class Student:
    def __init__(self, name, year):
        self.name = name
        self.year = year

    def getName(self):
        return self.name

    def getYears(self):
        return self.year

    def setYears(self, newYear):
        self.year = newYear

if __name__ == '__main__':
    # Symbol table example
    jane = Student('Jane', 2024)
    print(jane.getYears())
    jane.setYears(2023)
1) `jane = Student('Jane', 2024)`

2) `self.name = 'Jane'`
3) self.Year = Year

line 18 ends

Line 19

Jane.get.Year() = 2027.

get.Year goes away

get.Yer

Next Year

Set

Year
5) `jane.set Year(2023)`

```
< Student object > jane

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>'Jane'</td>
</tr>
<tr>
<td>Year</td>
<td>2023</td>
</tr>
</tbody>
</table>

< Student class >

```

```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>--init--</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getName</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getYear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>setYear</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Self.set Year

Self.year = new Year 2023

Self.Year = 2023
```
Lecture 27: L-systems

Koch Snowflake

- Start with initial shape in mind that turtle will draw:

- Represent initial shape using a string: `F++F++F`

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>F</code></td>
<td>Turtle move forward by a distance <code>d</code></td>
</tr>
<tr>
<td><code>-</code></td>
<td>Turtle turn right by an angle <code>θ</code></td>
</tr>
<tr>
<td><code>+</code></td>
<td>Turtle turn left by an angle <code>θ</code></td>
</tr>
</tbody>
</table>

Symbols represent L-system alphabet

- `d` can be anything for Koch Snowflake
- `θ` should be `60°` for Koch Snowflake
we apply replacement rules some number of times to make the Snowflake fractal

Replacement rule composed of:

1) Find string — e.g. ‘F’ < every occurrence of this

2) Replace string — e.g. ‘F-F+FF-F’

Shape with replacement rule applied
Iteration 0: 'F++F++F' base string

Iteration 1:

\[ \text{F-F++F-F} \] + \[ \text{F-F++F-F} \] + \[ \text{F-F++F-F} \]

Draws Star

Pictures

Iteration 0:

\[
\begin{align*}
\text{Iteration 1:} & \quad \begin{align*}
\quad \text{apply replacement rule again}
\end{align*}
\end{align*}
\]

Iteration 2:

\[
\begin{align*}
\end{align*}
\]