УДК 378.016:004.7

П. А. Хорошевич

P. A. Khoroshevich

Хорошевич Павел Александрович, преподаватель, БГПУ им. Максима Танка, г. Минск, Беларусь.

Khoroshevich Pavel Alexandrovich, lecturer, BSPU named after Maxim Tank, Minsk, Belarus.

ПРИМЕНЕНИЕ ИНСТРУМЕНТОВ ВИЗУАЛИЗАЦИИ В ПРОЦЕССЕ ОБУЧЕНИЯ СТУДЕНТОВ ЯЗЫКУ ПРОГРАММИРОВАНИЯ PYTHON

APPLICATION OF VISUALIZATION TOOLS IN THE PROCESS OF TEACHING STUDENTS THE PYTHON PROGRAMMING LANGUAGE

Аннотация. *В статье рассматривается целесообразность и особенности применения средств визуализации приложений в процессе обучения программированию на языке Python. Рассмотрены примеры применения онлайн-сервиса Pythontutor для визуализации особенностей синтаксиса языка.*

Annotation. *The article considers the relevance and specifics of application visualization tools in the process of learning Python programming. Examples of using the online service Pythontutor to visualize the properties of language syntax are considered.*

Ключевые слова: *обучение, программирование, визуализация.*

Keywords: *education, programming, visualization, Python.*

Изучение программирования, зачастую, вызывает ряд трудности у учащихся. Обучение предполагает формирование концептуального понимания того, как выполняется программа, какие управляющие структуры и типы данных при этом задействуются. Кроме этого, учащиеся должны освоить ряд программных инструментов, необходимых для разработки, тестирования и отладки программ. Программирование можно считать когнитивным навыком, так как программист создаёт умственную модель программы, то есть своё внутреннее представление и понимание её назначения, используемых данных и операций над ними. Затем данная модель накладывается на программный код для того, чтобы предсказать результат его работы, а также внести в него, в случае необходимости, требуемые изменения [1].

Чтобы ускорить процесс получения теоретических и практических навыков в области изучения программирования, разработано множество программных продуктов, многие из которых подразумевают ту или иную форму визуализации программы или алгоритма решения задачи.

Рассмотрим визуализацию некоторых алгоритмических конструкций и типов данных языка программирования Python, применив онлайн-приложения Pythontutor [2]. С помощью данного онлайн-приложения можно проиллюстрировать очерёдность выполнения команд и поток данных, которые изменяются в результате выполнения операторов. По мере пошагового выполнения программы, онлайн-приложение отображает значения переменных как базовых типов (*int*, *float*, *str*) так и ссылочных типов (*list*, *dict*, *tuple* и др.). Значения, хранящиеся в переменных последнего типа, показаны отдельно, а все переменные, которые ссылаются на эти данные, показаны стрелочками.

Рассмотрения сервиса Pythontutor начнём с примера его применения для объяснения особенностей работы цикла for. В отличие от других языков программирования, в Python цикл for предназначен для последовательного перебора элементов некоторой последовательности – строки, списка и др. [3]. Воспользуемся циклом for для перебора значений списка и символов в строке. На рисунке 1 представлен один из этапов визуализации программы.

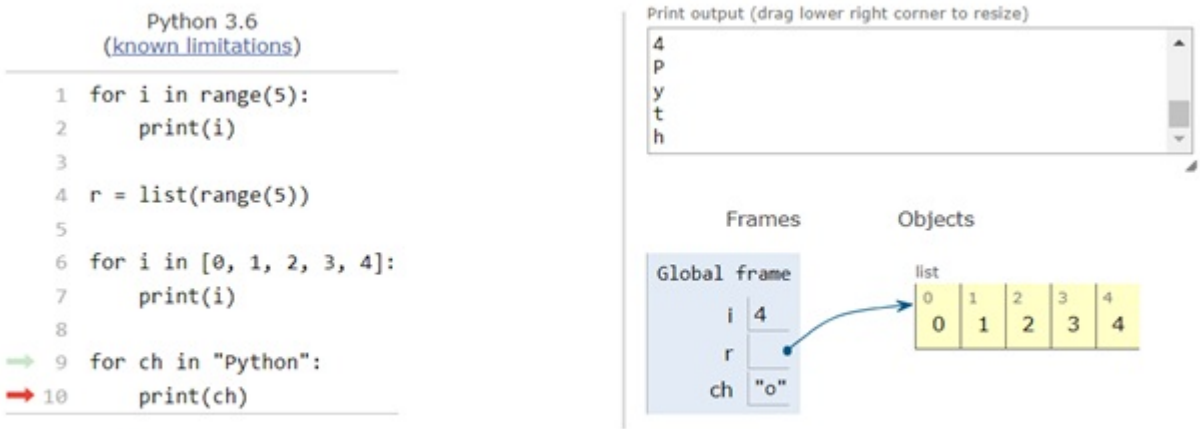

Рисунок 1. Работа цикла for

Левая часть страницы содержит исходный код программы, указатели выполненной строки, и строки, которая была выполнена предыдущей (зелёная и красная стрелки соответственно). В правой части страницы отображается результат вывода и графическое представление данных, используемых в исходном коде. Для неизменяемых типов значения отображаются непосредственно в фрейме, в случае с изменяемыми типами – стрелка указывает на данные, наглядное представление которых размещено справа от значений переменных базовых типов. В случае со списком изображение содержит пронумерованные клетки, каждая из которых содержит элемент данного списка.

Все переменные в Python являются экземплярами определённых классов – *int*, *str*, *list* и т.д. Функции, в свою очередь, также являются объектами, а это значит, что они могут быть элементами списка или передаваться в другие функции в качестве аргументов [3]. Пример на рисунке 2 иллюстрирует эту особенность функций в Python.

Рисунок 2. Функции как объекты

В данном примере объявлены три функции, которые затем помещаются в список. Наглядно представлено, что переменные *sum*, *mult*, *pow*, а также соответствующие элементы списка *func_lst* ссылаются на одни и те же объекты – функции.

В контексте изучения функций, важно понимать различия между изменяемыми и неизменяемыми типами данных. Неизменяемые типы передаются в функцию по значению, а изменяемые – по ссылке [3]. Данная особенность иллюстрируется примером, представленным на рисунке 3.

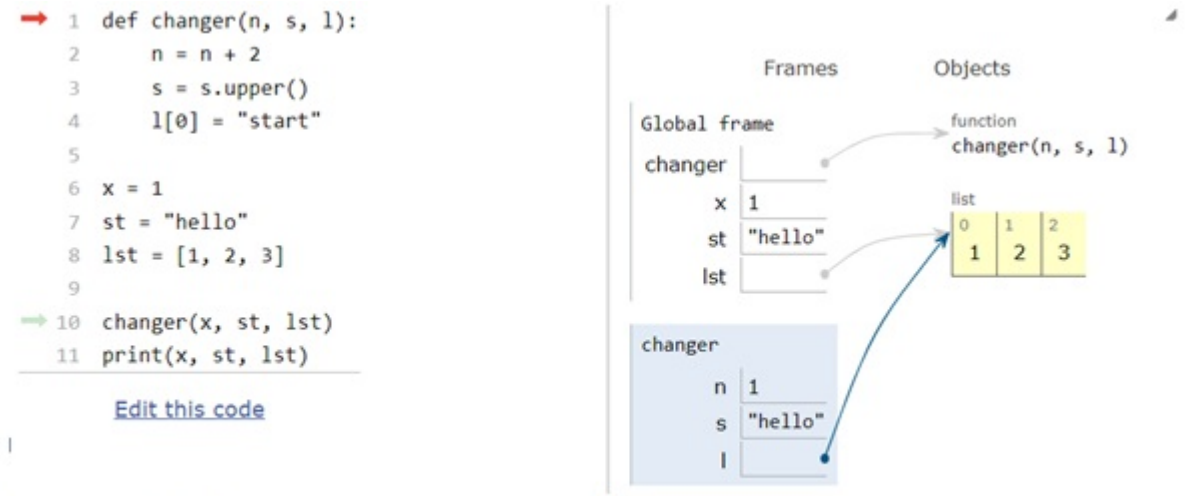

Рисунок 3. Передача аргументов в функцию по значению и по ссылке

В функцию *changer* передаются число, строка и список. В теле функции значения переданных аргументов изменяются. После вызова функции, переменные входящие в её область видимости отображаются в собственном фрейме. Число и строка были переданы по значению, а параметр *l* ссылается на тот же список что и переменная *lst*. Пример показывает, что изменив значение элемента списка внутри функции, мы поменяем это значение и для остальной программы. В случае с числом и строкой, их изменения будут носить локальный характер.

Изучение принципов объектно-ориентированного программирования в Python также имеет свои особенности. Каждый метод класса должен содержать как минимум один обязательный параметр, который принято называть *self* [3]. Этот параметр хранит ссылку на экземпляр класса, над которым выполняется действие. Зачастую его назначение не понятно учащимся и затрудняет процесс дальнейшего изучения ООП в Python.

На рисунке 4 показан код, иллюстрирующий процесс вызова метода класса *Point*.

Рисунок 4. Объявление класса и применение его методов

При вызове метода *getY*, параметр *self* ссылается на экземпляр класса Point, ссылка на который также хранится в переменной *p*, что наглядно изображено с помощью стрелочек.

Наследование, наряду с композицией является основой построения комплексных приложений. В примере, представленном на рисунке 5, описаны два класса – *Point* и его класс-наследник *LabeledPoint* [5]. На изображении показан один из шагов создания экземпляра класса *LabeledPoint* и последовательность вызовов конструкторов классов (методы *__init__*). При пошаговом выполнении примера, учащиеся могут проследить за тем, как с помощью функции *super* происходит обращение к методам родительского класса и на какие объекты ссылается параметр *self* в каждый момент времени.

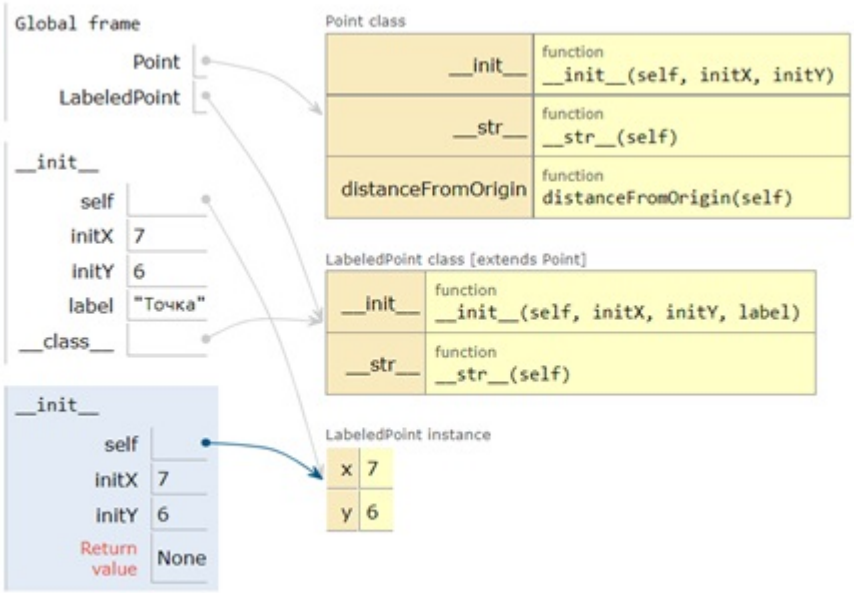

Рисунок 5. Пример наследования классов

В случае с композицией, одно из полей класса ссылается на экземпляр другого класса. С примером визуализации композиции можно ознакомиться по следующей ссылке [4].

Помимо иллюстрации основных синтаксических конструкций Python, визуализацию работы программы можно применить в ряде других случаев: демонстрация рекурсии, визуализация алгоритмов (нахождение НОД, численных методов и т.д.), визуализация различных структур данных (стек, очередь, и др.), демонстрация более сложных синтаксических конструкций (исключения, lambda-функции, декораторы, генераторы и т. д.).

Инструмент визуализации Pythontutor применялся нами для проведения факультативной учебной дисциплины, где показал свою эффективность, как при объяснении нового материала, так и проведении лабораторных работ. С его помощью учащиеся имеют возможность самостоятельно, или при участии преподавателя, сформировать представление о назначении и функционировании синтаксических конструкций и типов данных языка Python.

Список литературы

1. Effectiveness of Program Visualization: A Case Study with the ViLLE Tool [Текст]. / T. Rajala, M. Laakso, E. Kaila, T. Salakoski. // Journal of

Information Technology Education: Innovations in Practice, 2008. – № 7. – P. 17-31.
2. Pythontutor: сайт [Электронный ресурс]. – 2010. – URL : https:// pythontutor.com (дата обращения: 27.01.2022).
3. Любанович, Б Простой Python. Современный стиль программирования [Текст]. / Б. Любанович. – СПб. : Питер, 2016. – 480 с.
4. Пример композиции в Python: сайт [Электронный ресурс]. – 2022. – URL : https://cutt.ly/vTe3TrL (дата обращения : 27.01.2022).
5. Пример наследования в Python: сайт [Электронный ресурс]. – 2022. – URL : https://cutt.ly/5Te27rw (дата обращения : 27.01.2022).