An Exploratory Study on the Predominant Programming Paradigms in Python Code

Robert Dyer and Jigyasa Chauhan
Increasing Prevalence of Multi-Paradigm Languages
Increasing Prevalence of Multi-Paradigm Languages
Increasing Prevalence of Multi-Paradigm Languages
Motivating Example

class MyCounter:
    x = 1

def val(self):
    def wrapper():
        return self.x
    return wrapper

def __iter__(self):
    self.x = 1
    return self

def __next__(self):
    if self.x > 50:
        raise StopIteration
    tmp = self.x
    self.x += 1
    return tmp

print([y for y in MyCounter() if y % 2 == 0])
```python
class MyCounter:
    x = 1

    def val(self):
        def wrapper():
            return self.x
        return wrapper

    def __iter__(self):
        self.x = 1
        return self

    def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
```
class MyCounter:
    x = 1

    def val(self):
        def wrapper():
            return self.x
        return wrapper

    def __iter__(self):
        self.x = 1
        return self

    def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
class MyCounter:
    x = 1

def val(self):
    def wrapper():
        return self.x
    return wrapper

def __iter__(self):
    self.x = 1
    return self

def __next__(self):
    if self.x > 50:
        raise StopIteration
    tmp = self.x
    self.x += 1
    return tmp

print([y for y in MyCounter() if y % 2 == 0])
class MyCounter:
    x = 1

def val(self):
    def wrapper():
        return self.x
    return wrapper

def __iter__(self):
    self.x = 1
    return self

def __next__(self):
    if self.x > 50:
        raise StopIteration
    tmp = self.x
    self.x += 1
    return tmp

print([y for y in MyCounter() if y % 2 == 0])
class MyCounter:
    x = 1

    def val(self):
        def wrapper():
            return self.x
        return wrapper

    def __iter__(self):
        self.x = 1
        return self

    def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
class MyCounter:
    x = 1

    def val(self):
        def wrapper():
            return self.x
        return wrapper

    def __iter__(self):
        self.x = 1
        return self

    def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
class MyCounter:
    x = 1

def val(self):
    def wrapper():
        return self.x
    return wrapper

def __iter__(self):
    self.x = 1
    return self

def __next__(self):
    if self.x > 50:
        raise StopIteration
    tmp = self.x
    self.x += 1
    return tmp

print([y for y in MyCounter() if y % 2 == 0])
class MyCounter:
    x = 1

    def val(self):
        def wrapper():
            return self.x
        return wrapper

    def __iter__(self):
        self.x = 1
        return self

    def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
class MyCounter:
    x = 1

def val(self):
    def wrapper():
        return self.x
    return wrapper

def __iter__(self):
    self.x = 1
    return self

def __next__(self):
    if self.x > 50:
        raise StopIteration
    tmp = self.x
    self.x += 1
    return tmp

print([y for y in MyCounter() if y % 2 == 0])
class MyCounter:
    x = 1

def val(self):
    def wrapper():
        return self.x
    return wrapper

def __iter__(self):
    self.x = 1
    return self

def __next__(self):
    if self.x > 50:
        raise StopIteration
    tmp = self.x
    self.x += 1
    return tmp

print([y for y in MyCounter() if y % 2 == 0])
RQ1  What is the distribution of predominant paradigms for Python projects on GitHub?

RQ2  What are the most and least used features for some programming paradigms?

RQ3  Are project size and predominant paradigm related?

RQ4  How does predominant paradigm use change over time?
RQ1  What is the distribution of predominant paradigms for Python projects on GitHub?

RQ2  What are the most and least used features for some programming paradigms?

- method declarations: 2,575,397
- class declarations: 1,738,668

RQ3  Are project size and predominant paradigm related?

RQ4  How does predominant paradigm use change over time?
RQ1  What is the distribution of predominant paradigms for Python projects on GitHub?

RQ2  What are the most and least used features for some programming paradigms? method declarations 2,575,397
class declarations 1,738,668
built-in functions (functools/itertools) 990,333
array comprehensions 729,309

RQ3  Are project size and predominant paradigm related?

RQ4  How does predominant paradigm use change over time?
Research Questions

RQ1  What is the distribution of predominant paradigms for Python projects on GitHub?

RQ2  What are the most and least used features for some programming paradigms?

<table>
<thead>
<tr>
<th>Feature</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>method declarations</td>
<td>2,575,397</td>
</tr>
<tr>
<td>class declarations</td>
<td>1,738,668</td>
</tr>
<tr>
<td>built-in functions (functools/itertools)</td>
<td>990,333</td>
</tr>
<tr>
<td>array comprehensions</td>
<td>729,309</td>
</tr>
</tbody>
</table>

RQ3  Are project size and predominant paradigm related?

RQ4  How does predominant paradigm use change over time?

(it doesn’t)
<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>101,648</td>
</tr>
<tr>
<td>Revision (with a Python file)</td>
<td>15,254,331</td>
</tr>
<tr>
<td>Python Files</td>
<td>7,758,882</td>
</tr>
<tr>
<td>(main branch only)</td>
<td>3,658,391</td>
</tr>
<tr>
<td>Python File Snapshots</td>
<td>68,787,597</td>
</tr>
<tr>
<td>ASTs</td>
<td>105,907,774,611</td>
</tr>
</tbody>
</table>
Approach Overview

Phase 1: Manual Classification

Phase 2: Automated Classification
Phase 1: Manual Classification

Phase 2: Automated Classification

Sample 102 files (out of 98,537)
Phase 1: Manual Classification

Phase 2: Automated Classification

Sample 102 files (out of 98,537)

3 raters: 0.759 kappa ("good" agreement)
Phase 1: Manual Classification

Phase 2: Automated Classification
class MyCounter:
    x = 1

    def val(self):
        def wrapper():
            return self.x
        return wrapper

    def __iter__(self):
        self.x = 1
        return self

    def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
```python
class MyCounter:
    x = 1

def val(self):
    def wrapper():
        return self.x
    return wrapper

def __iter__(self):
    self.x = 1
    return self

def __next__(self):
    if self.x > 50:
        raise StopIteration
    tmp = self.x
    self.x += 1
    return tmp

print([y for y in MyCounter() if y % 2 == 0])
```
```python
class MyCounter:
    x = 1

    def val(self):
        def wrapper():
            return self.x
        return wrapper

    def __iter__(self):
        self.x = 1
        return self

    def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
```
```python
class MyCounter:
    x = 1

    def val(self):
        def wrapper():
            return self.x
        return wrapper

    def __iter__(self):
        self.x = 1
        return self

    def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
```

class MyCounter:
    
x = 1
    
def val(self):
        def wrapper():
            return self.x
        return wrapper
    
def __iter__(self):
        self.x = 1
        return self
    
def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
```python
class MyCounter:  # func oo
    x = 1  # oo

    def val(self):  # oo
        def wrapper():  # oo proc
            return self.x  # oo proc
        return wrapper  # func oo

    def __iter__(self):  # func oo
        self.x = 1  # oo
        return self  # oo

    def __next__(self):  # func oo
        if self.x > 50:  # oo
            raise StopIteration  # oo
        tmp = self.x  # oo
        self.x += 1  # oo
        return tmp  # oo

print([y for y in MyCounter() if y % 2 == 0])  # func oo proc
```
```python
class MyCounter:
    x = 1

    def val(self):
        def wrapper():
            return self.x
        return wrapper

    def __iter__(self):
        self.x = 1
        return self

    def __next__(self):
        if self.x > 50:
            raise StopIteration
        tmp = self.x
        self.x += 1
        return tmp

print([y for y in MyCounter() if y % 2 == 0])
```
class MyCounter:  # func oo
    x = 1  # oo
    imp

def val(self):  # oo
    def wrapper():  # oo proc
        return self.x  # oo proc
    return wrapper  # func oo

def __iter__(self):  # func oo
    self.x = 1  # oo
    return self  # oo

def __next__(self):  # func oo
    if self.x > 50:  # oo
        raise StopIteration # oo
    tmp = self.x  # oo
    self.x += 1  # oo
    return tmp  # oo

print([y for y in MyCounter() if y % 2 == 0])  # func oo proc
Human and Machine Judgements (on sample)

<table>
<thead>
<tr>
<th>Type</th>
<th>Human Judgements</th>
<th>Machine Judgements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imperative</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Mixed</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>Object-Oriented</td>
<td>53</td>
<td>55</td>
</tr>
<tr>
<td>Procedural</td>
<td>28</td>
<td>29</td>
</tr>
</tbody>
</table>
RQ1: Python files are small
RQ1: Python files are small
RQ1: Python files are small
RQ1: Python is Truly Multi-Paradigm
RQ3: Correlating # Statements with Predominant Paradigm

The diagram shows the comparison of files across different predominant paradigms: Object-Oriented (OO), Procedural, Imperative, and Mixed. The y-axis represents the number of files, ranging from 0 to 200,000. The x-axis categorizes the paradigms. The graph indicates a higher number of files in the OO paradigm compared to the other paradigms.
RQ3: Correlating # Statements with Predominant Paradigm
RQ3: Correlating # Statements with Predominant Paradigm

![Graph showing file counts for different paradigms (OO, Procedural, Imperative, Mixed)]
<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>101,648</td>
</tr>
<tr>
<td>Revision (with a Python file)</td>
<td>15,254,331</td>
</tr>
<tr>
<td>Python Files</td>
<td>7,758,882</td>
</tr>
<tr>
<td>(main branch only)</td>
<td>3,658,391</td>
</tr>
<tr>
<td>Python File Snapshots</td>
<td>68,787,597</td>
</tr>
<tr>
<td>ASTs</td>
<td>105,907,774,611</td>
</tr>
<tr>
<td>Category</td>
<td>Count</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Projects</td>
<td>101,648</td>
</tr>
<tr>
<td>Revision (with a Python file)</td>
<td>15,254,331</td>
</tr>
<tr>
<td>Python Files</td>
<td>7,758,882</td>
</tr>
<tr>
<td>(main branch only)</td>
<td>3,658,391</td>
</tr>
<tr>
<td>Python File Snapshots</td>
<td>68,787,597</td>
</tr>
<tr>
<td>ASTs</td>
<td>105,907,774,611</td>
</tr>
</tbody>
</table>

**Phase 1: Manual Classification**

**Phase 2: Automated Classification**
<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>101,648</td>
</tr>
<tr>
<td>Revision (with a Python file)</td>
<td>15,254,331</td>
</tr>
<tr>
<td>Python Files</td>
<td>7,758,882</td>
</tr>
<tr>
<td>(main branch only)</td>
<td>3,658,391</td>
</tr>
<tr>
<td>Python File Snapshots</td>
<td>68,787,597</td>
</tr>
<tr>
<td>ASTs</td>
<td>105,907,774,611</td>
</tr>
</tbody>
</table>

**Phase 1: Manual Classification**

**Phase 2: Automated Classification**
<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>101,648</td>
</tr>
<tr>
<td>Revision (with a Python file)</td>
<td>15,254,331</td>
</tr>
<tr>
<td>Python Files</td>
<td>7,758,882</td>
</tr>
<tr>
<td>(main branch only)</td>
<td>3,658,391</td>
</tr>
<tr>
<td>Python File Snapshots</td>
<td>68,787,597</td>
</tr>
<tr>
<td>ASTs</td>
<td>105,907,774,611</td>
</tr>
</tbody>
</table>

### Phase 1: Manual Classification

### Phase 2: Automated Classification

![Box plots showing file statements distribution by programming paradigms](image_url)
<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>101,648</td>
</tr>
<tr>
<td>Revision (with a Python file)</td>
<td>15,254,331</td>
</tr>
<tr>
<td>Python Files</td>
<td>7,758,882</td>
</tr>
<tr>
<td>(main branch only)</td>
<td>3,658,391</td>
</tr>
<tr>
<td>Python File Snapshots</td>
<td>68,787,597</td>
</tr>
<tr>
<td>ASTs</td>
<td>105,907,774,611</td>
</tr>
</tbody>
</table>

**Phase 1: Manual Classification**

**Phase 2: Automated Classification**

---

**Chart 1:**

- Y-axis: Statements (per file)
- X-axis: Statements, Functional, OO, Procedural, Imperative
- Data points show a range with a central tendency.

**Chart 2:**

- Y-axis: Files
- X-axis: OO, Procedural
- Bar chart showing distribution of files between OO and Procedural categories.