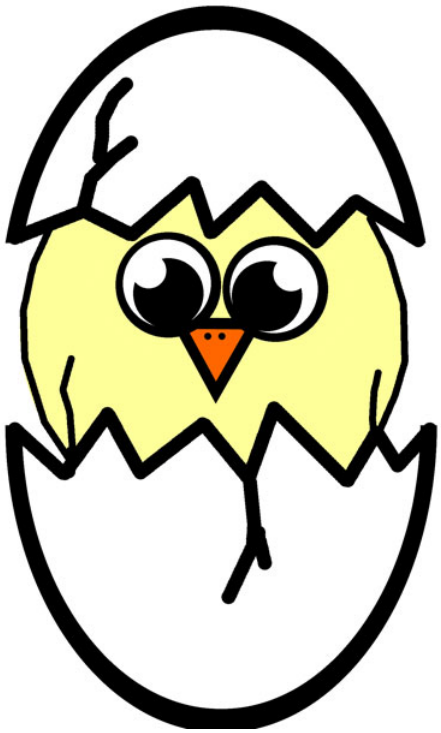
H a t c h - A Python Preprocessor

Addison Boyer

Natural Language Processing
Spring 2020



1 Getting Started

All Hatch statements begin with a special Hatch comment identifier (**#!**). If the Hatch comment identifier is not present, the line will be treated as interpretable Python code. All lines beginning with **#!** will "hatch" into interpretable Python code, given the correct Hatch syntax.

2 Keywords

1. **class** - Used to define a Python class in Hatch.
2. **get** - Used to define an attributes getter/getters in Hatch.
3. **set** - Used to define an attributes setter/setters in Hatch.
4. **str** - Used to define a classes toString() method in Hatch.
5. **hatch()** - Used to exit a Hatch interactive shell.

3 The Hatch Egg

The Hatch egg is where parameter and attribute names are passed. An egg consists of a comma delimited list of variable names surrounded by parenthesis. An empty hatch egg will result in the following error: *Empty egg to be hatched, aborting..*

4 Syntax

1. **Class Declarations**

   #! class Person = (name, age, ...)

2. **Getter Declarations**

   #! get = (name, age, ...)

3. **Setter Declarations**

   #! set = (name, age, ...)

4. **toString() Declarations**

   #! str = (name, age, ...)

5. **Variable Declarations**

   #! name = 'Addison'

   #! age = 20

   #! age = 20.0

5 Hello Hatch

```
# HelloHatch.Hatch
import sys

#! class HelloHatch = (hello, hatch)
   #! get = (hello, hatch)
   #! set = (hello, hatch)
   #! str = (hello, hatch)

def main(argv):

   hello_hatch = HelloHatch("Hello", "Hatch!")
   print(hello_hatch)

if(__name__ == "__main__"):
   main(sys.argv[1:])
```

make -B
./interpreter.out HelloHatch.Hatch > HelloHatch.py

```
# HelloHatch.py
import sys


class HelloHatch(object):
   def __init__(self,hello,hatch):
      self.hello = hello
      self.hatch = hatch


   def get_hello(self):
      return self.hello
   def get_hatch(self):
      return self.hatch


   def set_hello(self,hello):
      self.hello = hello
   def set_hatch(self,hatch):
      self.hatch = hatch


   def __str__(self):
      return str(self.hello) + ' ' + str(self.hatch)


def main(argv):

   hello_hatch = HelloHatch("Hello", "Hatch!")
   print(hello_hatch)

if(__name__ == "__main__"):
   main(sys.argv[1:])
```

python3 HelloHatch.py

Hello Hatch!