Инструментальные средства информационных систем

Автор: Сафиуллина Н.Ф.

Лабораторная работа 2 Hello Python

Python – это высокоуровневый язык программирования, который:
– является интерпретируемым,
– является объектно-ориентированным,
– имеет динамическую строгую типизацию,
– имеет автоматическое управление памятью,
– блоки кода выделяет за счет отступов (пробелов или табуляций),
– имеет “генератор списков” (list comprehensions),
– имеет “выражения-генераторы” (generator expressions).

Цель: познакомиться со скриптовым языком Python и решить 15 задач, выложить получившийся скрипт на GitHub.

Рекомендации:

Решая задачи в интерактивной режиме Python, сохраните весь текст из рабочего окна в текстовый файл, чтобы потом можно было выложить его на GitHub.

Пишите в коде комментарии, комментарий вводится # или ‘’’ (три одинарные кавычки).

Вы можете проверять работу коротких кусков кода через интерактивный режим, готовое решение пишите в PyCharm. Пишите решения всех задач в одном скрипте разделяя их комментарием к какой задаче относится код.

Результат на проверку:

код решения всех задач начиная со 2 по 15/16 в файле .py выложите в свой аккаунт на GitHub (все задания в одном файле), каждое решение должно иметь комментарий, в котором указан номер задания и пояснено решение.

Содержание:

Гвидо Ван Россум, создатель Python.

1 Установка Python

1.1 Установка интерпретатора

Python часто применяется для написания скриптов, а скрипты часто требуется запускать в автоматическом режиме, для того чтобы переводить наш код в исполняемые команды установим интерпретатор.

Скачать интерпретатор можно тут: https://www.python.org/downloads/

Оставляем галочку “Install launcher for all users” как она есть.

Ставим галочку “Add Python 3.10 to PATH” чтобы установщик добавил путь в PATH, опция нужна для того, чтобы появилась возможность запускать интерпретатор без указания полного пути до исполняемого файла при работе в командной строке.

Запускаем установку, выбираем “Install Now” (рисунок 1).

Установщик по умолчанию содержит интерпретатор, IDLE Python (интегрированная среда для разработки и обучения), pip (систему управления пакетами) и документацию.

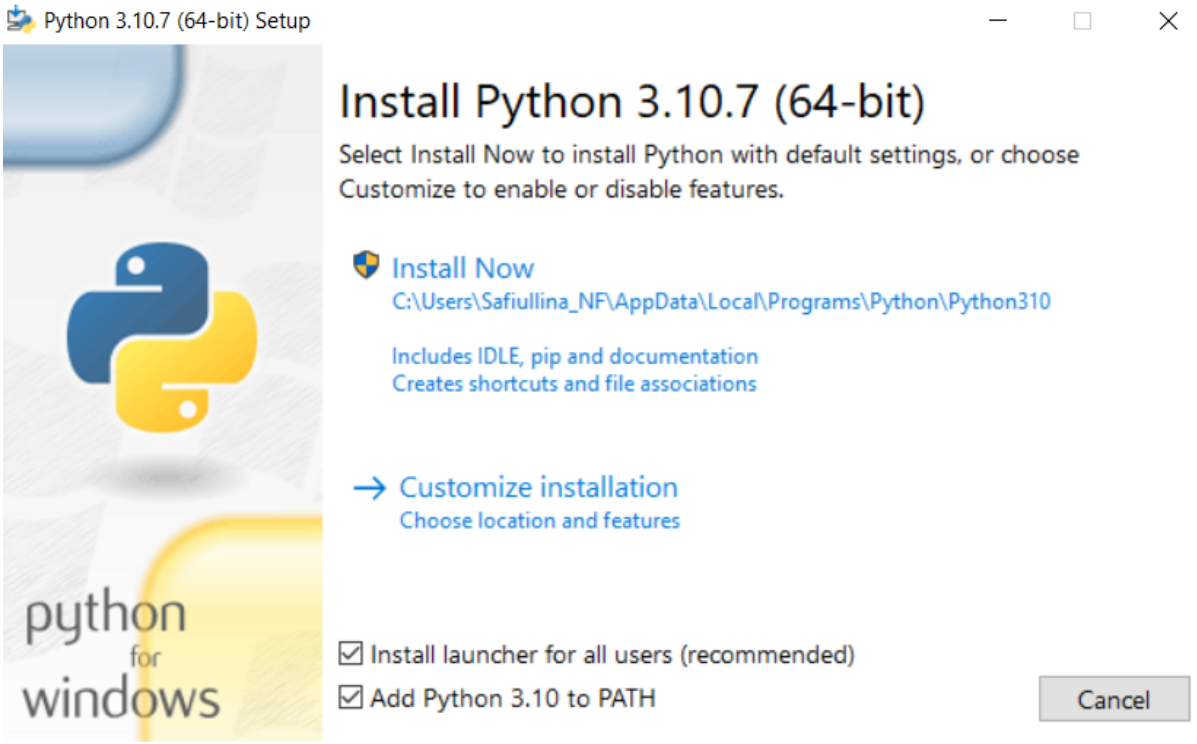

Рисунок 1 Установка интерпретатора Python

После завершения установки проверим работает ли интерпретатор. Запустите системную командную строку, вводим:

```
D:\>python --version
Python 3.10.7
```

Если команда сработала и мы видим версию, значит будем считать установка прошла успешно.

1.2 Использование интерпретатора

Попробуем различные способы взаимодействия с интерпретатором.

Первый способ – через командную строку:

1. Определитесь где у вас будут лежать скрипты для Python, через проводник Windows перейдите туда, создайте нужные папки, создайте там текстовый файл (можно через блокнот) с одной строкой внутри **print('Hello world')**, сохраните файл, поменяйте расширение файла на .py, должно получиться примерно как на рисунке 2.

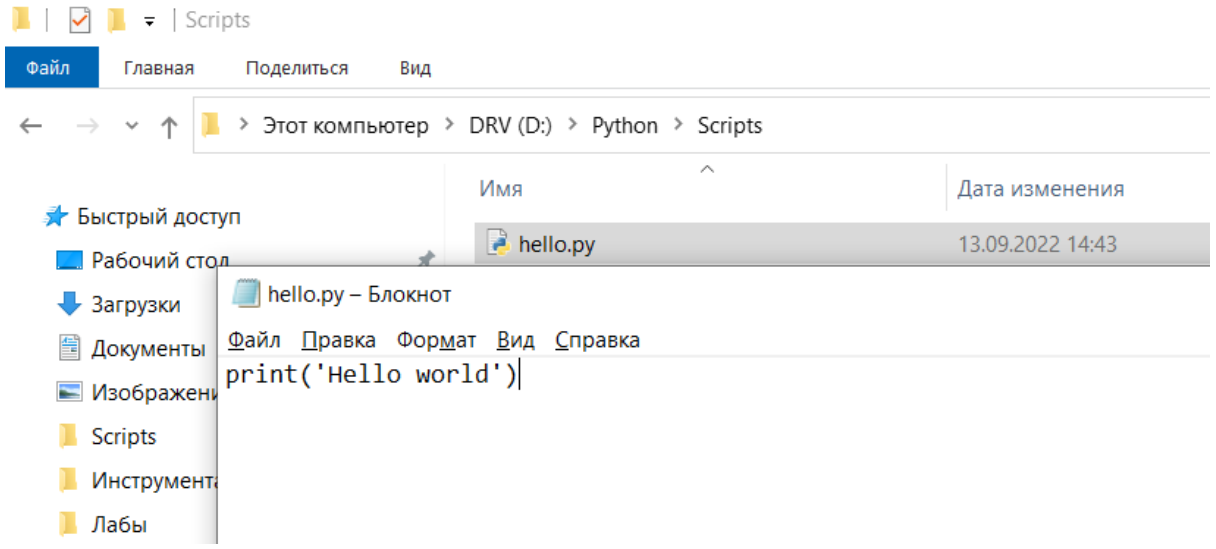

Рисунок 2 Создание скрипта в блокноте

2. В командной строке перейдите в папку со скриптом.
3. В командной строке передаем наш скрипт интерпретатору, пишем слово python пробел и имя файла скрипта:

```
D:\Python\Scripts>python hello.py
```

```
Hello world
```

Готово.

Второй способ – через интерактивный режим Python:

1. В командной строке пишем python и жмём Enter
2. Откроется интерактивный режим Python, вводим print('Hello world') и жмём Enter, интерпретатор выполнит скрипт:

```
D:\Python\Scripts>python
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC
v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>> print('Hello world')
Hello world
>>>quit()
```

3. Чтобы выйти из интерактивного режима набираем quit() и жмём Enter.

Задача 1.

Попробуйте ввести вместо print(...) просто математические операции, например 2+2, что получается? Что получится если произвести деление 37 на 10 через "/", через "//" и через "%"?

1.3 Установка IDE PyCharm

Перед установкой IDE должен быть установлен интерпретатор, так будет лучше. Скачать PyCharm от JetBrains можно тут:

https://www.jetbrains.com/ru-ru/pycharm/

Оставляем всё по умолчанию, галочки не ставим (рисунок 3).

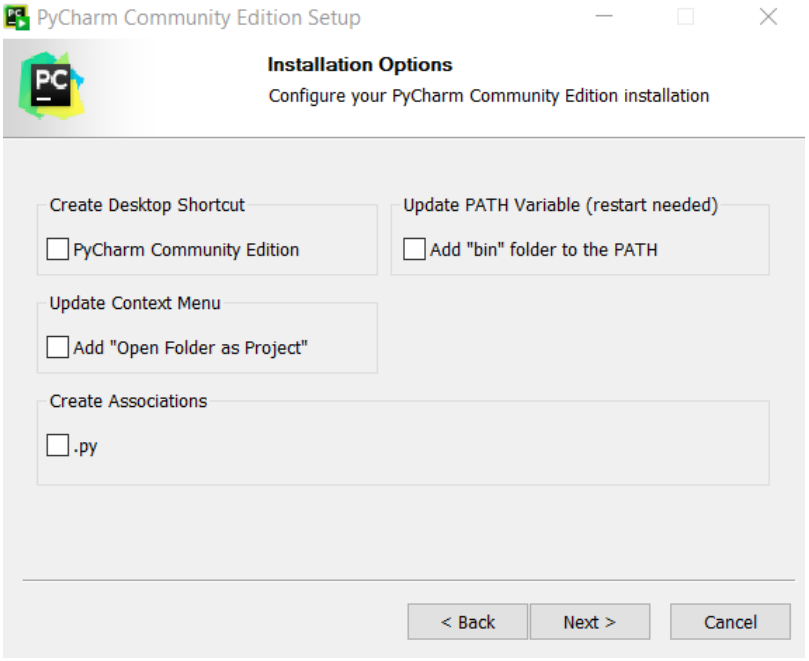

Рисунок 3 Опции установки PyCharm

1.4 Использование IDE PyCharm

После установки PyCharm запускаем её. На странице приветствия, где написано “Welcome to PyCharm”, выбираем “New Project”, далее потребуется сконфигурировать первый проект.

1. В поле Location укажите путь к вашему будущему проекту (определитесь где у вас будут лежать проекты Python, через проводник Windows перейдите туда, создайте папку для проекта).
2. В поле Base Interpreter автоматически должен быть указан путь к интерпретатору установленному ранее.
3. Нажимаем кнопку Create, PyCharm создаст проект и файл main.py, где будет представлен пример кода и пример описания и вызова функции.
4. Запустить main.py можно как показано на рисунке 4, результат выполнения видно в нижней части окна.

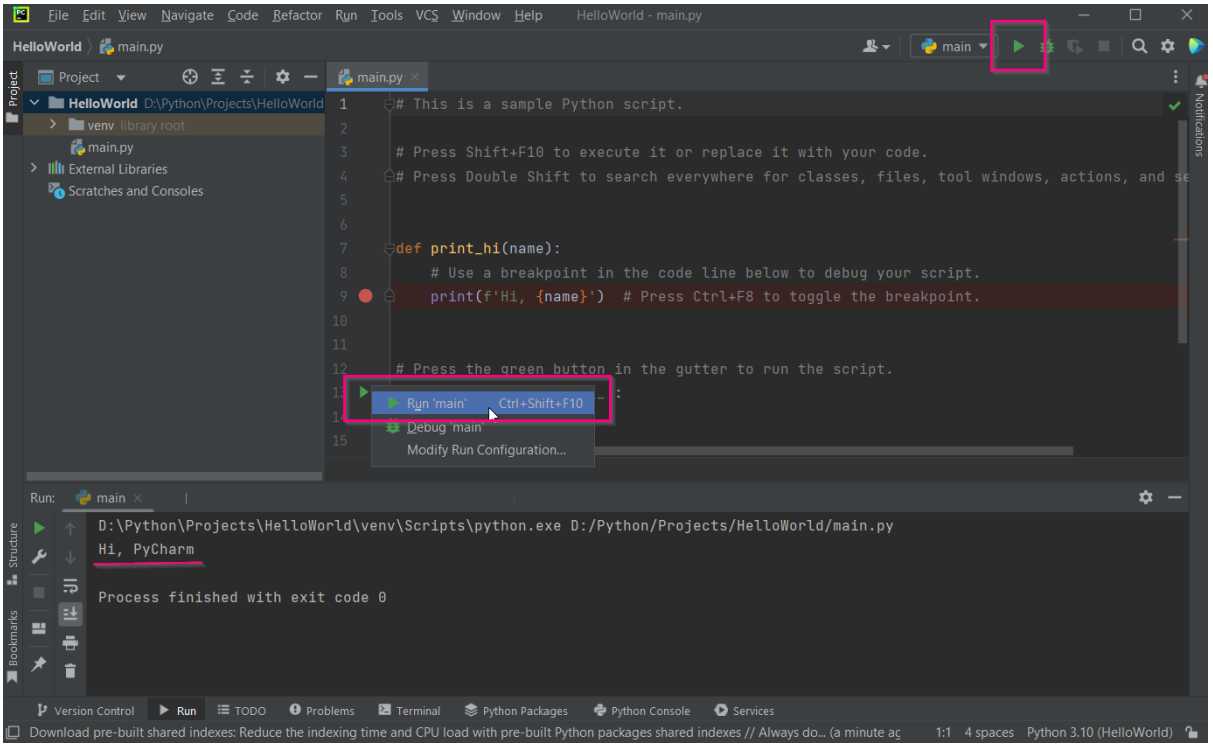

Рисунок 4 Запуск скрипта на выполнение

Задача 2.

Поменяйте код так, чтобы скрипт выводил “Hello world”.

2 Типы данных, ввод/вывод и условия

2.1 Типы данных

Стандартные типы:

`int` – целочисленный тип,

`float` – вещественный,

`bool` – логический,

`str` – строчный.

Функции приведения типов:

`int()` – возвращает целое число,

`float()` – возвращает число с плавающей точкой.

Функция получения типа объекта:

`type()` – возвращает тип объекта.

Строки. Рассмотрим подробнее строки.

Строка – это набор символов, каждый символ в строке имеет свой порядковый номер (индекс), который начинается с нуля.

Например есть строка:

```
s = 'Python'
```

то

`print(s[2])` – выдаст “t”.

Строки имеют следующие методы:

`s.upper()` – возвращает символы строки в верхнем регистре.

`s.lower()` – возвращает символы строки в нижнем регистре.

`s.count('c')` – подсчитывает сколько раз в строке встретился символ ‘c’.

`s.find('c')` – возвращает первое вхождение символа ‘c’.

`s.replace('c','r')` – возвращает строку с измененными символами ‘c’ на ‘r’.

Функция

`len(s)` – возвращает длину строки, списка, множества и словаря.

Далее мы рассмотрим **списки, множества и словари**, которые имеют много методов работы с ними, но мы не будем изучать их подробно. При необходимости можете изучить их самостоятельно.

Списки.

Список – это набор значений, элементами списка могут быть любые значения, каждое значение в списке имеет свой порядковый номер (индекс).

Например есть список:

```
friends = ['Dasha', 'Masha', 'Pasha']
```

то

`print(friends[2])` – выдаст “Pasha”.

Списки могут быть двумерными, т.е. это некая матрица n*m, например матрица 3*3:

1 2 3

4 5 6

7 8 9

Создание такого списка и заполнение можно выполнить следующим образом:

```
a = [[0,0,0],[0,5,0],[0,0,0]]
```

или с помощью генераторов списков (list comprehensions) заполним нулями, n переменная с любым значением, у нас n=3:

```
a = [[0 for j in range(n)] for i in range(n)]
```

Обращение к элементу такого списка возможно по двойному индексу, например, поменяем значение и выведем на экран:

```
a[1][1] = 5
print(a[1][1], end='\n\n')
```

Получим значение 5.

Вместо ноля мы можем использовать любые выражения или функции:

```
print('--- Sample matrix ---')
n = 3
a = [[i * n + j for j in range(n)] for i in range(n)]
print(a)

a = [[int(input()) for j in range(n)] for i in range(n)]
print(a)
```

Множества.

Множество или set – это набор уникальных значений.

`s = set()` – создание пустого множества.

`s = {'Dasha', 'Masha', 'Dasha'}` – создание множества со значениями.

Словари.

Словарь или dictionary – это набор пар <ключ:значение>, позволяет по ключу получать значение, ключи должны быть уникальными, значениями могут быть списки значений.

`d = dict()` – создание пустого словаря.

`d = {'name':'Ann', 'year': 2001, 10:100}` – создание словаря со значениями.

Получить значение по ключу можно:

`print(d['name'])` – вернёт ‘Ann’.

Задача 3.

Введите:

```
a = 3
print(type(a))
a = 3.5
print(type(a))
a = 'qwerty'
print(type(a))
a = True
print(type(a))
a = '123'
print(type(a))
```

Какие типы объектов мы видим? Что будет, если в конце попытаться к `a` прибавить число?

Задача 4.

Можно использовать интерактивный режим Python или PyCharm, это не имеет значения.

а) Приведите к целому типу число 5.7.

б) Приведите к целому типу число -5.7.

в) Вычислите значение `3**39 - int(float(3**39))` (** – возведение в степень). Что получилось?

2.2 Стандартные ввод/вывод

Присвоение какого-то значения в переменную:

```
a = 3
a = 'Hello'
```

где `a` – это имя переменной.

Так как Python имеет динамическую типизацию, то нельзя сказать, что переменная имеет какой-то тип. В одну и ту же переменную мы можем сохранить и число и строку. Получается, данные имеют какой-либо тип, а сама переменная нет.

Ввод

`input()`

– функция чтения с клавиатуры.

`a = input('Введите значение:')`

– ввод с приглашением и сохранение того, что ввели, в переменную.

`a = int(input('Введите число:'))`

– ввод значения и приведение значения к целому числу (любой ввод через input выдаёт строку).

К данным, которые ввели с клавиатуры, можно применить метод split(). Синтаксис:

`split([separator], [maxsplit])`

– разделяет строку по указанному separator (разделитель) на указанное количество частей в maxsplit, если разделитель не указан, то разделителем является пробел, если количество частей не указано, то разделит на всевозможные части.

Например:

```
a, b = input('Введите значения через пробел: ').split()
```

Вводим с клавиатуры два значения через пробел и нажимаем Enter:

```
Один Два
```

Если вывести результат:

```
print(a)
print(b)
```

Получим:

```
Один
Два
```

Вывод

`print()`

– функция вывода объекта на экран.

`print('Переменная a равна ', a)`

– выводить данные можно через запятую.

У print есть параметр end, где мы можем указать чем закончить вывод данных (‘\n’ – перевод строки, ‘\t’ – табуляция, ‘ ‘ – поставить пробел), например:

```
>>> a = 'Какой-то текст'
>>> print(a, end=' ')
Какой-то текст >>> print(a, end='\n')
Какой-то текст
>>>
```

Задача 5.

Напишите программу, которая будет запрашивать имя пользователя и затем будет приветствовать его.

Задача 6.

Доктор Иванов добирается на машине X часов до частной клиники, где работает. После обеда он идёт работать в поликлинику и доходит за Y минут. Напишите программу, которая будет рассчитывать сколько минут доктор Иванов проводит в дороге.

Рекомендации: Программа принимает значения X и Y со стандартного потока через input() и выводит результат через print().

Пример работы программы:

```
--- Task 6 ---
X= 1
Y= 20
Итого минут:  160
```

Задание со звёздочкой: Напишите эту программу, используя выражения-генераторы, в одну строку.

2.3 Условия

Для использования условий потребуется вспомнить логические операции: `not, and, or` (операции приведены в порядке приоритета выполнения).

Первый вид условной конструкции:

```
if <логическое условие>:
<действие 1>
else:
<действие 2>
```

– условная конструкция, если условие истинно – выполняется действие 1, иначе – выполняется действие 2.

Второй вид условной конструкции:

```
if <логическое условие 1>:
     <действие 1>
elif <логическое условие 2>:
     <действие 2>
…
else:
     <действие n>
```

– если условие 1 не выполнится, проверяем условие 2, если условие 2 выполняется – выполняем действие 2; если условие 2 не выполняется, то проверяется условие 3, если оно есть и т.д., если ни одно из условий не выполняется – выполняем действие под else.

Задача 7.

Присвойте переменным следующие значения:

a = False

b = True

c = False

Выполните следующее выражение:

not a or b and c

Что получилось? Расставьте скобки так, чтобы получить False.

Задача 8.

У доктора Иванова день рождение 29 февраля. Он хочет знать в какие года с 1900 по 3000 он может праздновать свой день рождения. Напишите программу, которая получает на вход год и:

– если год високосный, то выводит сообщение “С днём рождения!”,

– если год меньше 1900 или больше 3000, сообщает “Год не входит в выборку”,

– иначе выводит сообщение “Год обычный”.

Пример работы программы:

```
Введите год: 2000
С днём рождения!
```

Рекомендация: Високосными годами считаются те годы, порядковый номер которых либо кратен 4, но при этом не кратен 100, либо кратен 400 (например, 2000-й год являлся високосным, а 2100-й будет невисокосным годом).

3 Циклы

Циклы, как известно, это конструкции позволяющие выполнять одну и ту же последовательность действий множество раз. Каждый проход цикла называется **итерацией**.

Циклы могут быть вложенными.

3.1 Цикл while

Общая схема:

```
while <логическое условие>:
    <действие 1>
    <действие 2>
    …
    <действие n>
else:
    <действие после цикла>
```

– логическое условие проверяется перед каждой итерацией цикла, действия выполняются, если логическое выражение истинно.

Действие после цикла указанные в else выполняется, если цикл был завершен без досрочного завершения.

Не забывайте менять переменные участвующие в условии внутри цикла, чтобы логическое условие когда-нибудь стало ложным, иначе получится бесконечный цикл.

Задача 9.

Напишите программу, которая выводит все чётные числа от 1 до 20 в одну строку через пробел, используя цикл while.

Ниже приведен один из вариантов решения. Попробуйте написать код самостоятельно и затем сравните решения.

Пример работы программы:

```
--- Task 9 ---
2 4 6 8 10 12 14 16 18 20
```

Сам код программы:

```
print('--- Task 9 ---')
a = 1 #инициализация переменной начальным значением
while a <= 20: #логическое условие цикла
   if a%2 == 0: #если остаток от деления на 2 равен нулю, то число четное
       print(a, end=' ') #вывести значения и в конце добавить пробел
   a += 1 #инкремент, следующее значение
print('\n\n')
```

Задача 10.

Напишите программу, которая считывает со стандартного ввода целые числа, по одному числу в строке, и после первого введенного нуля выводит сумму полученных на вход чисел.

Пример работы программы:

```
 --- Task 10 ---
Введите число: 2
Введите число: 4
Введите число: 6
Введите число: 8
Введите число: 0
Итого: 20
```

Задача 11.

Доктор Иванов 29 февраля испек дома огромную пиццу, чтобы угостить своих коллег. Но он не знает заранее куда его сегодня вызовут на работу, если в клинику, то там будет X коллег, а если в поликлинику – то Y коллег. Он хочет разрезать пиццу дома на столько кусочков, чтобы каждому досталось поровну, но при этом кусочки должны быть максимально большие.

Напишите программу, которая получает на вход два значения X и Y и вычисляет необходимое количество кусочков.

Пример работы программы:

```
--- Task 11 ---
Введите X: 6
Введите Y: 9
Нужно кусков: 18
Пицца порезана
```

3.2 Операторы continue и break

Операторы:

`break` – позволяет досрочно завершить цикл.

`continue` – завершает выполнение текущей итерации, пропускает все действия ниже и переходит к следующей итерации.

Например: создадим цикл у которого условие всегда истинно, в цикле будем спрашивать хочет ли пользователь выйти из цикла, если пользователь ввел ‘Y’, то прерываем цикл (выходим из него), если пользователь ввел ‘N’, то переходим к следующей итерации. Обратите внимание, никогда не выполнится вызовы функций print('Сюда мы никогда не попадем').

```
while 1:
    #запрашиваем значение
    a = input('Выйти из цикла (y/n):').lower()
    if a == 'y':
        #ввели Y, прерываем выполнение цикла
        break
    elif a == 'n':
        #ввели N, переходим к следующей итерации
        continue
        print('Сюда мы никогда не попадем')
    else:
        print('Введено что-то кроме Y или N.')
else:
     print('Сюда мы никогда не попадем тоже')
```

3.3 Цикл for

Общая схема:

```
for <элемент> in <последовательность элементов> :
    <действие 1>
    <действие 2>
    …
```

– действия в теле цикла выполняются для каждого элемента из последовательности элементов.

Например, сложим все числа из заданной последовательности:

```
#сложим числа из заданной последовательности
b = 0   #обнуляем переменную в которой будет накоплена сумма
for a in 2,3,4: #для каждого a из списка
```

```
    b += a          #прибавим a к b
print('Сумма = ', b)
```

С циклом for тесно связана функция range(), которая возвращает последовательность чисел, имеет три параметра:

`range([start], stop[, step])`

`start` – значение, с которого начать, параметр не обязателен, по умолчанию имеет значение 0.

`stop` – последнее значение в последовательности, но не включает его в список.

`step` – шаг в последовательности, параметр не обязателен, по умолчанию равен 1.

Например, сложим все числа от 2 до 4 (включая 4):

```
#сложим числа из заданной последовательности
b = 0   #обнуляем переменную в которой будет сумма
for a in range(2,4+1): #для каждого a из списка
    print(a)    #выведем текущее значение a
    b += a      #прибавим a к b
print('Сумма = ', b, end='\n\n')
```

В качестве последовательности элементов в цикле for можно использовать строки, списки, множества и словари.

Задача 12.

Напишите программу, которая выводит все чётные числа от 1 до 20 в одну строку через пробел, используя цикл for.

Пример работы программы:

```
--- Task 12 ---
0 2 4 6 8 10 12 14 16 18
```

Задача 13.

Напишите программу, которая выводит на экран таблицу умножения. При чем только ту часть таблицы, которая требуется.

Пусть на вход подаются 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].

Ниже приведен один из вариантов решения. Попробуйте написать код самостоятельно и затем сравните решения.

В любом случае, если вы написали сами или скопировали код, разберите его и подпишите комментарии, что делает каждая строка.

Пример работы программы:

```
--- Task 13 ---
Введите значение: 2
Введите значение: 4
Введите значение: 3
Введите значение: 5
Результат:
     3    4    5
2    6    8    10
3    9    12   15
4    12   16   20
```

Рекомендация: для решения этой задачи нам понадобятся вложенные циклы.

```
a, b, c, d = [int(input()) for i in range(4)]
print('', end='\t')
for j in range(c, d + 1):
   print(j, end='\t')
print()
for i in range(a, b + 1):
   print(i, end='\t')
   for j in range(c, d + 1):
       print(i * j, end='\t')
   print()
```

Примечание.

`a, b, c, d = [int(input()) for i in range(4)]`

– это как раз list comprehensions, генератор списков, каждое значение вводится в своей строке.

`a, b, c, d = (int(i) for i in input().split())`

– а это тоже самое, только с помощью generator expressions, выражений-генераторов, значения вводятся через пробел.

Задача 14.

Доктор Иванов очень любит ковры, числа и спирали. Как-то он задумал выткать ковер, поделенный на равные квадраты, и в каждом квадрате должно располагаться одно число, причем числа должны идти по спирали от внешнего края ковра к центру. Ткацкий станок ткет ковер сверху вниз, поэтому необходимо увидеть заранее расположение чисел в квадратах. Напишите программу, которая получает на вход размер ковра в квадратах *n*, пусть ковер будет *n*n*. И затем выводит числа по спирали.

Пример работы программы:

```
--- Task 14 ---
Введите размер матрицы: 4
Результат:
1    2    3    4
12   13   14   5
11   16   15   6
10   9    8    7
```

Рекомендация: понадобится двумерный список и вложенные циклы for.

4 Полезное приложение

4.1 Библиотеки

Библиотека – это модуль, который содержит набор функций, которые можно использовать в своем скрипте подключая к нему весь модуль или отдельные функции.

Python имеет огромное количество библиотек.

Например, в стандартную библиотеку Python входит модуль tkinter, предназначенный для создания графического интерфейса и модуль time, предназначенный для работы с временем, и много других.

Чтобы импортировать модуль в программу используется:

```
import <имя модуля>
```

А для импорта отдельных функций указывается из какого модуля импортировать и имя функции. Например:

```
import time
from tkinter import Tk, messagebox
```

4.2 Использование функций tkinter

Рассмотрим следующую процедуру:

```
# This is a useful Python script.
import time
from tkinter import messagebox


if __name__ == '__main__':
   messagebox.showinfo('Useful Python', 'Вы долго смотрели в
монитор, теперь посмотрите в окно.')
```

Мы импортировали полностью модуль time и из модуля tkinter импортировали messagebox. Вызвали функцию messagebox и её метод showinfo.

Задача 15.

Перепишите программу так, чтобы данное окно появлялось вновь через 10 секунд после его закрытия, и так бесконечно.

Закройте PyCharm. Запустите скрипт из командной строки.

Теперь у вас есть напоминалка об отдыхе. Дело за малым, поменять период ожидания на 1 час и научиться запускать скрипт в фоновом режиме.

Задача 16 (необязательная).

Изучите скрипт ниже, вместо messagebox в нем используется Tk. Теперь можно создавать окна с набором кнопок и полями для ввода. Попробуйте решить задачу 15 использую Tk, сделайте две кнопки, при нажатии на одну из них – окно появится снова через какое-то заданное время бесконечно, а при нажатии на другую – произойдет завершение (quit()) программы.

```
from tkinter import *

#функция обработки нажатия
def clicked():
   lbl.configure(text='Кнопка была нажата',font=("Arial Bold", 30))

window = Tk() #создание окна
window.title('Окна и кнопки') #заголовок окна
window.geometry('400x250') #размеры окна
lbl = Label(window, text='Кнопка', font=('Arial Bold', 30))
lbl.grid(column=0, row=0)

#вызов функции clicked() при нажатии кнопки
btn = Button(window, text='НАЖМИ', command=clicked)

btn.grid(column=1, row=0)
window.mainloop() #бесконечный цикл окна, окно ждёт нажатий
```

Уроки по tkinter: https://pythonru.com/uroki/obuchenie-python-gui-uroki-po-tkinter