if __name__ == '__main__':
    main()

```



```python
#!/usr/bin/python
import sys

class NewClass(object):
    def __init__(self, *args, **kwargs):
        super(NewClass, self).__init__(*args, **kwargs)

    def print_name(self, name):
        print name


def main(argv):
    new_class = NewClass()
    new_class.print_name(argv[1])

if __name__ == '__main__':
    main(sys.argv)

```

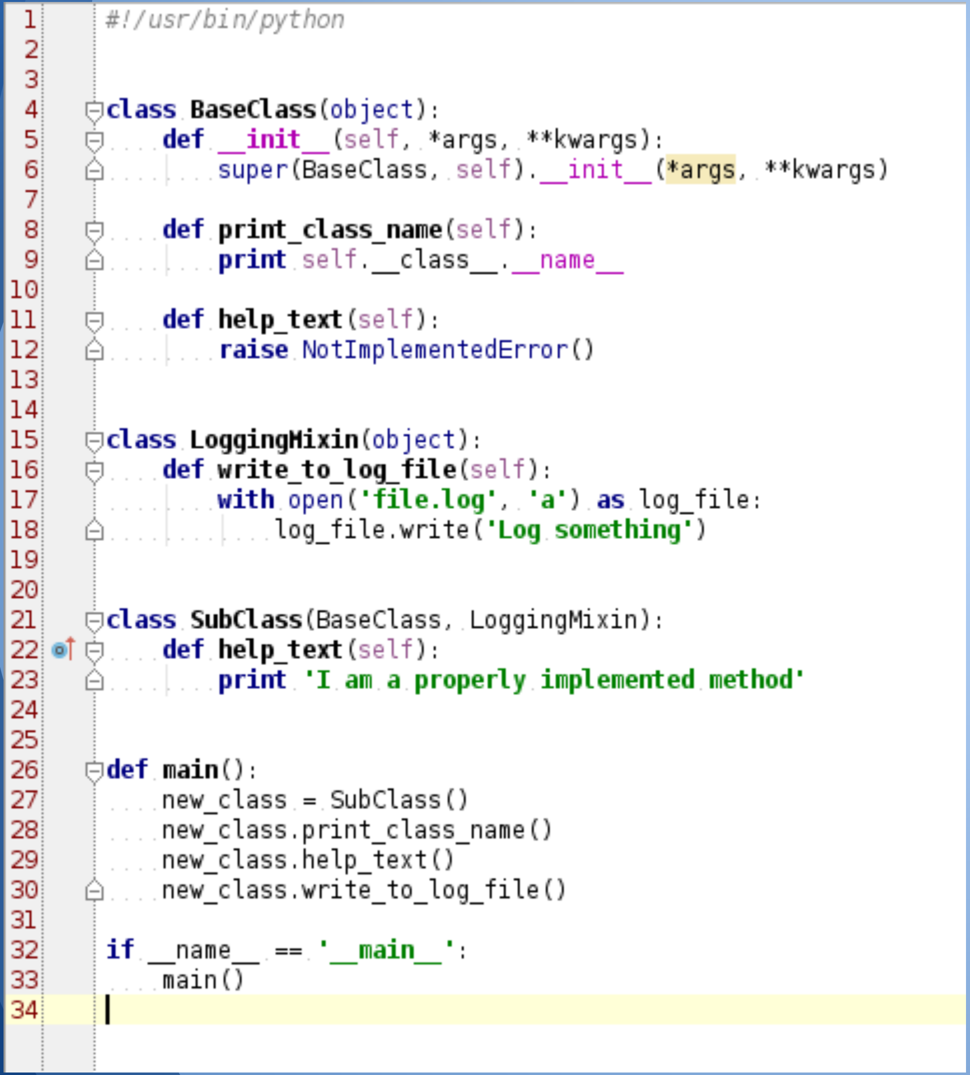


```python
#!/usr/bin/python

class BaseClass(object):
    def __init__(self, *args, **kwargs):
        super(BaseClass, self).__init__(*args, **kwargs)

    def print_class_name(self):
        print self.__class__.__name__

    def help_text(self):
        raise NotImplementedError()

class SubClass(BaseClass):
    def help_text(self):
        print 'I am a properly implemented method'

def main():
    new_class = SubClass()
    new_class.print_class_name()
    new_class.help_text()

if __name__ == '__main__':
    main()

```

Extensive Libraries

- datetime: date/time manipulation
- pycrypto: encryption algorithms
- psychopg2: PostgreSQL database interfacing
- math: mathematical functions
- os.path: file/path name manipulation
- csv: reading, writing and parsing CSV files
- threading: high level threading interface
- cProfile: built-in profiling tool
- subprocess: subprocess management
- Much, much more

Intuitive Object Orientation



```python
#!/usr/bin/python


class BaseClass(object):
    def __init__(self, *args, **kwargs):
        super(BaseClass, self).__init__(*args, **kwargs)

    def print_class_name(self):
        print self.__class__.__name__

    def help_text(self):
        raise NotImplementedError()


class LoggingMixin(object):
    def write_to_log_file(self):
        with open('file.log', 'a') as log_file:
            log_file.write('Log something')


class SubClass(BaseClass, LoggingMixin):
    def help_text(self):
        print 'I am a properly implemented method'


def main():
    new_class = SubClass()
    new_class.print_class_name()
    new_class.help_text()
    new_class.write_to_log_file()

if __name__ == '__main__':
    main()

```

Let's look at some code!

Feel free to access my Github repo to get some example Python and Django code.

https://github.com/ricomoss/python_examples