Лабораторная работа №4
Создание собственных функций, модулей и пакетов.

Цель работы: научиться разрабатывать и отлаживать программы с использованием стандартных модулей и функций, создавать пользовательские функции, объединять их в модули и подключать

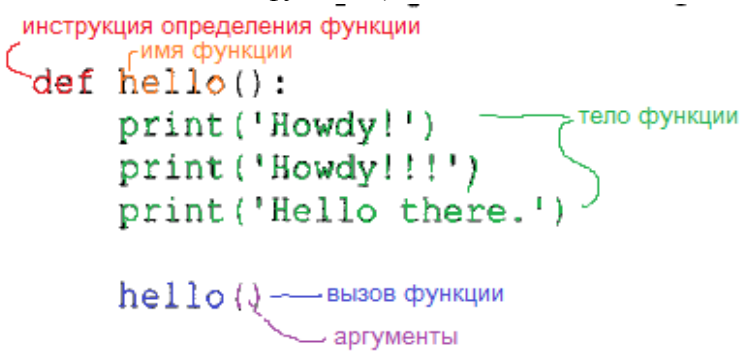
Задание на лабораторную работу:

1. Изучить теоретические сведения.
2. Написать программу в соответствии с вариантом.

Теоретические сведения

Основное назначение функции — группирование многократно выполняемого кода.

Они позволяют дать имя определённому блоку команд с тем, чтобы впоследствии запускать этот блок по указанному имени в любом месте программы и сколь угодно много раз (это называется *вызовом* функции).

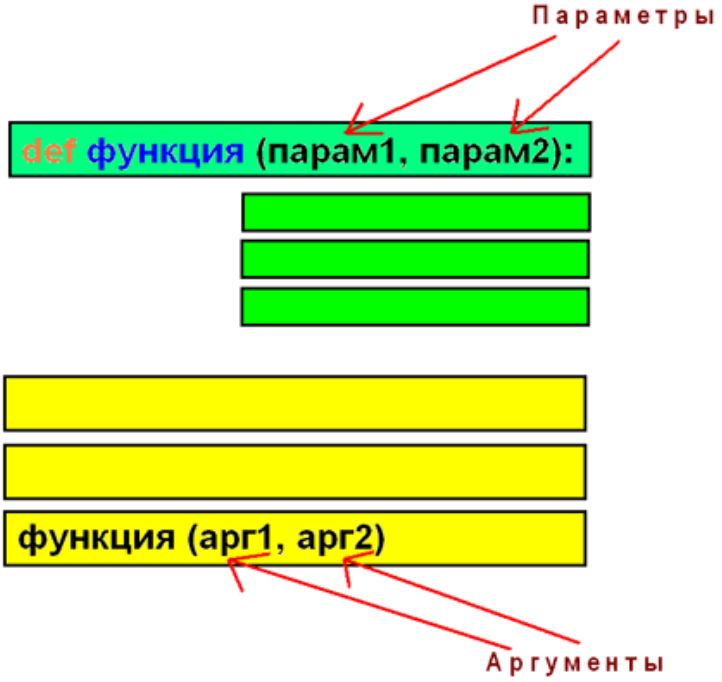


В языке Python определение функции должно предшествовать ее вызовам.

!!!! Код тела функции выполняется при вызове функции, а не при ее первоначальном определении.

Вызов функции обозначается указанием ее имени с последующей парой круглых скобок, которые могут содержать некоторое количество аргументов. Когда программа в процессе выполнения достигает вызова функции, управление передается первой строке кода этой функции, после чего выполняется весь ее код. По достижении конца функции управление возвращается строке, которая ее вызвала, и продолжается выполнение дальнейшего кода.

Функции могут передаваться другим функциям в виде аргументов, сохраняться в структурах данных и возвращаться функциями в виде результата.



Оператор return используется для возврата из функции, т.е. для прекращения её работы и выхода из неё. При этом можно также *вернуть некоторое значение* из функции.
Пример: Функция maximum возвращает максимальный из двух параметров, которые передаются ей при вызове.

```
def maximum(x, y):
    if x > y:
        return x
    elif x == y:
        return 'Числа равны.'
    else:
        return y
print(maximum(2, 3))
```

!!!! Отсутствие оператора return или оператор return без указания возвращаемого значения эквивалентно выражению return None.

Можно возвращать из функции несколько объектов, перечислив их через запятую после команды return:

```
def cyl():
    …
    return a,b          # Из функции возвращаются два значения
s, f = cyl()            # Первое из них присваивается s, второе – f
```

Параметры указываются в скобках при объявлении функции и разделяются запятыми.

О параметрах и переменных, получающих значения в теле вызванной функции, говорят, что они существуют в *локальной области видимости* этой функции. О переменных, значения которым присваиваются вне функций, говорят, что они существуют в *глобальной области видимости.*

Зарезервированное слово ***global*** используется для того, чтобы объявить, что X - это глобальная переменная, а значит, когда мы присваиваем значение имени X внутри функции, это изменение отразится на значении переменной X в основном блоке программы.

Существуют четыре правила, позволяющие судить о том, в какой области видимости находится переменная — локальной или глобальной.

1. Если переменная используется в глобальной области видимости (т.е. вне какой-либо функции), то она всегда является глобальной переменной.
2. Если переменная была объявлена в функции с использованием инструкции global, то она является глобальной.
3. В противном случае, если переменная используется в операции присваивания в функции, то она является локальной.
4. Но если переменной нигде в функции не присваивается значение, то она является глобальной.

Рекомендуется не изменять значения глобальных переменных внутри функции. Если ваша функция должна поменять какую-то переменную, пусть лучше она вернёт это значением, и вы сами при вызове функции явно присвоите в переменную это значение. Если следовать этому правилу, то функции получаются независимыми от кода, и их можно легко копировать из одной программы в другую.

!!!! Значениями по умолчанию могут быть снабжены только параметры, находящиеся в конце списка параметров (параметр со значением по умолчанию не может предшествовать параметру без значения по умолчанию).

```
можно
def func(a, b=5)
```

```
нельзя
def func(a=5, b)
```

Доступ к ***строке документации функции*** можно получить с помощью атрибута этой функции (т.е. имени, принадлежащего ей) __doc__ (обратите внимание на двойное подчёркивание**!!!!**):

```
имяФункции.__doc__
```

Переменное число параметров

Если перед последним аргументом в определении функции добавить символ звездочки (*), функция сможет принимать переменное число аргументов. Т.е. функция может быть определена так, что в нее можно передать хоть ни одного аргумента, хоть множество:

```
def oneOrMany(*a):
    print(a)
oneOrMany(1)
oneOrMany('1',1, 2, 'abc')
oneOrMany()
Результат:
(1,)
('1', 1, 2, 'abc')
()
```

Модули

Модули в языке Python могут присваиваться переменным, помещаться в структуры данных, такие как списки, и передаваться между частями программы в виде элемента данных.

Модуль должен находиться либо в том же каталоге, что и программа, в которую мы импортируем его, либо в одном из каталогов, указанных в sys.path.

Пример: простого модуля (сохраните как mymodule.py)

```
def sayhi():
    print('Привет! Это  мой модуль.')
    __version__ = '0.1'
# Конец модуля mymodule.py
```

Ещё один модуль (сохраните как mymodule_demo.py):

```
import mymodule
mymodule.sayhi()                                    # Привет! Это мой модуль.
print ('Версия', mymodule.__version__)     # Версия 0.1
```

Импортирование отдельных имен из модуля с помощью from..import (сохраните как mymodule_demo2.py):

```
from mymodule import sayhi, __version__
sayhi()                                  #имя модуля указывать не нужно!
mymodule.sayhi()                         # NameError
print('Версия', __version__)
```

Если в модуле, импортирующем данный модуль, уже было объявлено имя __version__, возникнет конфликт.

!!!!Поэтому всегда рекомендуется отдавать предпочтение оператору import.

```
from mymodule import *
```

Это импортирует все публичные имена, такие как sayhi, но не импортирует __version__, потому что оно начинается с двойного подчёркивания.

Добавление пути поиска модулей

Пример: создано два модуля, foo.py и и bar.py, их поместили в zip-файл с именем mymodules.zip, этот файл можно будет добавить в путь поиска Python:

```
import sys
sys.path.append("/tmp/mymodules.zip")
import foo, bar
```

Использование псевдонимов

Если название модуля слишком длинное, или оно вам не нравится по каким-то другим причинам, то для него можно создать псевдоним, с помощью ключевого слова as.
Пример:

```
import spam as sp
import socket as net
sp. foo()
sp.bar()
net.gethost name()
```

Когда модуль загружается под другим именем, как в примере, новое имя можно использовать только в файле или в контексте, где находится инструкция import. Другие модули по-прежнему могут загружать модуль под его оригинальным именем.

Возможность изменения имени импортируемого модуля может оказаться полезным инструментом, позволяющим писать расширяемый программный код.
Пример: есть два модуля, xmlreader.py и csv-reader.py, каждый из которых объявляет функцию read_data(filename), которая читает некоторые данные из файла, но работает с файлами разных форматов.
Программный код выбирает необходимый модуль чтения:

```
if format = 'xml':
    import xmlreader as reader
elif format == 'csv':
    import csvreader as reader
data =  reader.read_data(filename)
```

В языке Python отсутствует полноценная поддержка возможности ***повторной загрузки или выгрузки модулей***, импортированных ранее. Единственная возможность - это полностью перезапустить процесс интерпретатора Python.

Пакеты

Пакеты – это способ иерархической организации модулей. Каждый пакет содержит специальный файл __init__.py, который показывает Python, что этот каталог особый, так как содержит модули Python.
Например, пакет может иметь такую структуру:

```
Graphics/
    __init__.py
    Primitive/
        __init__.py
        lines.py
        fill.py
        text.py
        ...
    Graph2d/
        __init__.py
        plot2d.py
        ...
    Graph3d/
        __init__.py
        plot3d.py
        ...
    Formats/
        __init__.py
        gif.py
        png.py
        tiff.py
        jpeg.py
```

Загрузка модулей из пакета может быть выполнена с помощью инструкции import несколькими способами:

- `import Graphics.Primitive.fill`

Загрузит модуль Graphics.Primitive.fill. Объекты, содержащиеся в этом модуле, получат имена, такие как Graphics.Primitive.fill.floodfillCimg.x.y.color).

- `from Graphics.Primitive import fill`

Загрузит модуль fill, но сделает возможным обращаться к нему без использования префикса с именем пакета; например, fill.floodfillCimg.x.y.color).

- `from Graphics.Primitive.fill import floodfill`

Загрузит модуль fill, но сделает возможным обращаться к функции floodfill непосредственно; например, floodfillCimg.x.y.color).

Контрольные вопросы:

1. Что дает использование функций в программах?
2. Когда именно выполняется код функции: когда она определяется или когда вызывается?
3. С помощью какой инструкции создаются функции?
4. Чем отличается определение функции от ее вызова?
5. Сколько глобальных областей видимости может иметь программа на языке Python? Сколько локальных?
6. Что происходит с переменными, находящимися в локальной области видимости, при возврате из функции?
7. Что такое возвращаемое значение? Может ли возвращаемое значение быть частью выражения?
8. Каково возвращаемое значение функции, если в ней отсутствует инструкция return?
9. Как заставить переменную в функции ссылаться на глобальную переменную?

Задания

Задание 1 (В1-В13)

Отладить сценарий, который получает текст из буфера обмена, добавляет звездочку и пробел в начале каждой строки, а затем возвращает этот новый текст в буфер обмена.

Этапы выполнения:

1. Установить модуль pyperclip с сайта PyPI
   Помимо стандартной библиотеки модулей, входящей в поставку Python, можно использовать модули сторонних разработчиков.
   Основным средством для установки модулей Python, разработанных другими лицами, является утилита **pip,** включенная в современные версии Python. Она обеспечивает безопасную загрузку и установку модулей Python, доступных на сайте организации Python Software Foundation no адресу
   https://pypi.python.org/.
   Сайт PyPI (от англ. Python Package Index — каталог пакетов Python) играет роль своего рода "магазина бесплатных приложений" для Python.
   Об установке:
   https://packaging.python.org/tutorials/installing-packages/
2. Следуя рекомендациям (файл «модуль_pyperclip.pdf»), разработать сценарий

Задание 2 (В1-В13)

Напишите функцию collatz (), принимающую один параметр: number. Если number — четное число, функция collatz () должна вывести на экран и возвратить значение number // 2. Если же number— нечетное число, то функция должна вывести на экран и возвратить значение 3 * number + 1.

После этого напишите программу, которая предлагает пользователю ввести целое число, а затем последовательно вызывает функцию collatz () для этого числа и значений, возвращаемых очередным вызовом этой функции, пока на каком-то этапе не будет возвращено значение 1. (Любопытно отметить, что, независимо от выбора начального числа, вы все равно рано или поздно получите 1! Даже математики не могут объяснить, почему так происходит. Числовая последовательность,

которую вы исследуете с помощью этой программы, называется *последовательностью Коллатца* и иногда характеризуется как "простейшая из неразрешенных проблем математики".)

Предусмотреть проверку корректности ввода с помощью инструкции try и except.

Не забывайте о том, что возвращаемое функцией input () значение нуждается в преобразовании в целое число с помощью функции int (), иначе это будет строковое значение.

Для проверки условия четности значения — number % 2 == 0, нечетности — number % 2 == 1.

```
Например, пользователь ввел 3
Получим:
10
5
16
8
4
2
1
```

Задание 3

B1: Пользователь вводит длину и ширину прямоугольника. Напишите функцию, которая проверяет, что больше его площадь или периметр, и выводит большее значение с сообщением «Периметр больше и равен…». Если оба значения равны, выводится сообщение «Периметр и площадь равны». Предусмотреть обработку исключений.

B2: Пользователь вводит 2 строки. Напишите функцию, которая находит число совпадений букв из первой строки со второй строкой. Например, ввели строку «1234abcd» и «1234da», получили 4 (т.е. совпали 4 первых символа).

B3: Написать функцию stepen (a,n), которая вычисляет a^{n^3} (предусмотреть проверку: *a* - положительное число и *n* - целое число).

B4: Напишите программу, в которой пользователь задает четыре числа *a*, *b*, *c* и *d*. Программа должна вывести фрагмент таблицы умножения для всех чисел отрезка [*a*;*b*] на все числа отрезка [*c*;*d*].

Например, пользователь ввел 7,10,5,6, программа вывела

	5	6
7	35	42
8	40	48
9	45	54
10	50	60

B5: Пользователь вводит два числа A и B. Функция суммирует все целые числа от A до B включительно. Если пользователь задаст первое число большее чем второе, поменяйте их местами и посчитайте.

B6: Пользователь вводит N. Функция вычисляет $\frac{2}{3}+\frac{3}{4}+\frac{4}{5}+\ldots+\frac{9}{N}$.

Предусмотреть обработку ситуации ввода букв вместо цифр.

B7: Пользователь вводит строку. Напишите функцию, которая находит число совпадений букв из заданной строки с началом алфавита (абвгдеёжзиклмн…). Например, ввели «абвгдаа», получили 5.

B8: Напишите функцию, которая вычисляет значения $f(x)= (\sin x)^2+ x^3+10$ на отрезке [a;b] с шагом h. Результат представить в виде (если a=3, b=10, h=3):

значение x	значение y
3	…
6	…
9	…

B10: Дана строка и два маркера (начальный и конечный). Написать функцию, которая находит текст, заключенный между двумя этими маркерами. Начальный и конечный маркеры всегда разные, один символ, всегда есть в строке. Функция принимает три аргумента: все строки, второй и третий аргументы это начальный и конечный маркеры.

B11: Пользователь вводит строку. Напишите функцию, которая выводит первое слово из строки.

В12: Пользователь вводит два целых числа h - часы и m – минуты. Функция определяет сколько осталось минут до Нового года. Предусмотреть проверку: $0 \le h < 24$, $0 \le m < 60$.
Например, пользователь ввел 23 55, получили 5

В13: Напишите функцию, в которую передается курс доллара и количество, а функция переводит в рубли:

```
Введите текущий курс доллара: 2.84
Введите сумму белорусских рублей, которые вы хотите обменять: 100
100 бел.руб. можно обменять на 35 долларов, сдача 60 копеек
```

Задание 4 (В1-13)

Вынесите все функции в отдельный модуль, который будет подключаться к основным программам.