Министерство просвещения Российской федерации
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»
Институт математики, физики и информатики
Кафедра информатики, информационных технологий
и методики обучения информатике

Обучение будущих ИТ-специалистов функционально-ориентированному программированию на языке Python

Выпускная квалификационная работа
бакалавра по направлению подготовки
44.03.05 – Педагогическое образование (с двумя профилями подготовки)
Профиль «Математика и информатика»

Допущено к защите

«____» _____ 2023 г.

Зав. кафедрой: _____

Исполнитель:
обучающийся группы МиИ-1802
Павлова Е. Д.

_____

Руководитель:
к.п.н., доцент кафедры
ИИТиМОИ Газейкина А.И.

_____

Екатеринбург 2023

Оглавление

ВВЕДЕНИЕ

Современный мир с каждым годом развивается все стремительнее, информационные технологии проникают во все сферы жизни общества. Современное образование плотно связано с информационными технологиями. Потребность в ИТ-профессиях стремительно растет.

Современные языки программирования совершенствуются, добавляя концепции функционального программирования, некоторые новые языки сразу создаются функциональными. Поэтому, следуя тенденциям развития языков программирования, для успешной профессиональной деятельности, студентам, решившим связать свою жизнь с ИТ, необходимо изучать функциональное программирование.

Онлайн курсы набирают все большую популярность в связи с возможностью перехода на дистанционное обучение практически в любой сфере. Поэтому подача материала по программированию в виде онлайн курса может быть более востребованной, чем традиционное обучение. Созданный курс поможет всем заинтересованным изучить данную область языка, а также будет полезен в профессиональной деятельности педагогов.

Объект: процесс профессиональной подготовки будущего ИТ-специалиста.

Предмет: обучение будущего ИТ-специалиста функционально-ориентированному программированию.

Цель: разработать онлайн курс «Основы функционального программирования на языке Python» для обучения будущих ИТ-специалистов функциональному программированию на языке Python.

На основании цели исследования были сформулированы следующие задачи:

1. Освоить парадигму функционального программирования и особенности ее реализации в языке Python, определить место

функционального программирования в профессиональной подготовке будущего ИТ-специалиста

2. Обосновать выбор платформы Stepik для размещения онлайн курса для обучения функциональному программированию
3. Проанализировать существующие онлайн курсы для обучения функциональному программированию.
4. Разработать структуру, выполнить отбор материала и реализовать обучающий онлайн курс «Основы функционального программирования на языке Python» на платформе Stepik.
5. Предложить методические рекомендации по использованию созданного курса в процессе подготовки будущих ИТ-специалистов и провести его экспертную оценку.
6.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОБУЧЕНИЯ ФУНКЦИОНАЛЬНОМУ ПРОГРАММИРОВАНИЮ НА PYTHON БУДУЩИХ ИТ-СПЕЦИАЛИСТОВ

1.1. Парадигма функционального программирования и ее реализация в языке Python

Парадигмой в программировании называется совокупность идей и понятий, определяющих стиль описания компьютерных программ. Парадигма функционального программирования предполагает обходиться вычислением результатов функций от исходных данных и результатов других функций, и не предполагает явного хранения состояния программы. Также не предполагается и изменяемость этого состояния в отличие от императивного программирования, где одной из базовых концепций является переменная, хранящая своё значение и позволяющая менять его по мере выполнения алгоритма.

В рамках функционального программирования выполнение программы – это процесс вычисления, который трактуется как вычисление значений функций в математическом понимании последних (в отличие от функций как подпрограмм в процедурном программировании). [16]

Функциональное программирование имеет следующие отличительные черты:

- Функции являются объектами первого класса. Это означает, что с функциями вы можете работать, также, как и с данными: передавать их в качестве аргументов другим функциям, присваивать переменным и т.п.
- В функциональном программировании не используются переменные (как именованные ячейки памяти), а так как нет

переменных, то и нет операции присваивания, как это понимается в императивном программировании.

- Рекурсия является основным подходом для управления вычислениями, а не циклы и условные операторы.
- Используются функции высшего порядка. Функции высшего порядка – это функций, которые могут в качестве аргументов принимать другие функции. [14]

Функциональное программирование является одной из парадигм, поддерживаемых языком программирования Python. Python не является функциональным языком программирования, но его возможностей хватает, чтобы разрабатывать программы в функциональном стиле. Основными аспектами изучения функционального программирования на Python являются: понятие функции, безымянные лямбда-функции, функции высших порядков, понятия глобальных и локальных переменных, рекурсия, декораторы. [1] С данных тем необходимо начать изучение функционально-ориентированного программирования на Python, так как именно на них в дальнейшем придется опираться при изучении более сложного материала – например понятия генератора, понятия параметров *args и **kwargs, а также при написании больших программ, имеющих прикладное применение.

Рассмотрим подробнее аспекты, затрагивающие основы функционального программирования на языке Python.

Функции — это участки кода, изолированы от остальной программы и выполняющиеся только при вызове.

Функция в Python может быть определена с помощью оператора def или лямбда-выражением. В общем виде можно записать

```
def func(список_аргументов):
    выражение
```

или

```
lambda список_аргументов: выражение
```

В данном выражении *список_аргументов* – это список аргументов, отделенных запятой, и *выражение* – часть программного кода, которая в результате может выдать значение.

Таким образом оператор

```
def func(x, y):
    return x + y
res = 1
for i in range(1, 4):
    res += i
print(res)
```

будет эквивалентен оператору

```
func = lambda x, y: x+ y
```

В определении функции фигурируют формальные аргументы (в представленном примере – x и y). Некоторые из них могут иметь значения по умолчанию. Все аргументы со значениями по умолчанию следует указывать после аргументов без значений по умолчанию.

При вызове функции задаются фактические аргументы.

```
func(5, y=2)
```

В начале идут позиционные аргументы. Они сопоставляются с именами формальных аргументов по порядку. Затем следуют именованные аргументы. Они сопоставляются по именам и могут быть заданы в вызове функции в любом порядке. Разумеется, все аргументы, для которых в описании функции не указаны значения по умолчанию, должны присутствовать в вызове функции. Повторы в именах аргументов недопустимы.

Функция всегда возвращает только одно значение (или None, если значение не задано в операторе return или этот оператор не встречен по достижении конца определения функции). Однако возвращаемым значением также может быть кортеж.

Определив функцию с помощью лямбда-выражения, можно тут же её использовать:

```
(lambda x: x+2)(5)
```

Лямбда-выражения удобны для определения несложных функций, которые позже передаются другим функциям. [3; 12]

Последовательности, подаваемые на вход в качестве аргументов функций, являются итерируемыми объектами. Для обработки данных подаваемых на вход последовательностей используют итераторы. Итератор представляет собой объект перечислитель, который для данного объекта выдает следующий элемент, либо бросает исключение, если элементов больше нет. Итератор не имеет индексов и может быть использован только один раз. [24]

Функции являются объектами первого класса – их можно присвоить переменным и использовать. [6; 7]

```
>>> def mul5(x):
    return x*5
>>> V = 3
>>> F = mul5
>>> F(V)
15
```

В примере переменной F, в качестве значения, присваивается функция mul5, после этого появляется возможность вызвать ее как функцию.

В Python есть встроенные функции высших порядков, одним из аргументов которых являются другие функции. Например, функция range(), которая позволяет генерировать ряд чисел в рамках заданного диапазона и часто используется в цикле for. Также к таким функциям относят map(), zip(), filter(), reduce(), enumerate(). [11] Рассмотрим данные функции более подробно.

Функция map() позволяет обрабатывать одну или несколько последовательностей с помощью заданной функции. При этом за значения аргументов будут последовательно браться элементы указанных списков.

```
>>>list1 = [1, 2, 3, 4, 5]
>>>list2 = [-1, 2, -3, 4, -5]
>>>map(lambda x, y: x*y, list1, list2)
[-1, 4, -9, 16, -25]
```

При помощи функции zip() можно вернуть итератор с кортежами, в который n-ый кортеж состоит из n-ых элементов последовательностей, которые были переданы как аргументы. На вход функции аргументы передаются в указанной последовательности, порядок элементов соблюдается. Это позволяет поэлементно срастить две или более последовательности. Если на вход были переданы последовательности разной длины, то все они будут отрезаны до длины самой короткой последовательности.

```
>>>list1 = [1, 2, 3, 4, 5]
>>>list2 = [10, 20, 30, 40, 50]
>>>list3 = [100, 200, 300]
>>>list(zip(list1, list2, list3))
[(1, 10, 100), (2, 20, 200), (3, 30, 300)]
```

Функция filter() позволяет фильтровать значения последовательности. В результирующем списке будут только те значения, для которых значение функции для элемента истинно.

```
>>>list1 = [10, 4, 2, -1, 6]
>>>filter(lambda x: x < 5, list1)
[4, 2, -1]
```

В данном примере в результат попадают только те элементы x, для которых выражение $x < 5$ истинно.

Функция reduce() принимает 2 аргумента: функцию и последовательность, затем последовательно применяет функцию-аргумент к элементам списка, возвращает единичное значение. В Python 3 функция reduce() находится внутри библиотеки functools, а потому перед использованием её необходимо импортировать.

Вычисление суммы всех элементов списка при помощи reduce:

```
from functools import reduce
>>>list1 = [1,2,3,4,5]
>>>sum_all = reduce(lambda x, y: x + y, list1)
15
```

Вычисления при этом будут происходить в следующем порядке:

((((1+2)+3)+4)+5)

Цепочка вызовов связывается с помощью промежуточного результата x. Так же можно использовать третий параметр на случай подачи пустого списка в качестве аргумента.

```
>>> reduce(lambda x, y: x + y, [], 1)
1
```

В данном случае сумма нуля слагаемых будет равняться 1.

Функция enumerate() похожа на функцию range(), однако если range() позволяет получить только индексы элементов списка, то enumerate() – сразу индекс элемента и его значение. Данная функция генерирует кортежи, состоящие из двух элементов – индекса элемента и самого элемента. [18; 21]

```
>>>b = "hello"
>>> for i in enumerate(b):
     print(i)
(0, 'h')
(1, 'e')
(2, 'l')
(3, 'l')
```

```
(4, 'o')
```

В функциональном программировании нередко можно встретить рекурсию - вызов функции внутри нее самой. Таким образом можно создавать так называемые циклы без цикла. В качестве примера можно привести функцию, которая будет вычислять факториал числа.

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

Важно быть внимательным при написании рекурсивных функций во избежание возникновения бесконечных рекурсий. В данном примере важно ставить ограничение при n ровняющимся нулю, иначе произойдет вызов функции от -1, -2, -3 и так далее. Так же необходимо следить, чтобы функция вызывалась с правильным параметром – если внутри factorial(n) будет вызвана factorial(n), то получится бесконечная цепочка.

Поэтому при написании рекурсивной функции необходимо прежде всего оформлять условия завершения рекурсии, прописывать условие, из-за которого рекурсия когда-либо завершит работу. [23; 25]

Еще одним важным аспектом при изучении функционального программирования являются декораторы. Декоратор — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода.

```
def decorator_function(func):
    def wrapper():
        print('Функция-декоратор!')
        print('Выполняется обёрнутая функция')
        func()
        print('Выход из обёртки')
```

```
    return wrapper
```

В данном примере decorator_function() является функцией-декоратором. Она является функцией высшего порядка, так как принимает функцию в качестве аргумента, а также возвращает функцию. Внутри decorator_function() определяется другая функция, которая обёртывает функцию-аргумент и затем изменяет её поведение. Декоратор возвращает эту обёртку.

Пример работы данного декоратора:

```
>>> @decorator_function
>>> def hello_world():
        print('Hello world!')
>>> hello_world()
Выполняется обёрнутая функция
Hello world!
Выход из обёртки
```

Как можно заметить из примера – вызвать декоратор возможно через «@», разместив его над функцией, которую требуется задекорировать. Выражение @decorator_function вызывает decorator_function() с функцией hello_world в качестве аргумента и присваивает имени hello_world возвращаемую функцию.

Записать иначе можно следующим образом:

```
hello_world = decorator_function(hello_world)
```

Декораторы не обязательно должны быть функциями, это может быть любой вызываемый объект. Возвращать декораторы также могут объект любого типа, не обязательно функцию. [15]

```
>>> def decorator(func):
    return 'HELLO'
>>> @decorator
>>> def hello_world():
```

```
     print('hello world')
>>> hello_world
HELLO
```

Рассмотренный материал необходим и достаточен для того, чтобы освоить основы и принципы функционального программирования, понять их потенциал для студентов, изучающих программирование на Python. [17]

Современные языки программирования находятся в постоянном обновлении, совершенствуются с каждым годом, в том числе добавляя концепции функционального программирования. Некоторые новые языки сразу создаются функциональными, например Haskell, серия языков Lisp.

Для успешной профессиональной деятельности, студентам, решившим связать свою жизнь с ИТ, необходимо следовать за тенденциями развития языков программирования, в число которых входит изучение функционального программирования. [5; 20]

1.2. Особенности функционирования современных обучающих онлайн курсов

В наше время все большую популярность стали приобретать онлайн курсы. Это связано с удобством получения информации и их независимостью от внешних обстоятельств, так как пройти онлайн курс можно из любого места, где имеется подключение к интернету. [4]

Учебные курсы могут быть открытыми, найти их может любой пользователь сети интернет. Чаще всего для прохождения курса необходимо лишь зарегистрироваться на той платформе, которая его предоставляет. Существуют и закрытые курсы, на которые можно перейти только по индивидуальному приглашению, которое приходит на почту или аккаунт сервиса. Промежуточным вариантом между этими двумя являются курсы, прохождение которых работает по принципу «переход по ссылке». Найти в

сети интернет такие курсы нельзя, однако всем тем, кому была предоставлена ссылка, можно спокойно к нему подключиться.

Структура курса чаще всего предполагает несколько учебных блоков, включающих как теоретические, так и практические задания. Большинство сервисов предоставляет возможность обратной связи, а потому варианты практических заданий могут быть как в тестовом виде, так и в открытом – с последующей оценкой или ответом от преподавателя. Блоки можно создать сразу или же добавлять последовательно, редактировать и удалять.

Онлайн курсы имеют четко выстроенную структуру подачи материала, а потому не уступает качественно оффлайн занятиям с преподавателем. Так же, как и при традиционной учебе курс можно начать и закончить в определенное время – для этого достаточно в необходимое время предоставить студентам доступ и выставить завершающие даты сдачи последнего возможного или итогового задания.

Еще одним явным отличием курса от заданий на каком-либо обучающем сервисе является то, что при создании курса практически нет ограничений по добавляемому материалу – знакам, количеству заданий, блоков или медиа материалов. Создать курс может каждый именно таким по объему, какой предполагает изучаемая тема.

Современные сервисы для создания онлайн курсов в большинстве своем не могут предоставить возможность онлайн трансляций, однако авторы могут вставить ссылку на сторонние ресурсы, такие как «YouTube» или «Zoom» для того, чтобы студенты в нужное время могли подключиться и в режиме онлайн встретиться с преподавателем.

Курсы могут быть как бесплатные, так и коммерческие – большинство платформ никак не ограничивают своих пользователей в продаже доступа к курсу. Однако бесплатные варианты весьма распространены, а потому могут использоваться в учебных целях как учителями и преподавателями, так и сотрудниками фирм.

Явным минусом онлайн курсов является то, что часто студенты не могут обсудить появившиеся вопросы или возникшие проблемы в режиме реального времени, как они могли бы сделать при очном обучении. Однако этот минус можно компенсировать за счет явного плюса – доступности курса в любом месте и в любое время. После ответа преподавателя или решении проблемы студент всегда сможет продолжить выполнение заданий, в то время как при очном обучении время занятий ограничено расписанием.

Конечно, современное развитие не позволяет создать курсы по узконаправленным темам или темам, в которых необходимо выполнять много практических заданий, которые нельзя частично перенести на теорию. Но в целом онлайн курсы – это те же курсы, мало чем уступающие по качеству курсам, проведенным очно – ведь уровень подачи материала зависит только от преподавателя. Поэтому в условиях быстроменяющийся обстановки в мире обучающие онлайн курсы — это именно то, что позволит современной системе образования бесперебойно функционировать при дистанционном обучении и очень сильно расширить возможности дополнительного образования. [8]

Для реализации обучающих онлайн курсов существует много различных платформ, которые постоянно совершенствуются и дорабатываются, однако далеко не все могут подойти для обучения студентов программированию.

Для выбора подходящей платформы необходимо определить критерии, наличие которых будет оптимальным при создании обучающего онлайн курса по программированию. Были выделены следующие критерии:

1. возможность деления учебного материала на разделы и блоки для возможности составления учебного плана;
2. автоматизация учебного процесса;
3. возможность использования сетевых технологий для общения преподавателя со студентами в текстовом и видео формате;

4. возможность составления заданий в разных формах – тестовые, открытые, задания на сопоставление и т.д.;
5. наличие удобной мобильной версии;
6. возможность автоматической проверки открытых заданий, связанных с написанием программ. [2; 13]

Успешная реализация обучающего онлайн курса возможна в том случае, если все выделенные критерии, полностью или хотя бы частично будут выполнятся на платформе.

В качестве возможных платформ для дальнейшей реализации курса были выбраны:

- Online Test Pad;
- Google Classroom;
- Eduardo;
- Stepik.

Рассмотрим каждую из них относительно реализации выдвинутых критериев.

Online Test Pad – платформа, основной функцией которой является конструирование тестовых заданий различных видов – как стандартных на выбор одного или нескольких вариантов ответа, так и более редко используемые - составление фразы из предложенных слов, заполнение пропусков. Платформа позволяет задать время открытия и закрытия теста для студентов, имеется автоматическая проверка решенных заданий, результаты которой выводятся в виде сводной таблицу успеваемости или диаграммы. Так же у платформы есть мобильная версия.

Явным минусом является отсутствие возможности взаимодействия преподавателя напрямую со студентами – для объяснения материала, консультации или обратной связи потребуется воспользоваться сторонним сервисом. Так же явным недостатком является то, что среди предложенных вариантов заданий нет возможности создать открытое задание на

программирование с автоматической проверкой сервисом, только в виде задания открытого типа.

Google Classroom – платформа, разработанная компанией Google для организации дистанционного обучения. Платформа позволяет создавать отдельные разделы с прикреплением в них учебных материалов в виде документов или иных сервисов компании Google, например, гугл форм. Для каждого учащегося выводится журнал успеваемости, у педагога так же есть сводный журнал по всей группе. Платформа очень удобна для выстраивания учебного плана, так как каждому заданию можно установить свой срок сдачи, количество баллов. Для каждого задания возможно давать обратную связь в текстовом формате или прикрепляя проверенные файлы. При помощи сервисов Google так же возможно устраивать онлайн конференции.

Явным минусом данной платформы является её скудность в видах заданий. Гугл формы имеют вариант тестирования с автоматической проверкой ответов, однако для полноценного курса по программированию предлагаемых видов заданий будет недостаточно. Так же, как и в Online Test Pad, в Google Classroom невозможно реализовать задание, предполагающие написание учащимся программы, с последующей автоматической проверкой.

Eduardo – платформа, созданная образовательным проектом «Лекториум» для упрощения дистанционных или заочных занятий, разработки обучающих курсов и популяризации онлайн обучения. в курсах, созданных на Eduardo соблюдается следующая структура - есть несколько разделов, каждый раздел подразделяется на подразделы, а подразделы подразделяются на блоки. В каждый блок можно добавить один или несколько компонентов. Всего возможных компонентов четыре – форум, HTML, задача и видео. Форум добавит в блок чат, открытый для всех студентов и преподавателей для общения в реальном времени. В компонент HTML входит текст, изображения, объявление другие функции, которые возможно выполнить при помощи данного языка. В компоненте видео можно

импортировать видео с «YouTube» или других сервисах, а также добавлять субтитры – как встроенные сторонним сервисом, так и прописанные вручную. Несмотря на то, что импортировать видео придется целиком, можно настроить его воспроизведение начиная и заканчивая конкретным временем. Это очень удобно, если из длинного видео на данном этапе нужна лишь часть информации, при этом студентам всегда остается возможность просмотреть видео полностью, если они того пожелают.

Как к каждому разделу, подразделу или блоку, так и к каждому заданию можно настроить его время публикации и время, отведенное на его решение. В одном блоке задания могут появляться с любой периодичностью – у одних уже может истечь время выполнения, а другие еще даже не опубликоваться, так что курс можно распланировать на долгое время вперед и лишь проставить даты публикации заданий.

Платформа плохо адаптирована для мобильных телефонов. Некоторые задачи по программированию можно адаптировать к возможностям платформы, однако создать полноценную среду для обучения именно программированию платформа Eduardo так же не предоставляет возможности.

Stepik – образовательная платформа и конструктор онлайн курсов. Данная платформа похожа по структуре создаваемых курсов на Eduardo – сперва они делятся на модули, которые в свою очередь делятся на уроки, состоящие из шагов. Каждый шаг можно обозначить как обязательный или необязательный к выполнению, а также настроить ограничения начала и конца для выполнения каждого элемента курса. Для каждого шага можно настроить собственное обсуждение, где студенты в реальном времени могут задавать свои вопросы и отвечать на уже заданные. Платформа позволяет вставлять видео, таблицы, картинки, однако явно выделяет Stepik по сравнению с другими рассмотренными платформами большая вариативность для составления практических заданий. [9; 10]

Среди предлагаемых типов заданий есть «программирование» - задание предполагает написание студентом программы на изучаемом языке программирования. После отправки написанной программы платформа может автоматически проверить написанную программу при помощи заранее заготовленных педагогом тестов. Stepik поддерживает все популярные на данный момент языки программирования. В рамках одной задачи можно задать формат олимпиадного программирования – для каждого теста установить лимит времени и затрачиваемой памяти на его прохождение. Так же есть возможность дать студентам готовую программу, которую они могут использовать как заготовку дорабатывая или добавляя свои элементы.

Платформа полностью адаптирована для мобильных телефонов – на устройство можно скачать приложение или использовать мобильную версию через браузер.

Исходя из проведенного анализа можно сделать вывод что всем необходимым критериям в достаточной мере удовлетворяет только платформа Stepik, – она позволяет создать удобно структурированный курс и имеет возможность автоматизировать проверку всех необходимых задач для реализации курса по программированию.

Stepik — образовательная платформа и конструктор онлайн-курсов, основанная Николаем Вяххи в 2013 году. В январе 2016 года были выпущены мобильные приложения под iOS и Android.

Stepik имеет большой каталог открытых курсов, пройти любой из которых может каждый желающий. Можно смотреть подборки курсов по новизне, популярности или иным критериям, например, определенному предмету. Удобный интерактивный поиск позволяет найти все доступные курсы по любой интересующей теме. Так же можно указать на каком языке – русском или английском, необходимо осуществлять поиск по курсам.

Как говорилось ранее, курсы делятся на модули, которые делятся на уроки, состоящие из шагов. Шаги могут быть практической или

теоретической направленности. Теоретические шаги включают в себя возможность добавления теста, изображений, математических формул, ссылок или видео.

Практические задания на Stepik могут быть более чем 20 различных видов, проверяются только автоматически. Это могут быть тестовые задачи – выбор одного или несколько правильных ответов из предложенных, в том числе случайным образом из большого количества возможных – например 4 из 10. Так же есть задачи на сопоставление значений из двух списков, упорядочивание элементов, выбор одной или нескольких ячеек в таблице.

Задания на ввод ответа тоже имеют большую вариацию – заполнение пропуска вручную или из выпадающего списка, ввод численного ответа, в том числе с возможностью указать погрешность, ввод математической формулы, задача со случайной генерацией чисел из заданного диапазона, ввод текста или строки, а также возможность отправить полностью свободный ответ, не проверяемый преподавателями.

Более сложный пласт задач включает в себя задачу на программирование – написание программы на одном из языков программирования с последующей проверкой по заранее созданным педагогом тестам, задачу на ввод SQL запроса, а также HTML и CSS задачу – написание структуры и стилей html документа.

В базовой версии доступны три вида узконаправленных видов задач – это ввод химической формулы или химического уравнения, задача на таблицу Шульте по нахождению чисел в таблице в правильном порядке, а также программирование виртуального исполнителя и работа с заранее созданными файлами в TRIK Studio.

За прохождение каждого шага учащимся начисляются баллы. Теоретические материалы и видео всегда оцениваются в ноль баллов и автоматически отмечаются выполненными при перелистывании на следующий шаг. Все задачи по умолчанию оцениваются в один балл, однако

преподаватель может настроить иное количество баллов. Баллы могут уменьшатся из-за количества неверных попыток дать ответ на задачу. В текстовых задачах можно настроить выдачу баллов за частично-верный ответ.

Создать курс может любой желающий – для этого достаточно зарегистрироваться на платформе и в разделе «преподавание» нажать кнопку «новый курс». После этого платформа предложит вписать название курса – это обязательный этап, без которого нельзя начать работу над содержимым. После успешного ввода названия перед пользователем появится окно, сообщающее, что в курсе пока что нет ни одного урока и кнопкой для редактирования. Нажав на данную кнопку можно увидеть окошко конструктора курса, в которых пользователю предлагают дать название модулю и урокам внутри модуля, а также выбрать время начала каждого модуля.

Интересной функцией Stepik является то, что не обязательно каждый раз создавать урок с нуля. Если у пользователя уже есть созданный урок и он хочет его дублировать, то платформа предлагает осуществить это при помощи нажатия кнопки «добавить ваш существующий урок». Импортировать уроки можно как с того курса, над которым идет работа, так и с тех, что были созданы ранее.

Первый шаг всегда по умолчанию является теоретическим текстового вида. Изменить его на практический можно, если это необходимо, удалив его или поменяв местами с созданным позднее. Текстовый редактор в Stepik обладает всем необходимым функционалом – имеется возможность выделения шрифта жирным, курсивом или подчеркиванием, а также небольшой набор встроенных стилей, которые позволят поменять размер и начертание. Платформа позволяет менять цвет шрифта, создавать маркированные списки и изменять положение текста на странице.

В теоретические блоки можно вставить изображение, математическую формулу, таблицу и элемент кода. Если предлагаемый набор функций недостаточен для удовлетворения желаний разработчика, то всегда есть возможность перейти из наглядного варианта написания шага в вариант с тегами или полностью написанный кодом. Это позволяет использовать желаемые стили и форматирование.

Каждому уроку можно добавить свою собственную обложку для более наглядного отображения в программе курса. Уроки можно легко менять между собой просто перетаскивая их в режиме редактирования.

Конструктор курсов в Stepik удобен, интуитивно понятен и имеет большой функционал как для первичного создания, так и дальнейшего редактирования структуры курса и каждого шага в отдельности. Каждому модулю можно задать «мягкий дедлайн» - дату, после которой учащиеся будут получать меньшее количество баллов за задачи, «жесткий дедлайн» - дату, после которой учащиеся вообще не будут получать баллы за решения задач и конец модуля – учащиеся больше не будут видеть уроки из данного модуля.

У каждого курса есть промостраница – заглавная страница с описанием курса, которую гости увидят до того, как поступят на курс. На ней расположено краткое и полное описание, логотип и промовидео, язык, уровень сложности – рассчитан курс на начинающих, продолжающих или профессионалов, рекомендуемая нагрузка. Так же есть отдельные поля для указания целевой аудитории курса – можно указать для кого курс предназначен и кому он будет полезен.

Отдельными блоками можно прописать информацию о том, как будет проходить обучение и что именно входит в курс – будут ли видео-лекции, практические задания, какие виды заданий, будет ли обратная связь от преподавателей, зачетная или проектная работа, итоговый экзамен. Можно расписать полностью всю структуру курса для того, чтобы пользователям

было точно понятно подходит структура предстоящего курса под их требования или нет.

Следующим блоком можно указать все результаты, которые получит пользователь после и во время прохождения курса. Это могут быть как определенные знания, умения и навыки, так и просто описание приятных бонусов при обучении на курсе – доступ к базе с решениями, общение с другим студентами, сертификаты. Можно сообщить о конкретных результатах – у пользователей после прохождения курса может быть реализован конкретный продукт или проект что так же позволит заинтересовать потенциальных студентов.

Ученикам могут выдаваться сертификаты о прохождении курса. В базовой версии для возможности выдачи сертификата необходимо, чтобы курс имел только положительные отзывы, а также насчитывал более 500 учеников.

Всех создателей курса будет видно на карточках курса, в каталоге и в поиске. В качестве создателя можно указать не конкретного человека, а название организации.

Преподаватели также указываются на промостранице. У преподавателей по умолчанию нет прав доступа для управления курсом, однако это всегда можно поменять в настройках доступа. В качестве преподавателя можно пригласить человека, который никак не связан с созданием курса и выдать ему соответствующие права. Это очень удобно для обеспечения безопасности имеющихся данных, если преподаватель впервые будет работать на курсе.

Табель успеваемости доступен только в платных курсах. По курсу можно посмотреть полную статистику – сколько на данный момент в курсе модулей, уроков и шагов, а также количество учащихся, тестирующих, модераторов, преподавателей и администраторов, а также количество выданных сертификатов, в том числе с отличием.

Также есть статистика по шагам – можно посмотреть количество просмотров, уникальных просмотров и прохождений каждого шага.

Возможности платформы Stepik позволяют создать полноценный автономно работающий онлайн курс по программированию. В рамках обучения в ВУЗе, курс, разработанный на Stepik, позволит студентам упрощенно изучить парадигму функционально-ориентированного программирования на языке Python благодаря доступности, простого интерфейса и возможности в любой момент редактировать курс под запросы студентов.

1.3. Анализ существующих онлайн курсов для обучения функциональному программированию

Сейчас активно совершенствуются старые и появляются новые онлайн курсы по программированию. Рассмотрим некоторые из них: курсы «Python - Модуль 2», «"Поколение Python": курс для профессионалов», «Функциональное программирование с Python» на платформе Stepik и «Функциональное программирование и структуры данных на языке Python» на другой платформе.

В курс «Python - Модуль 2» помимо функционального программирования включены также модули «списки, кортежи, множества» и «словари». Раздел «Функции» состоит из 9 блоков, которые охватывают некоторые из основных аспектов функционального программирования:

1. Введение
2. Аргументы
3. Возвращения из функций
4. Локальные и глобальные переменные
5. Рекурсия
6. Лямбда функции
7. Функции map & filter

8. *args и **kwargs
9. Дополнительные задачи

Из-за того, что функциональный аспект не является основным в курсе, можно увидеть недостаток информации – среди функций высшего порядка рассмотрены только две, библиотека functools также не упоминается.

Структура модуля непоследовательна – аргументам посвящены два отдельных блока под номерами 2 и 8. Несмотря на то, что блок номер 8 дополняет блок номер 2 они расположены очень далеко друг от друга.

У курса есть положительные отзывы, после его прохождения учащимся будут выданы сертификаты от платформы Stepik.

Курс «Python - Модуль 2» можно назвать хорошим курсом по изучению языка для тех людей, кто уже знаком с основами. Однако именно парадигма функционального программирования не раскрыта в полной мере даже в рамках информации для начинающих.

Второй курс для рассмотрения — это курс "Поколение Python": курс для профессионалов» также на платформе Stepik. Данный курс не фокусируется только на функциональном программировании, а содержит 12 модулей на различные темы. В отличии от первого рассматриваемого курса данный включает три модуля, относящихся к функциональному программированию. Это модули под номерами 8-10:

- Рекурсия
- Функции
- Итераторы и генераторы

Курс включает в себя во много раз больше информации, чем первый рассмотренный курс. В модуле «функции» рассмотрены встроенные функции, функции высших порядков, вложенность функций, а также декораторы и модуль functools. Модуль рекурсии идет перед модулем функции.

Курс рассчитан на тех, кто уже знает основы функционального программирования и хочет совершенствовать свои знания. В курсе не учат писать обычные функции и лямбда-функции, так как предполагается, что студент уже умеет это делать. На главной странице курса предлагается пройти два предыдущих курса тех же авторов- «"Поколение Python": курс для начинающих» и «"Поколение Python": курс для продвинутых».

В курс включены текстовые лекции, написанные простым и понятным языком, задачи на программирование и тесты на усвоение материала с автоматической проверкой, поддержка преподавателей курса, обратная связь от однокурсников, сообщество в Discord.

Каждый блок модуля состоит из множества шагов – их количество может быть более 15. Среди блоков есть как задания различного тестового типа, так и задания на написание программ.

Рассмотренный курс предлагает много информации и задач на отработку только что изученного материала, однако он рассчитан уже на продвинутых студентов. Если человек решит изучать функциональное программирование с нуля, то ему не хватит информации только из этого курса – она будет слишком сложной и ему придется либо изучать предыдущие курсы, либо самостоятельно искать информацию. Если же студент пройдет только один из предыдущих курсов данного цикла, то получит только информацию о самых азах.

Третий рассматриваемый курс - «Функциональное программирование с Python». Курс открытый, поступить на него может любой зарегистрированный на платформе Stepik пользователь. Данный курс содержит всего один модуль и 4 урока:

1. Lambda функции
2. Функции высшего порядка
3. Генераторы
4. Декораторы

Курс начинается с анонимных функций, пропуская стандартные. Это не удобно, так как если обучающийся еще не знаком с ними, то ему придется пользоваться сторонними ресурсами. Первый урок достаточно короткий – содержит 8 шагов, 5 из которых являются заданиями – 2 на написание программ и 3 тестового вида. Задания тестового вида идут после написания программ и направлены на закрепление изученного материала.

Второй урок посвящен функциям высшего порядка. Он включает в себя 14 шагов – из них всего 4 теоретических. Сразу после изучения теории студентам предлагается ряд задач на написание программ, а затем ряд тестовых заданий. Теоретическая часть слишком коротка, рассмотрены не все стандартные функции высшего порядка, не ко всем функциям приведены примеры. Из-за этого с написанием программ в последующих заданиях у учащихся могут возникнуть сложности.

Третий урок посвящен понятию генератора, а четвёртый понятию декоратора в Python. Они состоят из 8 и 5 шагов соответственно, структурно составлены так же, как и предыдущие уроки – теория, задачи на написание программ, тестовые задачи.

Данный курс недоработан – в нем отсутствует теория о понятия функции в Python, приведено мало примеров встроенных функций, но при этом студентам предлагается много объемных заданий на написание программ. Так как теоретической информации, предоставленной в курсе, для решения заданий на программирование недостаточно, студентам потребуется пользоваться сторонними ресурсами. Немногочисленная теория написана достаточно сложным языком, что также может вызвать дополнительные трудности.

Последний курс для рассмотрения – «Функциональное программирование и структуры данных на языке Python». Этот курс возможно проходить как в онлайн формате, так и в очном режиме. Курс рассчитан на 78 академических часов (12 занятий по 4 академических часа +

30 часов домашней работы). Курс является третьим в линейке курсов, посвящённых языку программирования Python. Он доступен тем, кто уже овладел языком Python на уровне структурного программирования и уверенно пишет программы на уровне курса информатики средней школы.

Программа курса включает в себя такие модули, как:

- Парадигмы программирования, возможность их реализации на Python.
- Функциональное программирование: суть, преимущества.
- Функции в Python, типы аргументов.
- Способы вызова функций.
- Модули, использование функций из модулей стандартной библиотеки.
- Подключение внешних библиотек.
- Рекурсия, её сильные и слабые стороны.
- Анонимные функции (лямбда-выражения).
- Функции высших порядков map, filter, zip, reduce.

Курс является полностью закрытым – в отличии от курсов на платформе Stepik, где всегда есть возможность посмотреть одни или несколько шагов даже в платных курах. Создатели курса пишут, что он будет полезен и тем, кто пробует себя в олимпиадном программировании, и тем, кто планирует далее осваивать практическое применение языка Python в различных сферах.

При поиске курсов для анализа возник ряд трудностей. Первая заключалась в том, что на русском языке очень тяжело отыскать онлайн курсы, которые затрагивали бы все необходимые аспекты функционального программирования. Практически все найденные курсы про функциональное программирование были платными. Многие предполагали уже профессиональную направленность, не подходящую для начинающего программиста.

В большинстве курсов функциональное программирование является лишь одним из модулей, но не отдельной темой.

Рассмотрев существующие курсы, можно сделать вывод о том, что актуально создать онлайн курс для обучения основам функционально-ориентированного программирования на языке Python.

ГЛАВА 2. РАЗРАБОТКА ОБУЧАЮЩЕГО ОНЛАЙН КУРСА «ОСНОВЫ ФУНКЦИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON»

2.1 Отбор содержания учебного материала и формирование структуры курса

На платформе Stepik курс включает в себя модули, которые в свою очередь делятся на уроки, состоящие из шагов. Теоретическую информацию необходимо распределить на несколько разделов таким образом, чтобы поданная в них информация была последовательно изложена. В изучении основ функционального программирования можно выделить несколько последовательных этапов, которые и будут являются разделами написанного курса. Первый этап – изучение простейших функций, их виды и структура написания, пример использования и свойства. Второй этап – рассмотрение глобальных и локальных переменных, а также встроенных в Python функций. Третий этап – работа с более сложными структурами, такими как рекурсия и декораторы.

В каждом этапе информация так же делится на шаги. Информация будет изложена таким образом, чтобы студенты постепенно погружались в функциональное программирование. В блоках первого урока студенты узнают о парадигме функционального программирования на языке Python. После будет предложено рассмотреть структуру и виды функций, понятие безымянных лямбда-функций, а также предложены задания тестового вида для закрепления понимания о взаимосвязи между видами функций. В заключительных шагах урока появятся первые задания на написание программы для того, чтобы студенты с первого занятия уже начали писать свои собственные программы. В открытом задании студентам необходимо будет написать программы с использованием простых функций, которые автоматически проверятся платформой.

В уроках второго модуля будут рассмотрены действия с функциями, присваивание, вложенность. Блоки будут сопровождаться заданиями тестового типа для улучшения закрепления понимания материала и повышения интереса к обучению.

Третий модуль целиком посвящен понятиям глобальных и локальных переменных. При работе с функциями студенту так или иначе придется пользоваться разными видами переменных, поэтому до перехода на более сложные структуры идет изучение инструкций global и nonlocal. На данном этапе студенты смогут начать писать первые полноценные программы, а не обособленные, как это было в предыдущих модулях.

Второй этап изучения начинается с самого крупного четвертого модуля. Студенты узнают какие функции уже встроены в Python. Студентам будет предложено пять уроков для изучения. Каждый блок сопровождается тестовыми заданиями, и задачами открытого типа. В курсе рассмотрены 4 основные встроенные функции - map(), zip(), filter(), sorted() и две функции из библиотеки functools - reduce() и enumerate(), так как они наиболее понятны и полезны при изучении основ функционального программирования. К каждой функции приведены примеры использования, понимание закрепляется заданиями открытого и тестового вида.

На третьем этапе обучения студенты научатся писать рекурсивные функции и познакомятся с понятием декоратора. В модуле про рекурсию два урока – понятие рекурсивной функции и виды рекурсивных функций. Уроки будут содержать задачи для самостоятельно написания студентами.

Заключительный шестой модуль посвящен понятию декоратора. В данном модуле студентов познакомят с тем, что такое декораторы и как написать их самостоятельно. Будет приведен пример композиции нескольких декораторов. В данном блоке нет заданий тестового типа, а только одно открытого. Для его выполнения студентам будет необходимо написать программу с несколькими функциями и несколькими декораторами. Задание,

как и сам блок несет ознакомительный характер, а его цель – дать студентам понять, как работают декораторы, провзаимодействовать с ними. Данный модуль информационно является заключительным, так как декораторы взаимодействуют с уже готовыми функциями.

Вся информация в курсе будет сопровождаться примерами в виде строк кода для того, чтобы студенты могли скопировать их и наглядно посмотреть, как работают конкретные программы. Ко многим заданиям тестового типа есть пояснения, возникающие после правильного ответа, к некоторым – подсказки, которые можно открыть при необходимости.

Структура получившегося курса выглядит следующим образом:

Основы функционального программирования на языке Python.

1. Функциональное программирование.
    1.1. Введение в функциональное программирование
    1.2. Понятие функции
    1.3. Лямбда-функции
2. Работа с функциями
    2.1. Присваивание функций
    2.2. Вложенность функций
3. Глобальные и локальные переменные
    3.1. Понятие глобальных и локальных переменных. Инструкция global
    3.2. Инструкция nonlocal
4. Встроенные функции
    4.1. Функция map()
    4.2. Функция zip()
    4.3. Функция filter()
    4.4. Функция sorted()
    4.5. Функции из библиотеки functools
5. Рекурсивные функции

5.1. Рекурсивные функции

5.2. Виды рекурсивных функций

6. Декораторы

6.1. Понятие декоратора

Итогового задания в курсе не предполагается, так как прогресс считается по каждому модулю исходя из количества выполненных заданий. Всего возможно получить 36 баллов. Подробно количество заданий по каждому модулю представлено в таблице 1.

Таблица 1

Количество заданий в курсе

Модуль	Количество тестовых заданий	Количество заданий на написание программ
1. Функциональное программирование	10	2
2. Работа с функциями	2	-
3. Глобальные и локальные переменные	5	-
4. Встроенные функции	3	11
5. Рекурсивные функции	-	2
6. Декораторы	-	1

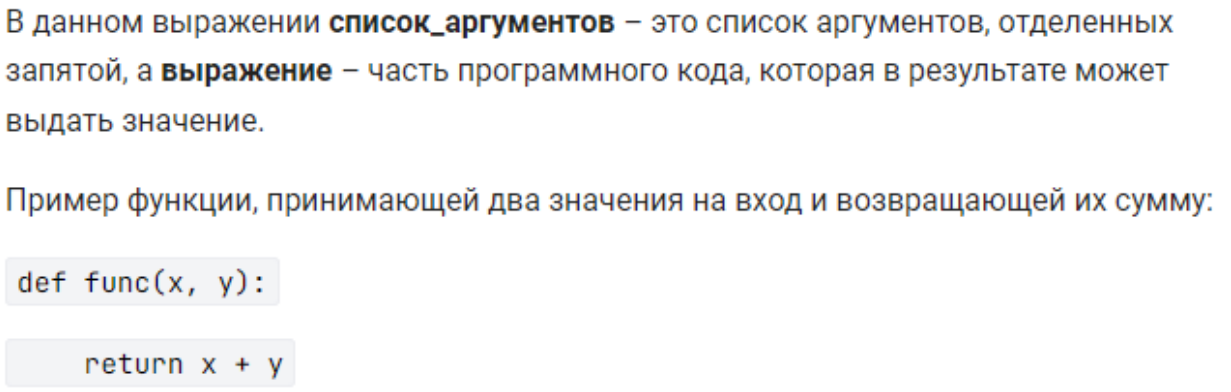
Предложенная структура курса позволит студентам постепенно погружаться в функциональное программирование на языке Python, не перегружая себя слишком сложной информацией и не перескакивая резко с темы на тему. Комментарии под каждым уроком открыты, а потому при возникновении трудностей студенты всегда смогут попросить помощи у других студентов или преподавателей.

2.2 Разработка заданий для обучающего онлайн курса

Разработку курса необходимо начать с создания модуля, урока и шага. Для удобства восприятия можно прописать сразу все урока в модуле.

Курс начинается с небольшого блока о функциональном программировании, как о парадигме языка программирования на Python. Все ключевые слова в процессе написания курса будут выделятся жирным шрифтом для удобства восприятия.

Во втором уроке студентам представлены структура функции и приведены строчки кода в качестве примера. Весь код из теоретической части курса можно копировать, он выделен другим шрифтом с соблюдением отступов.


В данном выражении **список_аргументов** – это список аргументов, отделенных запятой, а **выражение** – часть программного кода, которая в результате может выдать значение.

Пример функции, принимающей два значения на вход и возвращающей их сумму:

```
def func(x, y):
    return x + y
```

Рис. 1. Оформление текста в курсе

В этом и последующем уроках студентам будут предложены задания тестового типа. Самое первое на заполнение пропусков – необходимо написать функцию, позволяющую просуммировать все значения от 1 до n. Основные идеи сумматора уже будут вписаны, студенту необходимо дописать только те части, что относятся непосредственно к синтаксису функции. Остальные пять заданий из второго урока являются заданиями на выбор одного или нескольких вариантов ответа. Код везде прописан текстом поэтому у студентов при необходимости будет возможность легко скопировать данную программу, чтобы посмотреть результат ее работы.

Некоторые задания носят информационный характер. В шагах 6 и 7 рассмотрены случаи вызова функции, объявленной без переменных, без возвращаемого значения и без оператора return. Такие задания направлены на логику, так как на основе имеющийся неполной информации студентам

необходимо сделать вывод о новом нестандартном случае. После указания правильного ответа появится информационное окошко, которое объяснит студенту, почему его ответ верный.

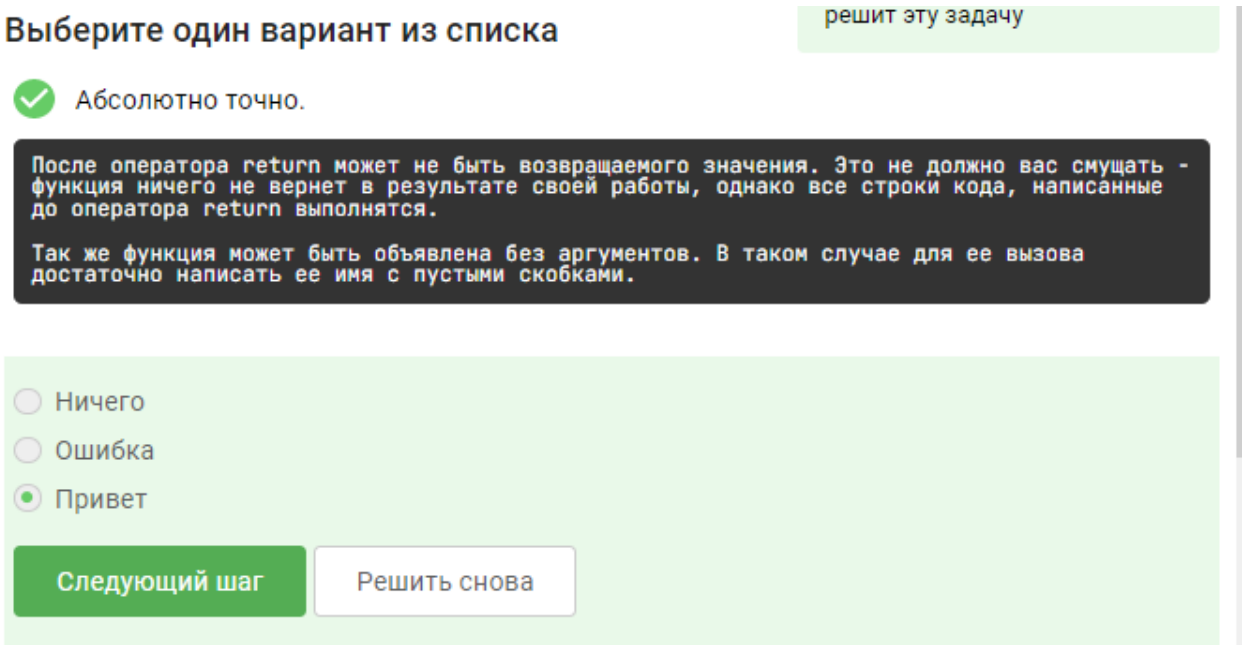


Рис. 2. Информационное окно с дополнительной информацией

Заключительным шагом урока является открытое задание на программирование. Оно звучит так: «Пользователь задает три числа. Если хотя бы одно из них делится на другое без остатка, то на экран выводится сумма чисел, если нет – произведение чисел.» Данное задание будет автоматически проверяться платформой Stepik.

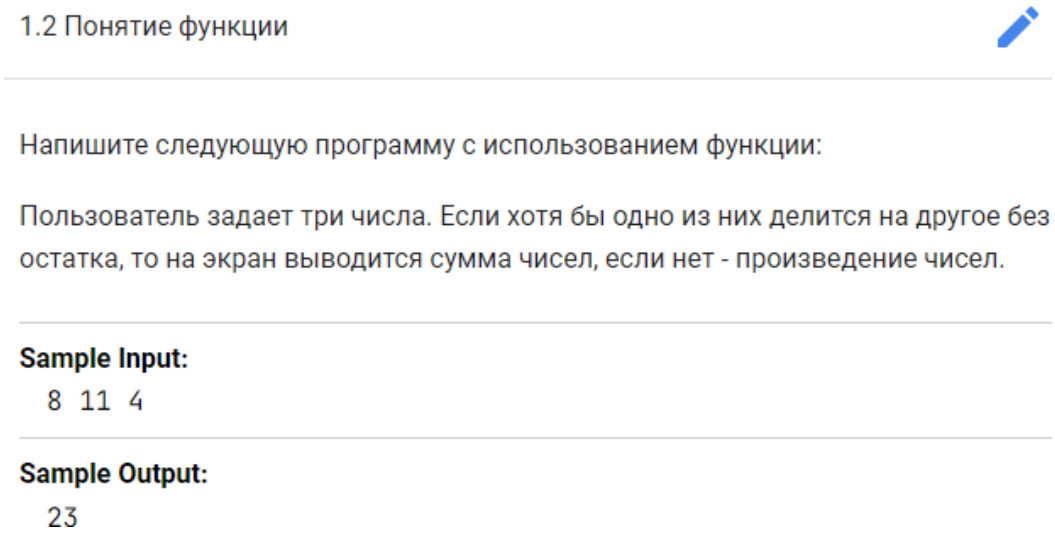


Рис. 3. Пример задания на написание программы

В третьем уроке про анонимные лямбда-функции упор сделан на задания на заполнение пропусков. Такой формат заданий позволяет наглядно показать, чем лямбда-функции отличаются от обычных функций. Помимо трех заданий на заполнение пропусков представлено задание на ввод текстового ответа и задание открытого типа на программирование. В нем студенту необходимо написать программу, содержащую лямбда- функцию, вычисляющую среднее арифметическое трех заданных чисел при вызове.

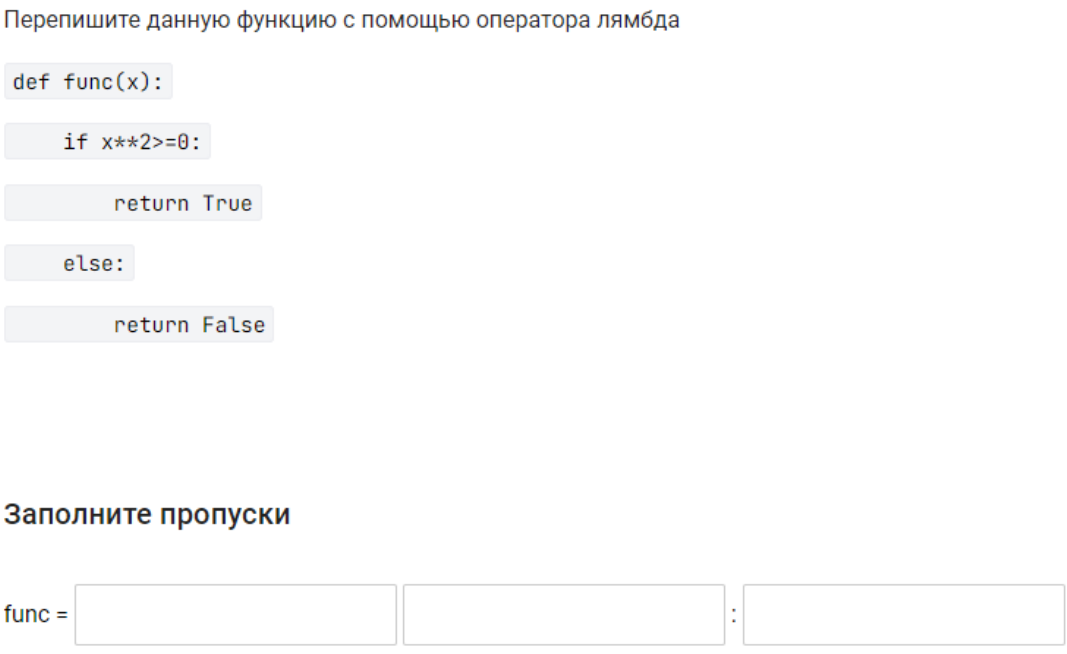


Рис. 4. Внешний вид задания на заполнение пропусков

В первом модуле содержится 6 заданий во втором уроке «Понятие функции» и 6 заданий в третьем уроке «Лямбда-функции». Подробные формулировки представлены в таблице 2.

Таблица 2

Содержание заданий 1 модуля курса

№	Вид задания	Формулировка задания	Форма ответа
1	Заполнение пропусков	Допишите фрагмент программы - функцию, позволяющий просуммировать все значения от 1 до n.	___ summ(___): for i in range (1, n+1): res+= i ___ ___
2	Выбор нескольких вариантов ответа	Строка с объявлением функции f выглядит следующим образом: def f(a, b, c=None, d="0"): Какие из последующих вариантов вызова не приведут к ошибке на этапе присваивания фактических параметров формальным?	1) f(1, 2) 2) f() 3) f(1, d=3) 4) f(1, 2, d=3, c=4) 5) f(1, 2, d=3) 6) f(1, 2, 3, 4)

3	Выбор одного правильного ответа	Что выведется в консоль при запуске данной программы?
		`def Func():`
		`  print("Привет")`
		`  return`
		`Func()`
			1) Ничего
			2) Ошибка
			3) Привет

4	Выбор одного правильного ответа	Что выведется в консоль при запуске данной программы?
		`def Func():`
		`  print("Привет")`
		`Func()`
			1) Ничего
			2) Ошибка
			3) Привет

5	Выбор нескольких вариантов ответа	Что будет выведено в консоль при запуске следующей программы:
		`def Func(x, y):`
		`  print(1)`
		`  if x%y==0:`
		`  print(2)`
		`  return True`
		`  print(3)`
		`  else:`
		`  print(4)`
		`  return False`
		`  print(5)`
			1) 1
			2) 2
			3) True
			4) 3
			5) 4
			6) False
			7) 5
			8) 6

		`    print(6)` `print(Func(-4, 2))`	
6	Написание программы	Напишите следующую программу с использованием функции: Пользователь задает три числа. Если хотя бы одно из них делится на другое без остатка, то на экран выводится сумма чисел, если нет - произведение чисел.	

7	Заполнение пропусков	Перепишите данную функцию, используя оператор лямбда
		`def func(x, y):`
		`    f = x*y`
		`    return f+x`
			func = ___ ___ , ___ : ___

8	Заполнение пропусков	Перепишите данную функцию, используя оператор лямбда
		`def func(x, y):`
		`    f = x*y`
		`    k = x+y`
		`    return f+k`
			func = ___ ___ , ___ : ___ + ___

9	Заполнение пропусков	Перепишите данную функцию, используя оператор лямбда
		`def func(x):`
		`    if x**2>=0:`
		`        return True`
		`    else:`
		`        return False`
			func = ___ ___ : ___

10	Выбор одного правильного ответа	Что выдаст компилятор при вызове следующей функции с аргументами (1, 2) `Func = lambda x, y: if x > 1: x + y else: x – y`	1) Ошибку 2) -1 3) Ничего 4) 3

11	Ввод текстового ответа	Что выведется в консоль при запуске данной программы?
		`Func = lambda x, y: Func2(x, y) * Func3(x, y) + Func4(y)`
		`def Func2(x, y):`
		`    return x+y`
		`def Func3(x, y):`
		`    return x*y`
		`def Func4(x):`
		`    return x**5`
		`Func(2, 3)`	

12	Написание программы	Напишите программу, содержащую лямбда- функцию, вычисляющую среднее арифметическое трех заданных чисел	

Во втором модуле впервые появляется задание на ввод численного ответа. Студентам необходимо внимательно смотреть написанные сложно структурированные программы для того, чтобы дать верный ответ. Это сделано специально для того, чтоб проверить внимательность в сложных конструкциях, которые и изучаются в разделах «присваивание функций» и «вложенность функций». Подробные формулировки представлены в таблице 3.

Таблица 3

Содержание заданий 2 модуля курса

№	Вид задания	Формулировка задания

1	Ввод текстового ответа	Можно использовать и более сложные конструкции, например вызвать функцию в аргументе другой функции
		
		`def Func1(a, b):`
		`  return a(b)`
		`def Func2(x):`
		`  return x**2`
		`def Func3(x):`
		`  return x**3`
		`print(Funk1(Funk3, Funk2(2)))`
		
		Какой результат будет после выполнения всех вычислений данной программы?

2	Ввод текстового ответа	Какой результат будет выведен в консоли при запуске следующей программы:
		
		`def Func(x, y):`
		`  if x%y==0:`
		`  def Func2(x, y):`
		`  return x+y`
		`  return True`
		`  else:`
		`  def Func2 (x, y):`
		`  return x-y`
		`  return False`
		`  print (Func2(2, 3))`
		`print (Func(-4, 2))`

В последующем разделе – «понятие глобальных и локальных переменных» подряд идут два шага с похожими заданиями на написание текстового ответа. Это специально сделано для того, чтобы студенты

наглядно увидели разницу и важность последовательности объявления переменных и присваивания им значений.

На данном этапе задания направлены не столько на отработку только что изученной информации, сколько на взаимосвязь между ранее изученным на курсе, только что полученной информацией и общими основами программирования на Python.

Все задания построены таким образом, что студент сначала может попробовать дать ответ, не создавая аналогичную программу, а лишь смотря на представленные строки кода на странице курса. Если же у него возникнут трудности, то код можно перенести в компилятор и узнать верное решение задания.

Помимо анализа написанных программ среди заданий так же встречаются теоретические задачи. Они представлены в таких видах заданий, как «флажки», то есть множественный выбор, и выбор одного варианта ответа. Такие задания направлены не только на проверку уже ранее изученной информации, но и на дополнительную, не сказанную тексте информационных шагов. Для того, чтобы дать ответ на такое задание студенту необходимо либо логически размышлять, сравнивая указанные структуры или инструкции с аналогичными из других модулей языка Python, либо проверить эмпирическим путем, написав программу с соответствующим условием.

Теоретических заданий не так много, как практических, так как курс больше является практико-ориентированным.

Рис. 5. Пример теоретического задания

Полный список заданий модуля «глобальные и локальные переменные» представлен в таблице 4.

Таблица 4

Содержание заданий 3 модуля курса

№	Вид задания	Формулировка задания	Форма ответа

1	Заполнение пропусков	Немного изменим программу, написанную ранее:
		
		`def msg():`
		`  global m`
		`  m = "Я внутри функции"`
		`  print(m)`
		`msg()`
		`m = "Я вне функции"`
		`print(m)`
		`msg()`
		
		Введите строчки, которые будут выведены в консоль после запуска программы
			Заполните пропуски:
			_____
			_____
			_____

2	Заполнение пропусков	А теперь еще немного поменяем строчки местами: `def msg():`	Заполните пропуски: _____ _____ _____

		global m m = "Я внутри функции" print(m) msg() print(m) m = "Я вне функции" msg() Введите строчки, которые будут выведены в консоль после запуска программы	
3	Выбор нескольких вариантов ответа	Выберите верные правила использования инструкции global	1) Если значение определено на выходе функции, то оно автоматически станет глобальной переменной. 2) Нет необходимости использовать global для объявления глобальной переменной вне функции. 3) Переменные, на которые есть ссылка внутри функции, неявно являются глобальными. 4) Ключевое слово global используется для объявления глобальной переменной внутри функции.

4	Ввод текстового ответа	Что выведется в консоль в результате работы следующей программы:
		
		def Funk1():
		total = 0
		def Funk2():
		def Funk3():
		nonlocal total
		total += 1
		return total
		return total
		return Funk2()
		f = Funk1()
		print(f)	

5	Выбор нескольких вариантов ответа	На инструкцию nonlocal накладываются следующие ограничения:	1) При использовании nonlocal поиск имен осуществляется только в областях видимости внешних функций 2) После инструкции nonlocal можно написать имя только одной

			переменной, для каждой последующей необходимо вновь прописывать инструкцию nonlocal 3) Имя переменной, которое объявляется в инструкции nonlocal, на момент вызова во вложенной функции должно уже существовать во внешней функции 4) В случае использования инструкции nonlocal во вложенной функции интерпретатор осуществляет поиск во всей области видимости программы

В модуле про встроенные функции первые четыре урока посвящены различным встроенным функциям. В них практически не встречается задний тестового типа – упор делается на задачи по программированию.

Всего в четвертом модуле 14 заданий – 11 на написание программ, 3 тестового вида. Подробные формулировки заданий представлены в таблице 5.

Таблица 5

Содержание заданий 4 модуля курса

№	Вид задания	Формулировка задания	Форма ответа
1	Написание программы	Напишите программу, которая на вход будет получать список чисел, а выводить их квадраты используя функцию map()	
2	Написание программы	Напишите программу, которая будет получать на вход список слов и добавлять символ "!" в конец каждого слова используя функцию map()	
3	Написание программы	Напишите программу, которая будет получать на вход список чисел, а затем возводить их в степень, равную их индексу	
4	Написание программы	Напишите программу, которая будет получать на вход два списка с числами, сравнивать числа с одинаковыми индексами в данных списках и выводить наибольшее	
5	Написан	Напишите программу, которая из заданного	

	ие программы	списка чисел вернет те, что попадают в заданный диапазон, используя функцию filter()	
6	Написание программы	Напишите программу, которая получит на вход строку из букв и вернет ее же, преобразовав все буквы в заглавные. Используйте функцию filter()	
7	Написание программы	Напишите программу, вычисляющую квадратный корень из чисел заданного списка. Используйте функции filter() и map()	

8	Выбор одного варианта ответа	Какой будет результат работы данной программы?
		`a = [['hello', 5, 'print', 0, 'sun', 10, 'dogs']`
		`b = sorted(a)`
		`print(b)`
			1)[0, 5, 10, 'dogs', 'hello', 'print', 'sun']
			2)['dogs', 'hello', 'print', 'sun', 0, 5, 10,]
			3)[0,'dogs', 'hello', 'print', 'sun', 5, 10,]
			4)Ошибка

9	Написание программы	Используя функцию sorted(), напишите программу, которая отсортирует числа из заданного списка по убыванию их модулей	
10	Написание программы	Используя функцию sorted(), напишите программу, которая сможет отсортировать смешанный список из второго шага по принципу - сначала числа по убыванию, затем слова по алфавиту	
11	Написание программы	Напишите программу, посимвольно конвертирующую строки в числа. Используйте функции reduce()	
12	Написание программы	Напишите программу, позволяющую вычислить корни квадратного уравнения вида ax^2+bx+c=0 через дискриминант. Используя функцию partial(), создайте аналогичную функцию для уравнений с единичным коэффициентом при x^2. На вход подаются значения коэффициентов, на выход значения корней уравнения с заданными коэффициентами в первой строке и с коэффициентом, равном 1 вместо заданного во второй строке	
13	Выбор одного варианта ответа	Какая встроенная функция Python лучше всего подходит для цепочечных вычислений?	1) reduce() 2) filter() 3) sorted() 4) map()
14	Выбор нескольких вариантов ответа	Какие функции позволяют организовать обработку сразу двух и более последовательностей?	1) filter() 2) reduce() 3) map() 4) zip()

В модулях, содержащих материал про рекурсивные функции и декораторы студенту необходимо будет решить два задания открытого типа на программирование в уроке про рекурсивные функции и одно задание на программирование в уроке, посвященному декораторам. Формулировки заданий представлены в таблице 6.

Таблица 6

Содержание заданий 5 модуля курса

№	Вид задания	Формулировка задания
1	Написание программы	Напишите программу, получающую на вход два числа и возводящую первое в степень, равную второму числу. Используйте рекурсивное вычисление
2	Написание программы	Напишите программу, которая запрашивает у пользователя слово, а выводит его же, где каждый начиная с середины и далее через один символы отделены знаком скобки
3	Написание программы	Напишите программу, получающую на вход некоторую строку и выдающую информацию короткая эта строка или длинная. Короткой будет считаться строка меньше 15 символов. Проверку необходимо производить при помощи декоратора с конструкцией if - else.

На этом разработка курса завершается – весь курс выполнен в едином стиле, с выделением ключевых слов жирным, все строки кода написаны специальным шрифтом, при необходимости ключевые слова – например названия функций, так же выделены жирным шрифтом. Курс удобен для восприятия новой информации, так как она изложена последовательно, с последующим закреплением при помощи заданий тестового или открытого вида.

Всего в курсе 36 заданий как открытого, так и тестового типа. Для удобства восприятия можно представить все задания в виде следующей таблицы:

Данный курс подходит как для полностью самостоятельного изучения студентами, так и для совместной с педагогом аудиторной работы.

После разработки курса необходимо максимально полно заполнить промостраницу курса – написать описание, краткое содержание получаемых знаний, а также на какую целевую аудиторию нацелен курс. Промостраница должна полностью отражать суть курса, не вводя будущих обучающихся в заблуждение.

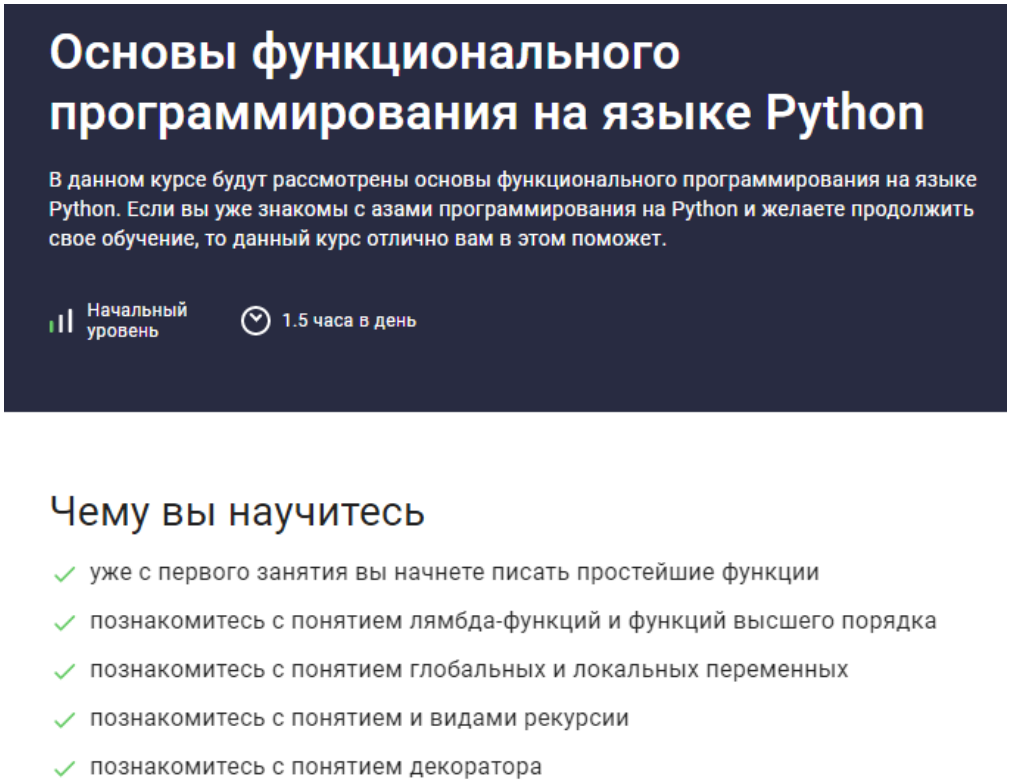


Рис. 6. Промостраница созданного курса

Разработанный курс подойдет как для самостоятельного изучения, так и для групповой работы в рамках предмета, посвященному изучению программированию на языке Python в ВУЗе. На данном этапе разработка курса окончена, можно приступать к апробации и экспертной оценке курса.

2.3. Методические рекомендации по применению и экспертная оценка учебного онлайн курса

Перед апробацией разработанного онлайн курса «Основы функционального программирования на языке Python» были разработаны методические рекомендации по его применению в учебном процессе.

Список рекомендаций включает в себя:

1. Проходите курс последовательно, не начинайте новый модуль, пока не закончили предыдущий. Если у вас возникают сложности с решением представленных заданий, то обратитесь за помощью к другим студентам или преподавателю.
2. В модуле «Встроенные функции» изучайте функции по очереди, так как решения задач в уроке про одну функцию может требовать наличие знаний про другую функцию из предыдущих уроков.
3. Платформа Stepik считает попытки, с которых выполнено задание. Поэтому не торопитесь – перепроверьте ваши ответы перед отправкой. Особое внимание следует уделить задачам на программирование – проверьте корректность их работы перед отправкой при помощи кнопки «запустить код». Кнопка находится справа под окошком для ввода кода программы.
4. В задачах на программирование всегда виден один открытый тест, однако на самом деле их два или более. Поэтому если ваша программа верно выполняет первый тест, но при этом не засчитывается как верная попробуйте протестировать ее на других входных данных в сторонних компиляторах для обнаружения ошибки.
5. Обращайте внимание на формат входных и выходных данных. Если, например, вы верно решили задачу, но на выходе у вас

массив чисел, вместо ряда, выведенного через пробел, то, очевидно, задача будет засчитана как неверно выполненная.

6. На все задачи на программирование наложено ограничение по времени и памяти, однако оно достаточно большое – 15 секунд и 256 мегабайт. В рамках задач данного курса можно сказать, что ограничений вообще нет.
7. Пробуйте запускать в компиляторах все программы или куски программ, представленные в информационных блоках. Попробуйте что-то поменять и запускать заново для того, чтобы на практике увидеть работу различных функций.
8. Во всех заданиях тестового вида, где представлены строки кода в условии, сначала попробуйте самостоятельно понять суть задачи и отправить ответ на проверку. В случае, если он оказался неверным или вы хотите перепроверить свое решение, можете скопировать строки кода в компилятор.
9. В случае внеаудиторного прохождения курса не пытайтесь пройти его целиком за один день. Давайте себе отдохнуть, не перегружая себя большим количеством новой информации.

Пользуясь данными рекомендациями студентам будет легко начать прохождение курса. Также, при соблюдении рекомендаций, информация будет более качественно усваиваться, а процесс изучения курса станет более приятным и менее напряженным, особенно для студентов, впервые столкнувшимся с практикой прохождения онлайн курса.

В рамках апробации студентам групп МиИ-1801 и МиИ-1802 было предложено оценить разработанный курс по 6 критериям используя шкалу от 1 до 5. В оценке поучаствовало 35 человек. Подробно критерии и результаты анализа и оценки представлены в таблице 7.

Таблица 7

Результаты экспертной оценки

№	Критерий	Средний балл
1	Соответствие представленного материала заявленной теме, цели и задачам	5
2	Полнота представленного материала	4,83
3	Доступность представленного теоретического материала	4,89
4	Доступность представленного практического материала (формулировок заданий)	4,89
5	Возможность использования в педагогической деятельности	4,86
6	Увеличение мотивации и заинтересованности при изучении программирования (в сравнении с традиционным способом обучения)	4,91

Проанализировав результаты экспертной оценки, представленные в таблице 7, можно утверждать, что все участники экспертной группы положительно оценили разработанный курс. Наивысшей оценкой отметили возможность использования разработанного курса в педагогической деятельности 30 из 35 человек, увеличение мотивации и заинтересованности при изучении программирования (в сравнении с традиционным способом обучения) 32 из 35 опрошенных.

В курсе доступным языком изложен теоретический материал, формулировки заданий соответствуют возрасту учащихся, а также курс возможно использовать в педагогический деятельности для увеличения мотивации и заинтересованности при изучении программирования.

Заключение

В результате проделанной работы можно утверждать, что цель курсовой работы достигнута, все поставленные задачи выполнены: освоена парадигма функционального программирования и особенности ее реализации в языке Python, обоснован выбор платформы Stepik для дальнейшей реализации курса, проанализированы существующие онлайн курсы, посвященные изучению функционального программирования на языке Python, разработана структура, выполнен отбор материала и реализован курс «Основы функционального программирования на языке Python» на платформе Stepik, а также составлен список методических рекомендаций и проведена его экспертная оценка.

Реализованный на платформе Stepik онлайн курс имеет потенциал для дальнейшего использования в учебной и будущей профессиональной деятельности.

Список использованной литературы

1. Балса, А. Р. Особенности парадигмы функционального программирования на языке Python / А. Р. Балса, О. А. Гуляеев, М. А. Соседов // Российская наука в современном мире – Москва: Общество с ограниченной ответственностью "Актуальность.РФ", 2020. – С. 82-84.
2. Васильев, В. Н. От традиционного дистанционного обучения к массовым открытым онлайн-курсам / В. Н. Васильев, С. К. Стафеев, Л. С. Лисицына[и др.]// Научно-технический вестник информационных технологий, механики и оптики. – 2014. – № 1. – С. 199-205.
3. Все, что нужно знать о lambda-функциях в Python // URL: https://pythonru.com/osnovy/vse-chto-nuzhno-znat-o-lambda-funkcijah-v-python (дата обращения: 07.03.2023). – Текст : электронный.
4. Вьюшкина Е. А. Массовые открытые онлайн-курсы: теория, история, перспективы использования / Е. А. Вьюшкина // Известия Саратовского университета – Саратов: Изд-во СГЮА, 2015. – С. 78–83.
5. Газейкина, А. И. Формирование научного мировоззрения будущего ИТспециалиста в процессе обучения программированию / А. И. Газейкина// Педагогическое образование в России. – 2015. –№ 7. – С. 36-41.
6. Городняя Л. В. Основы функционального программирования. Курс лекций // М.: Интернет-университет информационных технологий, 2004. – 280 с.
7. Григорьева, В. В. Ознакомление с функционально-ориентированным программированием на языке Python / В. В. Григорьева // Перспективы развития математического образования в

эпоху цифровой – Тверь: Тверской государственный университет, 2023. – С. 43-46.

8. Гречушкина, Н. В. Онлайн-курс: определение и классификация /Н. В. Гречушкина. // Высшее образование в России. – 2018. – № 6. – С. 125-134.
9. Завьялова О.А., Зубаков А.Ф. Особенности разработки и использования электронных онлайн курсов на платформе Stepik [Электронный ресурс] // Состояние и перспективы развития ИТ-образования; Сборник докладов и научных статей Всероссийской научно-практической конференции Чебоксары: Изд-во Чуваш. ун-та, 2019 г. 323 с.
10. Зубаков, А. Ф. Использование платформы Stepik при организации смешанного обучения в высшем образовании / А. Ф. Зубаков // Актуальные вопросы / – Иваново, 2020. – С. 294-297.
11. Колотов, И. В. Функциональное программирование на языке «python» / И. В. Колотов // Экспериментальная наука: механизмы, трансформации, регулирование УФА: Общество с ограниченной ответственностью "Аэтерна", 2022. – С. 31-37.
12. Лямбда-функции и анонимные функции в Python // URL: https://pythonru.com/osnovy/ljambda-funkcii-i-anonimnye-funkcii-v-python (дата обращения: 07.03.2023). – Текст : электронный.
13. Методические рекомендации по разработке массовых открытых онлайн-курсов / сост. Л. К. Габышева [др.]. – Тюмень : ТИУ, 2017. – 24 с.
14. Основы функционального программирования на Python // URL: https://habr.com/ru/post/555378/ (дата обращения:). – Текст : электронный.
15. Полезные декораторы // URL: https://tirinox.ru/useful-decorators/(дата обращения: 12.05.2022).

16. Прохоренок Н. А. Python. Самое необходимое. // СПб: БХВ, 2011. — 416 с.
17. Роганова Н. А. Функциональное программирование: Учебное пособие для студентов высших учебных заведений // М.: ГИНФО, 2002. — 260 с.
18. Розов К.В., Подсадников А.В. Язык программирования Python в педагогическом вузе: от основ программирования до искусственного интеллекта. Информатика и образование. 2019; (6):26-33.
19. Руководство по глобальным переменным // URL: https://pythonru.com/osnovy/globalnye-peremennye-python (дата обращения: 15.03.2023). – Текст : электронный.
20. Русанова, Е. И. О востребованности молодого специалиста в сфере ИТ на современном рынке труда / Е. И. Русанова //Перспективы науки и образования. – 2013. – № 5. – С. 224-226.
21. Федоров Д. Ю. Программирование на языке высокого уровня Python// М.: Юрайт, 2019. — С. 14-19
22. Филд А., Харрисон П. Функциональное программирование: Пер. с англ. // М.: Мир, 1993. — 637 с.
23. Python и функциональное программирование // URL: https://pythonchik.ru/osnovy/funkcionalnoe-programmirovanie-v-python (дата обращения: 05.03.2023). – Текст : электронный.
24. Python. Итераторы и генераторы // URL: https://devpractice.ru/python-lesson-15-iterators-and-generators/ (дата обращения: 07.03.2023). – Текст : электронный.
25. Python. Лекция 3. Элементы функционального программирования. // URL: https://ideafix.name/wp-content/uploads/2012/03/Python-3.pdf (дата обращения: 05.03.2023). – Текст : электронный.