

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №11
дисциплины «Программирование на Python»

Выполнил:
Гайчук Дарья Дмитриевна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика
и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А.-доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: «Работа с множествами в языке Python».

Цель работы: приобретение навыков по работе с множествами при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы

1. Создала общедоступный репозиторий на GitHub, в котором будет использоваться лицензия MIT.

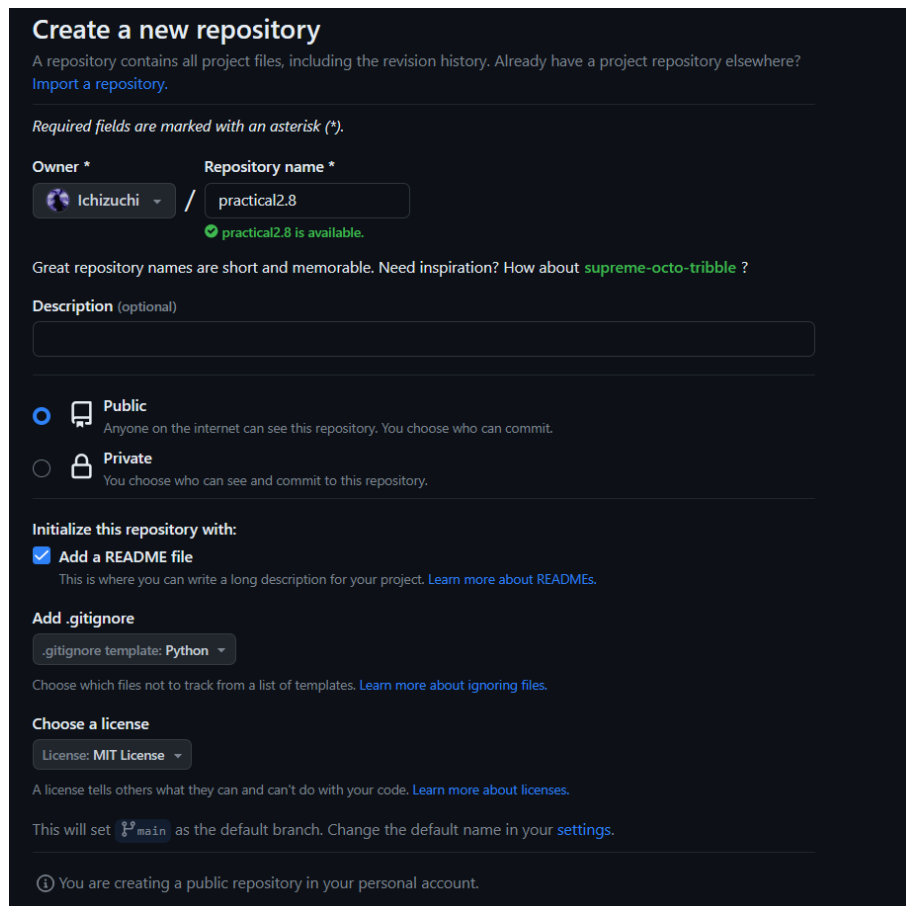


Рисунок 1. Новый репозиторий

2. Скопировала репозиторий на свой компьютер.

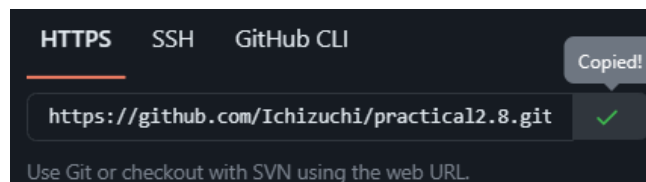


Рисунок 2. Клонирование репозитория

3. Использовала систему ветвления git-flow

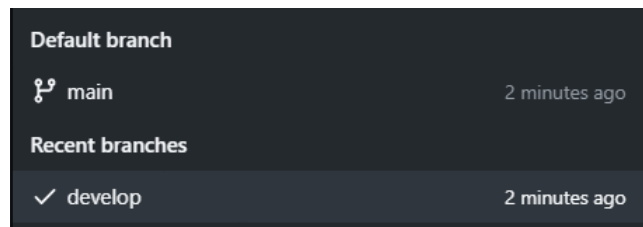


Рисунок 3. Ветка develop

4. Задача №1: основная ветка программы, не считая заголовков функций, состоит из двух строки кода. Это вызов функции `test()` и инструкции `if __name__ == '__main__':`. В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция `positive()`, тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция `negative()`, ее тело содержит выражение вывода на экран слова "Отрицательное".

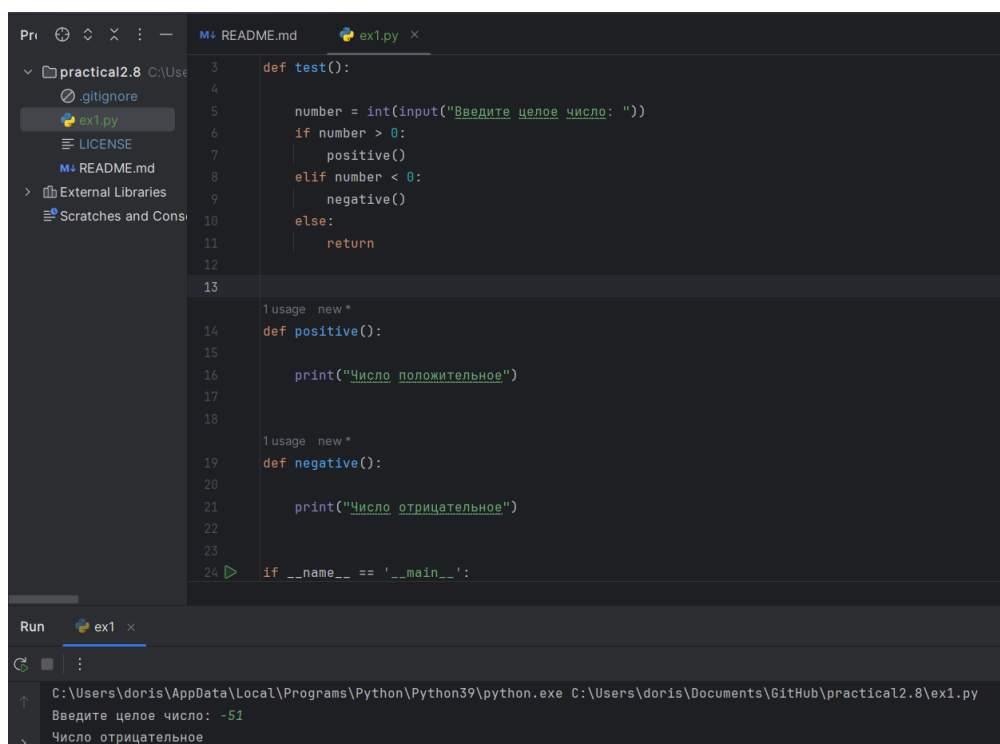
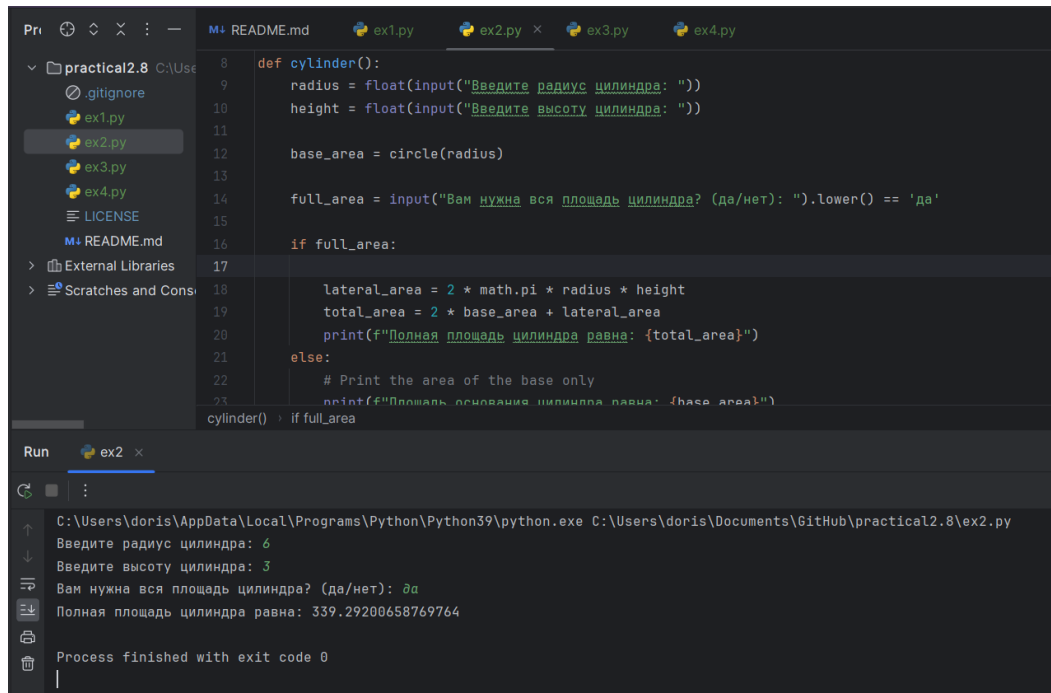


Рисунок 4. Программа и ее результат

5. Задача №2: в основной ветке программы вызывается функция `cylinder()`, которая вычисляет площадь цилиндра. В теле `cylinder()` определена функция `circle()`, вычисляющая площадь круга по формуле . В теле `cylinder()` у пользователя спрашивается, хочет ли он получить только площадь боковой поверхности цилиндра, которая вычисляется по формуле , или полную

площадь цилиндра. В последнем случае к площади боковой поверхности цилиндра должен добавляться удвоенный результат вычислений функции `circle()`.



```
def cylinder():
    radius = float(input("Введите радиус цилиндра: "))
    height = float(input("Введите высоту цилиндра: "))

    base_area = circle(radius)

    full_area = input("Вам нужна вся площадь цилиндра? (да/нет): ").lower() == 'да'

    if full_area:
        lateral_area = 2 * math.pi * radius * height
        total_area = 2 * base_area + lateral_area
        print(f"Полная площадь цилиндра равна: {total_area}")
    else:
        # Print the area of the base only
        print(f"Площадь основания цилиндра равна: {base_area}")

cylinder()
```

Run ex2 x

C:\Users\doris\AppData\Local\Programs\Python\Python39\python.exe C:\Users\doris\Documents\GitHub\practical2.8\ex2.py

Введите радиус цилиндра: 6

Введите высоту цилиндра: 3

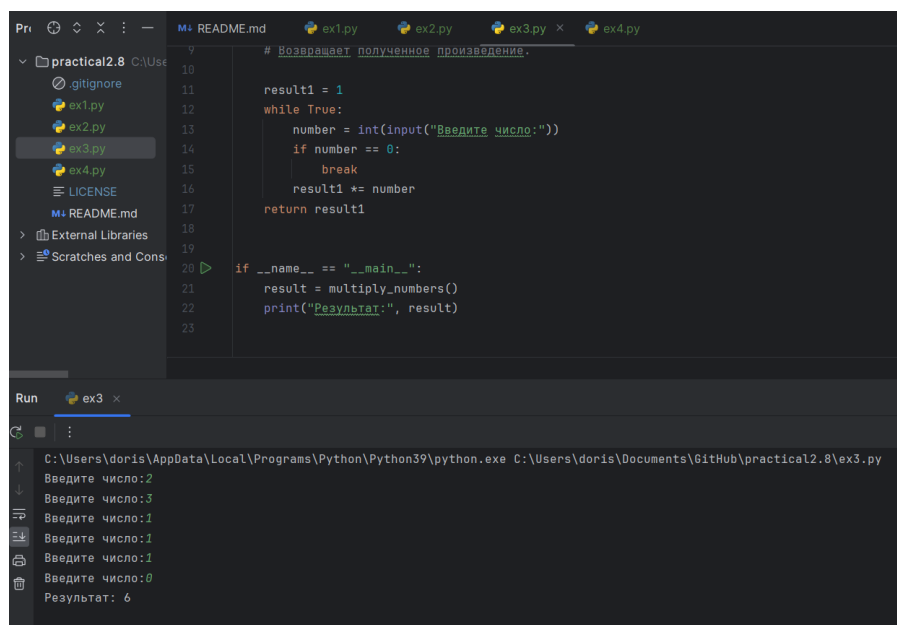
Вам нужна вся площадь цилиндра? (да/нет): да

Полная площадь цилиндра равна: 339.29200658769764

Process finished with exit code 0

Рисунок 5. Программа и ее результат

6. Задача №3: напишите функцию, которая считывает с клавиатуры числа и перемножает их до тех пор, пока не будет введен 0. Функция должна возвращать полученное произведение. Вызовите функцию и выведите на экран результат ее работы.



```
# Возвращает полученное произведение.

result1 = 1
while True:
    number = int(input("Введите число:"))
    if number == 0:
        break
    result1 *= number
return result1

if __name__ == "__main__":
    result = multiply_numbers()
    print("Результат:", result)
```

Run ex3 x

C:\Users\doris\AppData\Local\Programs\Python\Python39\python.exe C:\Users\doris\Documents\GitHub\practical2.8\ex3.py

Введите число: 2

Введите число: 3

Введите число: 1

Введите число: 1

Введите число: 1

Введите число: 0

Результат: 6

Рисунок 6. Программа и ее результат

7. Задача №4: напишите программу, в которой определены следующие четыре функции:

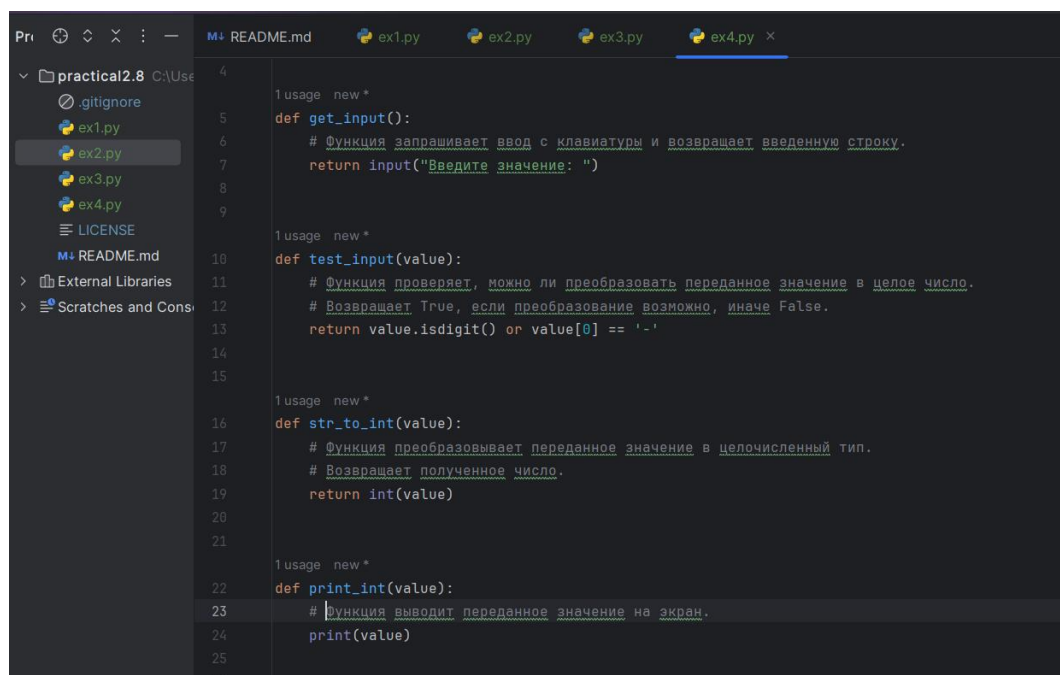
1) функция `get_input()` не имеет параметров, запрашивает ввод с клавиатуры и возвращает в основную программу полученную строку.

2) функция `test_input()` имеет один параметр. В теле она проверяет, можно ли переданное ей значение преобразовать к целому числу. Если можно, возвращает логическое `True`. Если нельзя – `False`.

3) функция `str_to_int()` имеет один параметр. В теле преобразовывает переданное значение к целочисленному типу. Возвращает полученное число.

4) функция `print_int()` имеет один параметр. Она выводит переданное значение на экран и ничего не возвращает.

5) в основной ветке программы вызовите первую функцию. То, что она вернула, передайте во вторую функцию. Если вторая функция вернула `True`, то те же данные (из первой функции) передайте в третью функцию, а возвращенное третьей функцией значение – в четвертую.



```
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

1 usage new *
2 def get_input():
3     # Функция запрашивает ввод с клавиатуры и возвращает введенную строку.
4     return input("Введите значение: ")
5
6
7 usage new *
8 def test_input(value):
9     # Функция проверяет, можно ли преобразовать переданное значение в целое число.
10    # Возвращает True, если преобразование возможно, иначе False.
11    return value.isdigit() or value[0] == '-'
12
13
14 usage new *
15 def str_to_int(value):
16    # Функция преобразовывает переданное значение в целочисленный тип.
17    # Возвращает полученное число.
18    return int(value)
19
20
21
22 usage new *
23 def print_int(value):
24    # Функция выводит переданное значение на экран.
25    print(value)
```

Рисунок 7. Четыре основные функции

Ответы на контрольные вопросы:

1. Каково назначение функций в языке программирования Python?

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции. Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы. Других ключевых отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы.

2. Каково назначение операторов `def` и `return` ?

Ключевое слово `def` сообщает интерпретатору, что перед ним определение функции. За `def` следует имя функции. Оно может быть любым, также как и всякий идентификатор, например, переменная. В программировании весьма желательно давать всему осмысленные имена. Функции могут передавать какие-либо данные из своих тел в основную ветку программы. Говорят, что функция возвращает значение. В большинстве языков программирования, в том числе Python, выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором `return`. Если интерпретатор Питона, выполняя тело функции, встречает `return`, то он "забирает" значение, указанное после этой команды, и "уходит" из функции.

3. Каково назначение локальных и глобальных переменных при написании функций в Python?

В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об областях видимости. Соответственно, локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей

программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение. К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции. При выходе из нее, локальные переменные исчезают. Компьютерная память, которая под них отводилась, освобождается. Когда функция будет снова вызвана, локальные переменные будут созданы заново.

4. Как вернуть несколько значений из функции Python?

В Питоне позволительно возвращать из функции несколько объектов, перечислив их через запятую после команды `return`. Фокус здесь в том, что перечисление значений через запятую (например, 10, 15, 19) создает объект типа `tuple`.

5. Какие существуют способы передачи значений в функцию?

В программировании функции могут не только возвращать данные, но также принимать их, что реализуется с помощью так называемых параметров, которые указываются в скобках в заголовке функции. Количество параметров может быть любым. Параметры представляют собой локальные переменные, которым присваиваются значения в момент вызова функции. Конкретные значения, которые передаются в функцию при ее вызове, будем называть аргументами. Следует иметь в виду, что встречается иная терминология. Например, формальные параметры и фактические параметры. В Python же обычно все называют аргументами. Когда интерпретатор переходит к функции, чтобы начать ее исполнение, он присваивает переменным-параметрам переданные в функцию значения- аргументы. Изменение значений переменных в теле функции никак не скажется на значениях, переданных в нее. Они останутся прежними. В Python такое поведение характерно для неизменяемых типов данных, к которым относятся, например, числа и строки. Говорят, что в функцию данные передаются по значению. Существуют изменяемые типы данных. Для Питона, это, например, списки и

словари. В этом случае данные передаются по ссылке. В функцию передается ссылка на них, а не сами данные. И эта ссылка связывается с локальной переменной. Изменения таких данных через локальную переменную обнаруживаются при обращении к ним через глобальную. Это есть следствие того, что несколько переменных ссылаются на одни и те же данные, на одну и ту же область памяти.

6. Как задать значение аргументов функции по умолчанию?

В Python у функций бывают параметры, которым уже присвоено значение по умолчанию. В таком случае, при вызове можно не передавать соответствующие этим параметрам аргументы. Хотя можно и передать. Тогда значение по умолчанию заменится на переданное. Например, в следующем определении функции параметр со значением по умолчанию будет `r`: `def cylinder(h, r=1)`.

7. Каково назначение lambda-выражений в языке Python?

Python поддерживает интересный синтаксис, позволяющий определять небольшие однострочные функции на лету. Позаимствованные из Lisp, так называемые lambda-функции могут быть использованы везде, где требуется функция. `lambda` – это выражение, а не инструкция. По этой причине ключевое слово `lambda` может появляться там, где синтаксис языка Python не позволяет использовать инструкцию `def`, –внутри литералов или в вызовах функций, например.

9. Как осуществляется документирование кода согласно PEP257?

PEP 257 описывает соглашения, связанные со строками документации python, рассказывает о том, как нужно документировать python код. Цель этого PEP - стандартизировать структуру строк документации: что они должны в себя включать, и как это написать (не касаясь вопроса синтаксиса строк документации). Этот PEP описывает соглашения, а не правила или синтаксис. Строки документации - строковые литералы, которые являются первым оператором в модуле, функции, классе или определении метода. Такая строка документации становится специальным атрибутом `__doc__` этого объекта. Все

модули должны, как правило, иметь строки документации, и все функции и классы, экспортируемые модулем, также должны иметь строки документации. Публичные методы (в том числе `__init__`) также должны иметь строки документации. Пакет модулей может быть документирован в `__init__.py`. Для согласованности, всегда нужно использовать `"""triple double quotes"""` для строк документации. Можно использовать `r"""raw triple double quotes"""`, если удет присутствовать обратная косая черта в строке документации. Существует две формы строк документации: однострочная и многострочная.

10. В чем особенность однострочных и многострочных форм строк документации?

Одиночные строки документации предназначены для действительно очевидных случаев. Они должны уместиться на одной строке. Однострочная строка документации не должна быть "подписью" параметров функции / метода (которые могут быть получены с помощью интроспекции). Этот тип строк документации подходит только для C функций (таких, как встроенные модули), где интроспекция не представляется возможной. Тем не менее, возвращаемое значение не может быть определено путем интроспекции. Многострочные строки документации состоят из однострочной строки документации с последующей пустой строкой, а затем более подробным описанием. Первая строка может быть использована автоматическими средствами индексации, поэтому важно, чтобы она находилась на одной строке и была отделена от остальной документации пустой строкой. Первая строка может быть на той же строке, где и открывающие кавычки, или на следующей строке. Вся документация должна иметь такой же отступ, как кавычки на первой строке

Вывод: в результате выполнения работы были приобретены навыки по работе с функциями при написании программ с помощью языка программирования Python версии 3.x.