PYTHON КАК ЯЗЫК, ОДНОВРЕМЕННО ПОДДЕРЖИВАЮЩИЙ НЕСКОЛЬКО ПАРАДИГМ ПРОГРАММИРОВАНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Аксёнов В. И.

Моженкова Е. В. – м-р техн. наук, ассистент

Анализируются проблемы соотношения императивных и декларативных подходов в современном программировании. Показывается, что такой высокоуровневый язык программирования общего назначения, как Python, который принято считать процедурным и объектно-ориентированным языком, он содержит все необходимое для поддержки полностью функционального подхода к программированию

Python — свободно распространяемый, высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Хотя и принято считать, что Python является процедурным и объектно-ориентированным языком, он содержит все необходимое для поддержки полностью функционального подхода к программированию.

В языках, называемых функциональными, хорошо поддерживаются нижеперечисленные подходы, а все прочие подходы поддерживаются плохо или не поддерживаются вовсе:

1) Функции – объекты первого класса, т.е., все, что можно делать с данными, можно делать и с функциями (вроде передачи функции другой функции в качестве параметра).

2) Использование рекурсии в качестве основной структуры контроля потока управления. В некоторых языках не существует иной конструкции цикла, кроме рекурсии.

3) Акцент на обработке списков (lists, отсюда название Lisp – LISt Processing). Списки с рекурсивным обходом подсписков часто используются в качестве замены циклов.

4) Чистые функциональные языки избегают побочных эффектов. Это исключает почти повсеместно распространенный в императивных языках подход, при котором одной и той же переменной последовательно присваиваются различные значения для отслеживания состояния программы.

5) Функциональное программирование не одобряет утверждения, используя вместо этого вычисление выражений (т.е. функций с аргументами). В предельном случае, одна программа есть одно выражение (плюс дополнительные определения).

6) Функциональное программирование акцентируется на том, что должно быть вычислено.

7) Большая часть языков функционального программирования использует функции высокого порядка (функции, оперирующие функциями).

Базовые элементы функционального программирования в Python – функции map(), reduce(), filter() и оператор lambda. В Python версии 1.0 введена также функция apply(), удобная для прямого применения функции к списку, возвращаемому другой. Python 2.0 предоставляет для этого улучшенный синтаксис.

Этих функций и всего нескольких базовых операторов достаточно для написания любой программы на Python; в частности, все управляющие утверждения (if, elif, else, assert, try, except, finally, for, break, continue, while, def) можно представить в функциональном стиле, используя исключительно функции и операторы.

В процессе исключения команд управления потоком Python замыкает накоротко вычисление логических выражений, что предоставляет эквивалент блока if/elif/else в виде выражения:

```
if <cond1>: func1()
elif <cond2>: func2()
else: func3()
```

Эквивалентное накоротко замкнутое выражение:

```
(<cond1> and func1()) or (<cond2> and func2()) or (func3())
```

Выражение lambda позволяет в общей форме представить условные возвращаемые значения. Базируясь на предыдущем примере:

```
>>> pr = lambda s:s
>>> namenum = lambda x: (x==1 and pr(one))
... or (x==2 and pr(two))
... or (pr(other))
>>> namenum(1)
one
>>> namenum(3)
Other
```

Замена циклов на выражения так же проста, как и замена условных блоков. for может быть впрямую переведено в map():

Пример функционального цикла for в Python:

```
for e in lst: func(e)
map(func,lst)
```

Императивное программирование по большей части состоит из утверждений. Функция map() позволяет выразить их следующим образом:

```
do_it = lambda f: f()
```

```
map(do_it, [f1,f2,f3])
```

В общем случае, вся главная программа может быть вызовом map() со списком функций, которые надо последовательно вызвать, чтобы выполнить программу.

Императивная версия цикла While в Python :

```
def while_block():
 <pre-suite>
 if <break_condition>:
 return 1
else:
 <suite>
 return 0
while_FP = lambda: (<cond> and while_block()) or while_FP()
while_FP()
```

Наш вариант while все еще требует функцию while_block(), которая сама по себе может содержать не только выражения, но и утверждения (statements).

Императивная версия функционального цикла echo в Python:

```
def echo_IMP():
 while 1:
x = raw_input(IMP -- )
 if x == quit:
 break
 else
 print x
echo_IMP()
def monadic_print(x):
 print x
return x
echo_FP = lambda: monadic_print(raw_input(FP -- ))==quit or echo_FP()
echo_FP()
```

Мы достигли того, что выразили небольшую программу, включающую ввод/вывод, циклы и условия в виде чистого выражения с рекурсией. Мы все еще используем служебную функцию monadic_print(), но эта функция совершенно общая и может использоваться в любых функциональных выражениях.

Выше продемонстрированы способы замены практически любой конструкции управления потоком в Python на функциональный эквивалент. Огромный процент программных ошибок и главная проблема, требующая применения отладчиков, случается из-за того, что переменные получают неверные значения в процессе выполнения программы. Таким образом, данные способы применяемые в функциональном программировании, обходят эту проблему.

Материалы данного доклада в расширенном виде апробированы студентами-вечерниками ИИТ специальности «Программное обеспечение информационных технологий» весной 2013 года как вспомогательные материалы по курсу «Функциональное программирование».

Список использованных источников:

1. Доусон М. Программируем на Python. — СПб.: Питер, 2012. — 432 с.
2. Лутц Марк. Программирование на Python / Пер. с англ. — 4-е изд. — СПб.: Символ-Плюс, 2011. — Т. I, II. — 992 с.
3. Шапошникова. Основы программирования на Python. Учебник. Вводный курс. — версия 2. — 2011.— 44 с
4. Лутц Марк. Изучаем Python / Пер. с англ. — 4-е изд. — СПб.: Символ-Плюс, 2010. — 1280 с.
5. Бизли, Дэвид М. Подробный справочник / Пер. с англ. — 4-е изд. — СПб.: Символ-Плюс, 2010. — 864 с.
6. Бизли, Дэвид М. Язык программирования Python. Справочник. — К.: ДиаСофт, 2002. — 336 с.
7. Хахаев И.А. Практикум по алгоритмизации и программированию на Python. Учебник. — М.: Альт Линукс, 2010. — 126 с. — (Библиотека ALT Linux).
8. Саммерфилд Марк. Программирование на Python 3. Подробное руководство / Пер. с англ. — СПб.: Символ-Плюс, 2009. — 608 с.