ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ / INFORMATICS AND INFORMATION PROCESSES

DOI: https://doi.org/10.60797/IRJ.2025.153.82

ФОРМИРОВАНИЕ АЛГОРИТМИЧЕСКОГО МЫШЛЕНИЯ ПОСРЕДСТВОМ ИЗУЧЕНИЯ ЯЗЫКА PYTHON

Научная статья

Сакалова К.А.[1,] *, Быкова К.И.[2], Богданова М.В.[3], Сергеева О.В.[4]
[3] ORCID : 0000-0001-6769-0024;
[1, 2, 3, 4] Воронежский Государственный Педагогический Университет, Воронеж, Российская Федерация

* Корреспондирующий автор (s9515571372[at]gmail.com)

Аннотация

Статья посвящена исследованию роли языка программирования Python в формировании алгоритмического мышления у учащихся. В работе рассматриваются ключевые аспекты обучения программированию, включая разработку алгоритмов, структур данных и логику решения задач. Особое внимание уделяется практическим примерам использования Python для создания эффективных решений различных типов задач. Также анализируются методы преподавания, направленные на развитие критического мышления и способности к абстракции через программирование. Результаты исследования показывают, что Python является удобным инструментом для формирования базовых навыков алгоритмического мышления и может служить основой для дальнейшего изучения более сложных концепций информатики.

Ключевые слова: алгоритмическое мышление, Python, программирование, обучение, учебные ресурсы, практические примеры, проекты.

BUILDING ALGORITHMIC THINKING THROUGH LEARNING PYTHON

Research article

Sakalova K.A.[1,] *, Bykova K.I.[2], Bogdanova M.V.[3], Sergeeva O.V.[4]
[3] ORCID : 0000-0001-6769-0024;
[1, 2, 3, 4] Voronezh State Pedagogical University, Voronezh, Russian Federation

* Corresponding author (s9515571372[at]gmail.com)

Abstract

The article is dedicated to the study of the role of Python programming language in the development of algorithmic thinking in students. The paper examines key aspects of programming education, including the development of algorithms, data structures and logic of problem-solving. Particular attention is paid to practical examples of using Python to create effective solutions to various types of problems. Teaching methods aimed at developing critical thinking and the ability to abstract through programming are also analysed. The results of the research show that Python is a convenient tool for forming basic skills of algorithmic thinking and can serve as a basis for further study of more complex computer science concepts.

Keywords: algorithmic thinking, Python, programming, learning, learning resources, practical examples, projects.

Введение

Алгоритмическое мышление представляет собой способность структурировать задачу таким образом, чтобы её можно было решить пошагово, используя определенные правила и процедуры. Этот навык необходим не только программистам, но и специалистам в различных областях, где требуется анализ данных и принятие решений на основе этих данных [4]. Освоение языка программирования Python – это один из самых действенных методов развития алгоритмического мышления. Благодаря понятному и удобному синтаксису этот язык становится доступным для изучения даже теми, кто только начинает свой путь в программировании.

Стоит отметить, что способность структурировать задачи и решать их шаг за шагом – важный навык, который оказывает значительное влияние на учебный процесс.

При алгоритмическом подходе ученики учатся эффективно организовывать информацию. Это особенно полезно при изучении новых тем или при подготовке к экзаменам, когда важно легко усваивать и запоминать материал.

Алгоритмическое мышление развивает логическое мышление, что помогает ученикам лучше понимать причинно-следственные связи. Они учатся анализировать ситуации и предсказывать результаты своих действий.

Способность мыслить алгоритмически – это умение, которое находит применение не только в информатике. Этот навык полезен и в других областях знаний: математике, физике, экономике и даже в гуманитарных науках. Он помогает ученикам решать задачи и делать логические выводы в любой области.

Алгоритмическое мышление — важный инструмент для всестороннего развития учащихся. Оно улучшает процесс обучения и позволяет достигать высоких результатов в учёбе.

Основная часть

Процесс формирования алгоритмического мышления при изучении Python можно разделить на несколько этапов:

На первом этапе обучающиеся знакомятся с основными понятиями программирования, такими как переменные, типы данных, операторы, циклы и условия. Они учатся писать простые программы, выводящие текстовые сообщения на экран, выполнять арифметические операции и обрабатывать ввод пользователя.

Пример программы на Python:

```
print("Привет, мир!")
```

Следующим шагом является изучение способов работы с различными типами данных (целыми числами, строками, списками, словарями) и структурами данных (списками, кортежами, множествами). Обучающиеся осваивают операции над этими структурами, такие как добавление, удаление, изменение элементов, сортировка и фильтрация [7].

Пример работы со списком:

```
my_list = [1, 2, 3, 4]
print(my_list[0]) # Выведет первый элемент списка
```

Пример работы со строками:

```
 s = ‘Привет, мир!’
len(s) #Выведет 12
```

После освоения основ работы с данными обучающиеся переходят к изучению базовых алгоритмов и логических конструкций. Здесь они учатся реализовывать такие алгоритмы, как сортировка, поиск, обработка массивов данных, использование рекурсии и итерационных методов [9].

Пример функции для нахождения максимального значения в списке:

```
def find_max(arr):
    max_val = arr[0]
    for i in range(1, len(arr)):
        if arr[i] > max_val:
            max_val = arr[i]
    return max_val
numbers = [5, 7, 9, 12, 6]
result = find_max(numbers)
print(result)  # Выведет максимальное значение в списке
```

Пример функции для нахождения среднего арифметического:

```
def mean(values):
    sum = 0
    for v in values:
        sum+=v
        pass
    return sum/len(values)
```

Освоив основные алгоритмы, обучающиеся начинают изучать функции и модули. Функции позволяют разбивать сложные задачи на более мелкие части, делая код более читаемым и удобным для повторного использования. Модули же помогают организовывать код и обеспечивают возможность его многократного применения в разных проектах [3].

Простая функция сложения:

```
def add(x, y):
    return x + y
sum_result = add(10, 20)
print(sum_result) # Выведет сумму двух чисел
```

Пример функции вычисления суммы цифр числа:

```
def summa(n):
    s=n%10
    if n>=10:
        s+=summa (n//10)
    return s
```

На следующем этапе обучающиеся изучают объектно-ориентированный подход к программированию (ООП). Данный подход позволяет лучше понимать принципы абстракции, инкапсуляции, наследования и полиморфизма, что значительно упрощает разработку крупных приложений и систем.

Пример простого класса:

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def bark(self):
        print(f"{self.name} говорит гав!")
dog1 = Dog("Бадди", 3)
dog1.bark() # Выведет 'Бадди говорит гав!'
```

Для закрепления полученных знаний и развития алгоритмического мышления обучающимся предлагается участвовать в разработке реальных проектов. Это могут быть простые приложения, такие как калькуляторы, игры («Крестики-нолики»), или более сложные системы управления базами данных.

Пример проекта – игра "Камень, ножницы, бумага":

```
import random
options = ["камень", "ножницы", "бумага"]
user_choice = input("Ваш выбор (камень/ножницы/бумага): ")
computer_choice = random.choice(options)
if user_choice == computer_choice:
```

```
    print("Ничья!")
elif (user_choice == "камень" and computer_choice == "ножницы") or \
     (user_choice == "ножницы" and computer_choice == "бумага") or \
     (user_choice == "бумага" and computer_choice == "камень"):
    print("Вы выиграли!")
else:
    print("Компьютер выиграл.")
```

Одно из главных достоинств языка программирования Python – его чистый и понятный код. По сравнению с другими языками, такими как C++ или Java, код на Python более лаконичен и лёгок для восприятия. Это особенно важно при использовании языка для решения задач ЕГЭ, поскольку такой подход позволяет учащимся сосредоточиться на алгоритме задачи, не отвлекаясь на лишние конструкции.

Код на Python обычно занимает лишь треть или даже одну пятую часть объёма аналогичного кода на других языках. Это позволяет тратить меньше времени на ввод, отладку и сопровождение программы. Для учителя информатики, который работает с группой из 10–15 учеников, это огромное преимущество, так как позволяет быстро выявлять и исправлять ошибки учащихся.

Ещё одно важное преимущество Python заключается в том, что он учит писать «правильный» код, используя отступы как обязательный элемент синтаксиса. Это помогает формировать алгоритмическое мышление, которое является ключевым навыком для специалистов в области информационных технологий.

Стоит обратить внимание, что учащиеся могут столкнуться с некоторыми трудностями при изучении Python. К ним можно отнести:

1. Непонимание базовых концепций программирования. Многие ребята могут использовать трудности с некоторыми понятиями такими как переменные, циклы, условия. Это может привести к сложностям при решении задач. Для решения этой проблемы стоит уделять больше времени на изучение программирования, детальнее разбирать моменты, которые вызывают трудности.

2. Ошибка с форматированием. Python чувствителен к отступам, и неправильно отформатированный код вызывает ошибки, которые трудно отследить. Для того чтобы таких ошибок не возникало, можно предложить использовать текстовые редакторы с поддержкой подсветки синтаксиса, которые помогают визуально отделять блоки кода. Подсветка отступов может облегчить выявление ошибок.

3. Сложности с библиотеками и фреймворками. Изобилие доступных библиотек и фреймворков может быть подавляющим для новичков. Для решения этой проблемы рекомендуем предложить учащимся начинать с простых проектов. Создание небольших проектов поможет учащимся изучить использование библиотек и фреймворков поэтапно.

4. Отладка кода. Ошибки в коде могут быть неочевидными, и учащиеся часто могут не знать, как эффективно отлаживать свои программы. Здесь стоит обратить внимание ребят на изучение методов отладки. Изучение методов отладки, познакомить их с инструментами отладки, такими как print() для вывода данных, и использование встроенных средств отладки в используемой среде.

Заключение

Освоение языка Python открывает перед вами новые перспективы в развитии логического мышления и подготовке к экзамену по информатике. Вот несколько аргументов в пользу изучения этого языка.

Простота и доступность Синтаксис Python легко изучить, что особенно полезно для начинающих программистов.

Многофункциональность. Python находит применение в различных областях: веб-разработке, анализе данных, искусственном интеллекте и автоматизации. Знание этого языка расширит ваши карьерные возможности.

Активное сообщество. У Python много преданных пользователей, которые всегда готовы помочь и поделиться ресурсами. Это огромный плюс, ведь у вас будет возможность общаться с людьми, добившимися успеха в программировании.

Разнообразие инструментов и библиотек. Для Python существует множество библиотек и фреймворков, упрощающих выполнение задач. Например, Pandas и NumPy для работы с данными, Django и Flask для веб-разработки. Развитие навыков. Изучение Python помогает развивать логическое мышление и решать проблемы. Эти умения могут пригодиться не только на экзамене, но и в реальной жизни.

Стоит также отметить, что учащиеся с большой охотой переходят на язык программирования Python с любого другого, отмечая при этом, простоту и легкость решения задач на данном языке программирования. Также данный язык отлично применим на экзамене по информатике, т.к. имеет множество различных встроенных функций, применение которых может значительно сократить время на экзамене. За счет простоты и понятности синтаксиса программы на Python значительно короче, чем на других языках и, как показывает практика, усваивается данный язык гораздо быстрее любого другого.

В целом, знание языка Python не только может облегчить подготовку к экзамену по информатике, но и открывает много возможностей для дальнейшего развития в IT-сфере.

Конфликт интересов

Не указан.

Рецензия

Все статьи проходят рецензирование. Но рецензент или автор статьи предпочли не публиковать рецензию к этой статье в открытом доступе. Рецензия может быть предоставлена компетентным органам по запросу.

Conflict of Interest

None declared.

Review

All articles are peer-reviewed. But the reviewer or the author of the article chose not to publish a review of this article in the public domain. The review can be provided to the competent authorities upon request.

Список литературы / References

1. Боклаг Н.Ю. Основы программирования на языке Python / Н.Ю. Боклаг. — Москва: Бибком, 2020. — 685 с.
2. Бэрри П. Изучаем программирование на Python / П. Бэрри. — Москва: Эксмо, 2023. — 897 с.
3. Гуриков С.Р. Основы алгоритмизации и программирования на Python / С.Р. Гуриков. — Москва: Форум, 2018. — 535 с.
4. Гэддис Т. Начинаем программировать на Python / Т. Гэддис. — Спб: БХВ-Петербург, 2019. — 768 с.
5. Златопольский Д.М. Основы программирования на языке Python / Д.М. Златопольский. — Москва: ДМК Пресс, 2022. — 610 с.
6. Лутц М. Python. Карманный справочник / М. Лутц. — Москва: Вильямс, 2015. — 321 с.
7. Поляков К.Ю. Программирование. Python. C++ / К.Ю. Поляков. — Москва: Бином. Лаборатория знаний, 2019. — 147 с.
8. Прохоренок Н. Python 3 и PyQt 5. Разработка приложений / Н. Прохоренок, В. Дронов. — Спб: БХВ-Петербург, 2019. — 832 с.
9. Светозарова Г.И. Практикум по программированию на алгоритмических языках / Г.И. Светозарова, Е.В. Сигитов. — Москва: Наука, 2018. — 320 с.
10. Федоров Д.Ю. Программирование на языке высокого уровня Python: учебное пособие / Д.Ю. Федоров. — Москва: Юрайт, 2019. — 161 с.

Список литературы на английском языке / References in English

1. Boklag N.Ju. Osnovy programmirovanija na jazyke Python [Basics of Python Programming] / N.Ju. Boklag. — Moscow: Bibkom, 2020. — 685 p. [in Russian]
2. Bjerri P. Izuchaem programmirovanie na Python [Learning Python programming] / P. Bjerri. — Moscow: Eksmo, 2023. — 897 p. [in Russian]
3. Gurikov S.R. Osnovy algoritmizatsii i programmirovanija na Python [Fundamentals of algorithmization and programming in Python] / S.R. Gurikov. — Moscow: Forum, 2018. — 535 p. [in Russian]
4. Gjeddis T. Nachinaem programmirovat' na Python [Starting Out with Python] / T. Gjeddis. — Spb: BHV-Peterburg, 2019. — 768 p. [in Russian]
5. Zlatopol'skij D.M. Osnovy programmirovanija na jazyke Python [Basics of Python Programming] / D.M. Zlatopol'skij. — Moscow: DMK Press, 2022. — 610 p. [in Russian]
6. Lutts M. Python. Karmannyj spravochnik. [Python. Pocket reference book] / M. Lutts. — Moscow: ID Vil'jams, 2015. — 321 p. [in Russian]
7. Poljakov K.Ju. Programmirovanie. Python. C++ [Programming. Python. C++] / K.Ju. Poljakov. — Moskva: Binom. Laboratorija znanij, 2019. — 147 p. [in Russian]
8. Prohorenok N. Python 3 i PyQt 5. Razrabotka prilozhenij [Python 3 and PyQt 5. Application development] / N. Prohorenok, V. Dronov. — Spb: BHV-Peterburg, 2019. — 832 p. [in Russian]
9. Svetozarova G.I. Praktikum po programmirovaniju na algoritmicheskih jazykah [Workshop on programming in algorithmic languages] / G.I. Svetozarova, E.V. Sigitov. — Moscow: Nauka, 2018. — 320 p. [in Russian]
10. Fedorov D.Ju. Programmirovanie na jazyke vysokogo urovnja Python: uchebnoe posobie [Programming in the high-level Python language: a tutorial] / D.Ju. Fedorov. — Moscow: Jurajt, 2019. — 161 p. [in Russian]