Hatch - A Python Preprocessor

Addison Boyer

Natural Language Processing

Spring 2020

1 Getting Started

All Hatch statements begin with the special Hatch comment identifier (**#!**). If the preprocessor 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 class in hatch.

2. **get**

   Used to define getter(s) in hatch.

3. **set**

   Used to define setter(s) in hatch.

4. **str**

   Used to define a toString() 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 into. An empty hatch egg will result in the following error: *Empty egg to be hatched, aborting..*

4 Hatch Syntax

#! class Person = (name, age)

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!
```