



Модели разработки программ. Структурное программирование.

Базовые принципы:

- * **блочная структура кода – блоки и подпрограммы.**
- * **типовые структуры управления – последовательность, ветвление, цикл**

Турашова Анна Николаевна
Преподаватель
anna1turashova@gmail.com
Telegram: @anna1tur



Блок кода

Блок кода - (также говорят блок команд, блок инструкций) в программировании — это логически сгруппированный набор идущих подряд инструкций в исходном коде программы.

В Python блок кода задается при помощи отступов: пробелами или символами табуляции.

Инструкция - наименьшая автономная часть языка **программирования**; команда или действие.

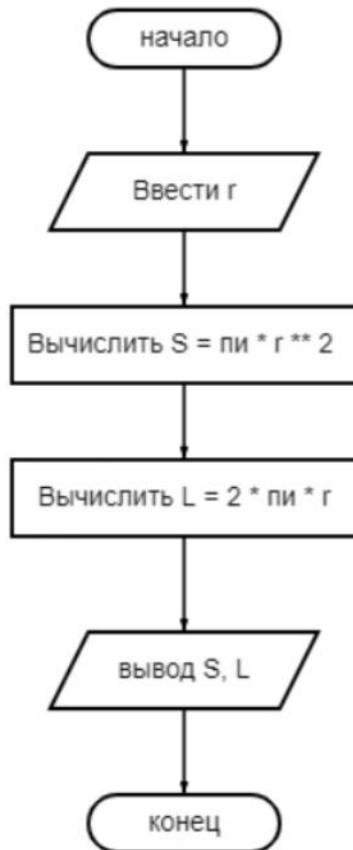
```
if paddle.colliderect(ball):
```

```
    ball_xdir *= -1  
    ball_ydir *= -1
```

← Блок кода

Следование

- Последовательное выполнение - однократное выполнение действий (операторов) в том порядке, в котором они записаны в тексте программы





Знаки отношений

>

<

больше, меньше

≥

больше или равно

≤

меньше или равно

==

равно

!=

не равно

Операторы ветвления

Условный оператор if - называемый также оператором выбора, позволяет в зависимости от истинности или ложности некоторого условия выполнять ту или иную последовательность команд в блоке кода.

```
age = int(input("Введите возраст: "))
```

```
if age >= 18:  
    print("Совершеннолетний")
```

```
print("Пока!")
```

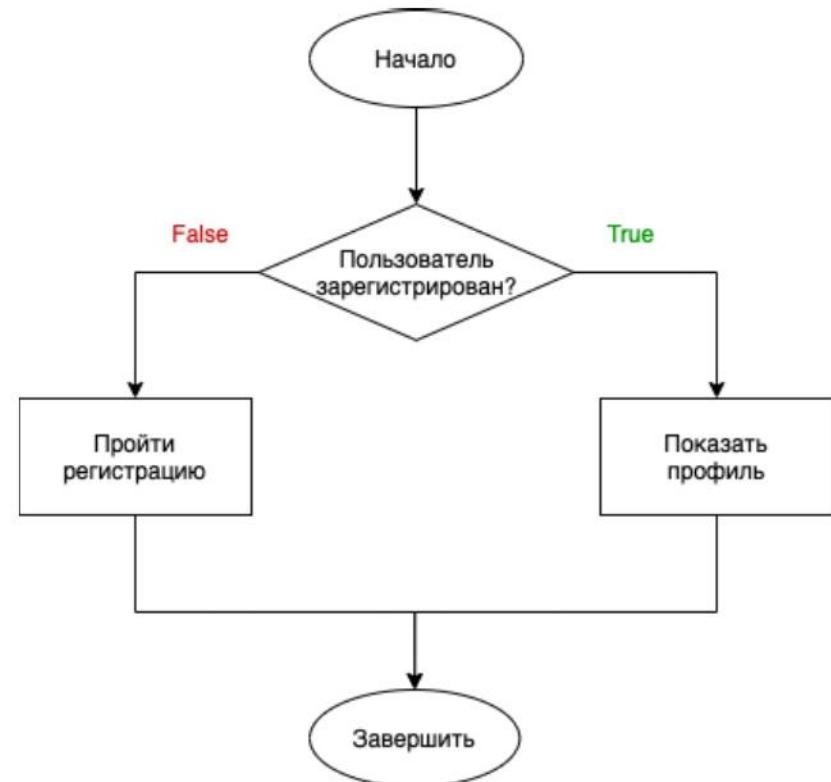


Оператор if-else

Оператор **if** выполняет блок кода, если указанное условие выполняется (истинно). Если условие не выполняется (ложно), то в противном случае выполняется блок кода внутри **else**.

```
registered = True
```

```
if registered:  
    print("Показать профиль")  
else:  
    print("Пройти регистрацию")  
  
print("До встречи!")
```





Задачи

Определить равенство трех чисел

1. На входе получить три числа и сохранить в переменные x , y , z
2. Если x равно y равно z , то сообщить о равенстве

Иначе сообщить о неравенстве

Определить число четное или нечетное?

1. Пользователь вводит целое число
2. Если число делится на 2 без остатка, то сказать четное

Иначе нечетное

Проверить делимость одного целого числа на другое

1. Пользователь вводит числа a и b
2. Если a делится на b без остатка, то число a делится нацело на b

Иначе, в остальных случаях, есть остаток и он не равен нулю.

Следовательно, a не делится на b

Программа для расчета площади и длины окружности круга

1. Пользователь вводит радиус
2. Если радиус больше или равен 0, то

Вывести чему равна длина окружности и площадь круга

Иначе

Попросить ввести положительное число

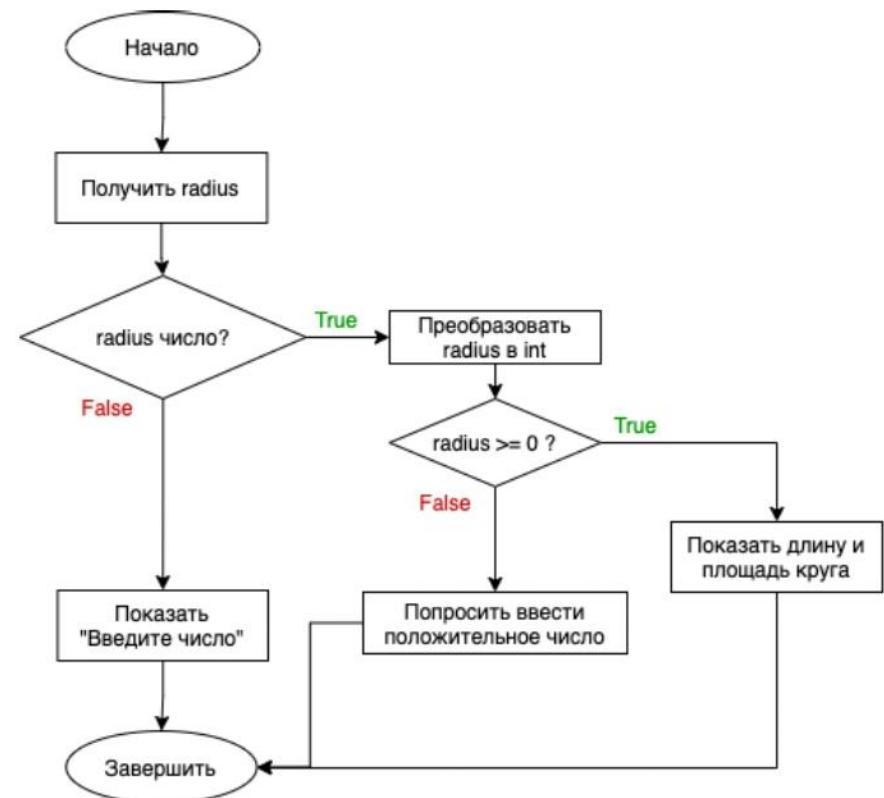
Оператор if внутри другого if

Если требуется больше проверок, то можно один оператор if поместить внутри другого оператора if

```
input_radius = input("Введите радиус: ")

if input_radius.isdigit():
    radius = int(input_radius)

    if radius >= 0:
        print("Длина окружности = ", 2 * 3.14 * radius)
        print("Площадь = ", 3.14 * radius ** 2)
    else:
        print("Введите положительное число")
else:
    print("Введите число")
```





Задачи

Задача: определить оценки студента на основе введенных баллов

Пользователь вводит оценку

Если оценка ≥ 90

Вывод: "Ваша оценка А (отл)"

Иначе

Если оценка ≥ 80

Вывод: "Ваша оценка В (хор)"

Иначе

Если оценка ≥ 70

Вывод: "Ваша оценка С (средне)"

Иначе

Если оценка ≥ 80

Вывод: "Ваша оценка D (плохо)"

Иначе

Вывод: "Вы не сдали"

Вложенные условия

Задача: определить оценки студента на основе введенных баллов

Пользователь вводит оценку

Если оценка ≥ 80

 Вывод: "отлично"

Иначе если оценка ≥ 60

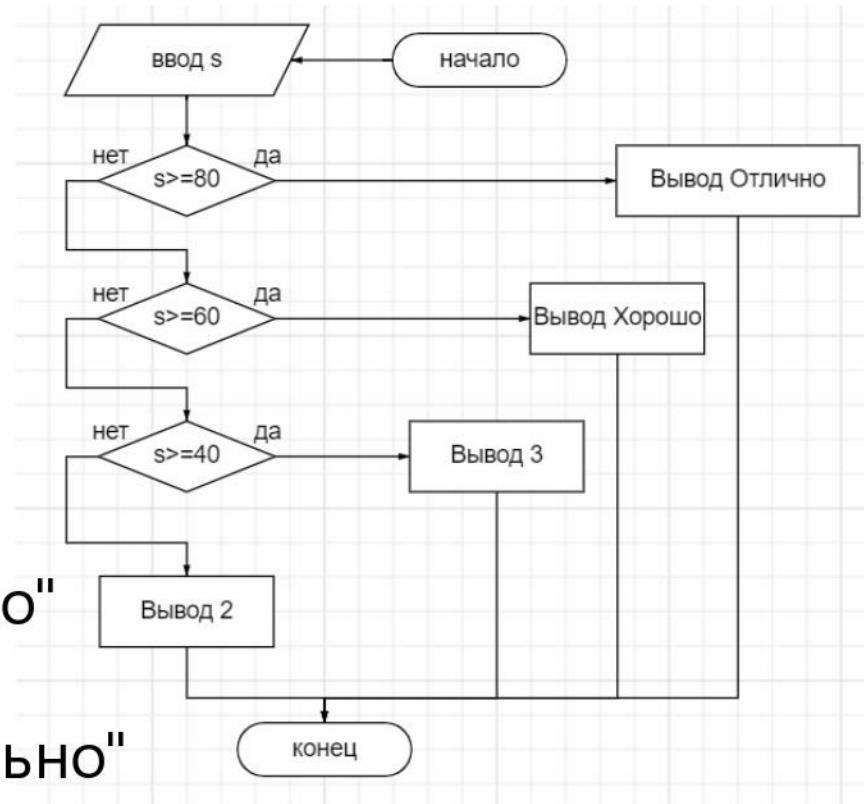
 Вывод: "хорошо "

Иначе если оценка ≥ 40

 Вывод: "удовлетворительно"

Иначе

 Вывод: "неудовлетворительно"





Вложенные условия в Python

```
score = int(input())
if score >= 80:
    print("отлично")
else:
    if score >= 60:
        print("хорошо")
    else:
        if score >= 30:
            print("удовлетворительно")
        else:
            print("неудовлетворительно")
```



Каскадные условия в Python

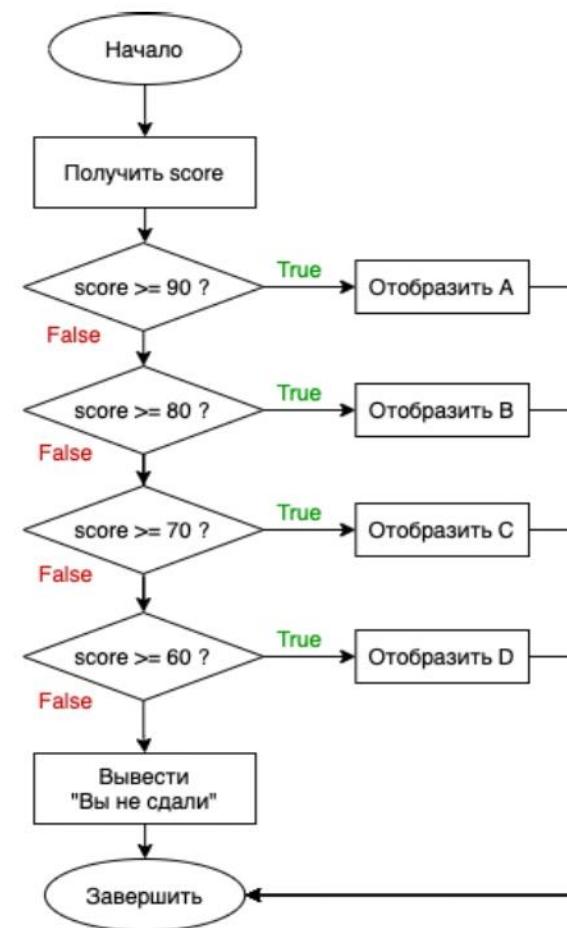
```
score = int(input())
if score >= 80:
    print("отлично")
elif score >= 60:
    print("хорошо")
elif score >= 30:
    print("удовлетворительно")
else:
    print("неудовлетворительно")
```

Оператор if-elif-else

Оператор if-elif-else — это альтернативное представление оператора if-else, которое позволяет проверять несколько условий, вместо того чтобы писать вложенные if-else.

```
score = int(input("Введите вашу оценку: "))

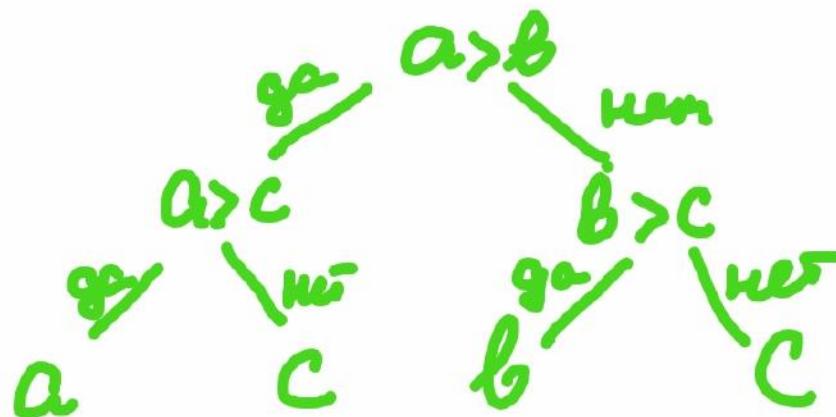
if score >= 90:
    print("Ваша оценка А (отл)")
elif score >= 80:
    print("Ваша оценка В (хор)")
elif score >= 70:
    print("Ваша оценка С")
elif score >= 60:
    print("Ваша оценка D")
else:
    print("Вы не сдали экзамен")
```





Каскадные условия в Python

```
a = int(input())
b = int(input())
c = int(input())
if a > b:
    if a > c:
        print(a)
    else:
        print(c)
elif b > c:
    print(b)
else:
    print(c)
```



```
a = int(input())
b = int(input())
c = int(input())
if a <= b <= c:
    print(c)
elif b <= a <= c:
    print(c)
elif a <= c <= b:
    print(b)
elif c <= a <= b:
    print(b)
else:
    print(a)
```



Задачи

Задача: Расчет массы, плотности или объема

Пользователь вводит один символ: m - расчет массы, d - плотности или v - объема

Если 'm'

Запросить плотность и объем

Вычислить по формуле $m = dv$

Сохранить результат в переменной result

Иначе если 'd'

Запросить массу и объем

Вычислить по формуле $d = m/v$

Сохранить результат в переменной result

Иначе

Запросить массу и плотность

Вычислить по формуле $v = m/d$

Сохранить результат в переменной result

Отформатировать вывод оставив два знака после запятой

Вывести отформатированное значение переменной result



Логические операторы

Логические операторы принимают на входе логические значения и возвращают True или False. Позволяют объединить несколько "простых условий" в более "сложное".

A	B	A and B	A or B	not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True



Задачи

Задача: Определить трудоспособный возраст

1. Пользователь вводит возраст
2. Если возраст меньше 18 или больше 60

Вывести: Вам нельзя работать

Иначе

Вывести: Вы можете работать

Задача: В какой четверти находится точка?

Пользователь вводит координаты точки x, y

Если $x > 0$ и $y > 0$:

```
print("Точка в I четверти")
```

Иначе если $x < 0$ и $y > 0$:

```
print("Точка во II четверти")
```

Иначе если $x < 0$ и $y < 0$:

```
print("Точка во III четверти")
```

Иначе если $x > 0$ и $y < 0$:

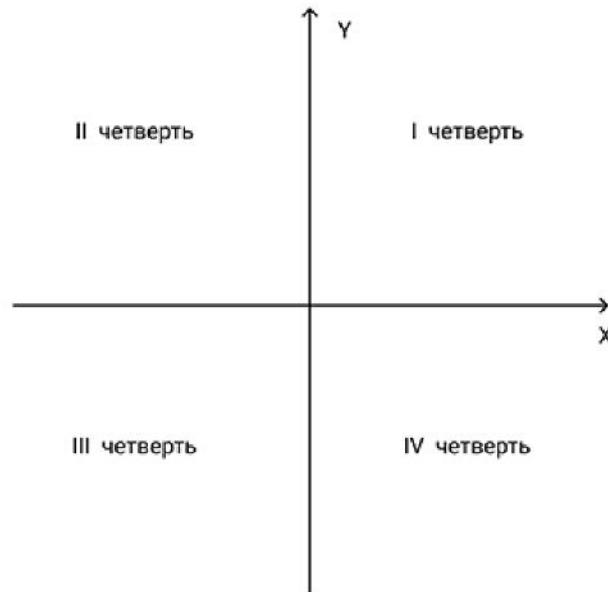
```
print("Точка во IV четверти")
```

Иначе если $x == 0$:

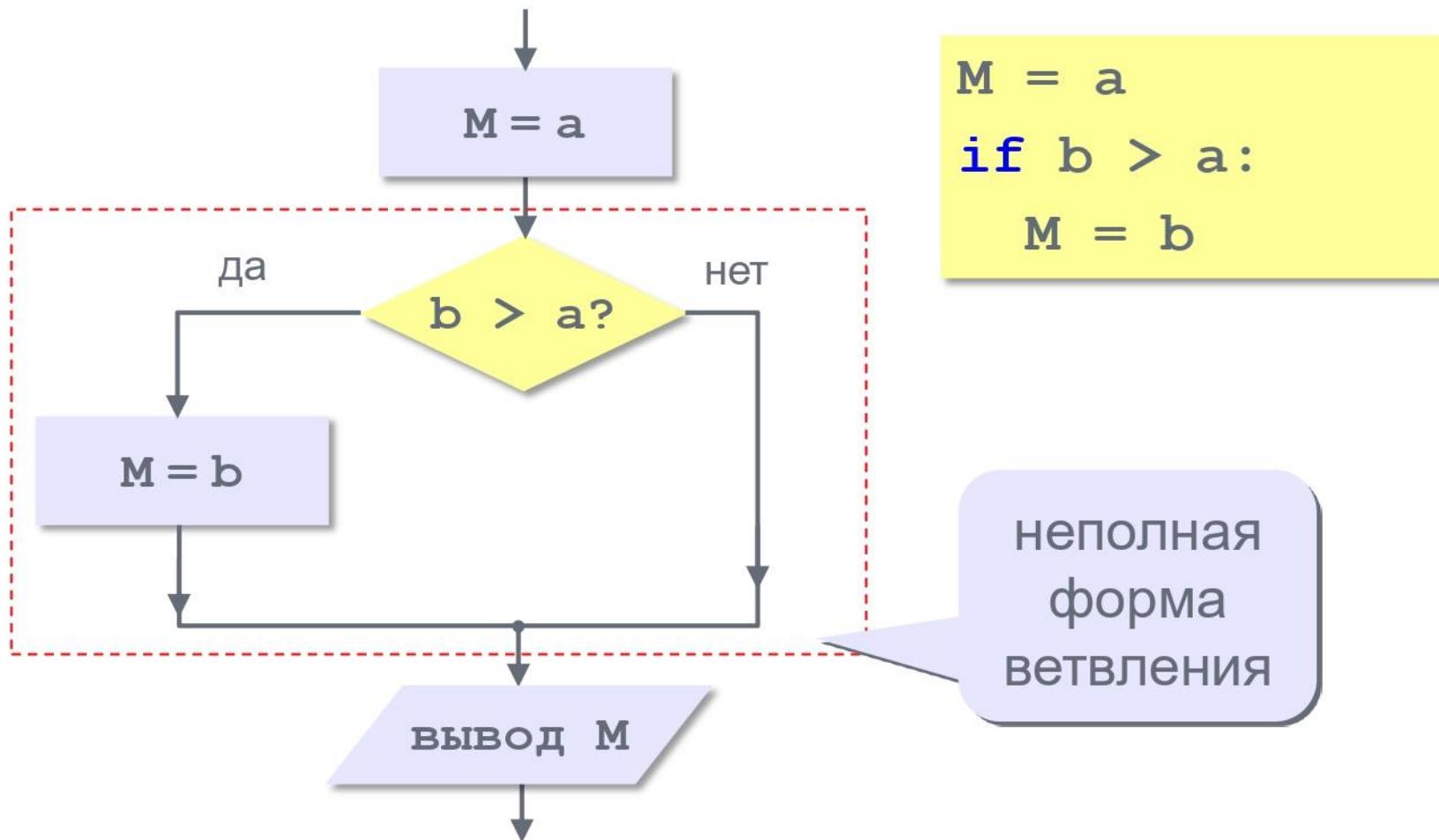
```
print("Точка на оси X")
```

Иначе если $y == 0$:

```
print("Точка на оси Y")
```



Неполная форма



Решение в стиле Python:

`M = max(a, b)`

`M=a if a>b else b`







Введение в строки

Тип данных - множество значений и операций над этими значениями

Строка - объект имеющий тип данных str. Значением строки является последовательность одного или более символов, помещенных в кавычки. Строки являются неизменяемым типом данных.

```
s1 = "Строка в двойных кавычках"  
s2 = 'Строка в одинарных кавычках'  
multi_s = """
```

Это многострочная
строка

```
"""
```



Доступ к символам

Индекс - число, представляющее позицию элемента в строке.

Доступ к каждому символу в строке производится при помощи индекса.

```
s = "Hello World!"
```

0	1	2	3	4	5	6	7	8	9	10	11
H	e	l	l	o		W	o	r	l	d	!

```
print(s[0]) # -> "H"  
print(s[1]) # -> "e"  
print(s[3]) # -> "l"  
print(s[6]) # -> "W"
```



Строковые операторы

Оператор	Операция	Пример	Результат
+	Конкатенация	"Hello" + "World"	"HelloWorld"
*	Умножение	"Hello" * 3	"HelloHelloHello"
in	Проверка на вхождение	"cat" in "my cat"	True



Управляющие символы

Кодировка - это набор числовых значений, которые ставятся в соответствие группе алфавитно-цифровых символов, знаков пунктуации и специальных символов. В Python 3 по умолчанию используется кодировка utf-8

Управляющие символы - символы в кодировке, которым не приписано графическое представление, но которые используются для управления устройствами, организации передачи данных и других целей.

Символ	Результат
\n	Перевод каретки на новую строку
\b	Возврат каретки на один символ назад
\t	Горизонтальная табуляция
\f	Перевод каретки на новую страницу
\r	Возврат каретки на начало строки
\v	Вертикальная табуляция
\u, \U	16-битовый и 32-битовый символ Unicode



Экранирование управляемых символов

Экранирование - замена в тексте управляемых символов на соответствующие текстовые подстановки.

Пример	Результат
'I\'m super'	I'm super
"Hello \"Mi\"!"	Hello "Mi"!
"C:\\user"	C:\user
"Hello\\nWorld"	Hello\nWorld



Задачи

1. Напишите программу Python, которая отображала бы ваши данные, такие как имя, возраст, адрес, в трех разных строках. Например, чтобы она вывела:

Name: Alex

Age: 25

Address: Moscow, Tverskaya 1



Понятие объект, метод и свойство

Python — объектно-ориентированный язык программирования.

Почти все в Python — это объект с его атрибутами и методами.

Объект - это способ организации данных и кода в процессе работы программы, а также способ разделения сложных задач на более простые, что облегчает их решение. Более подробно понятие объекта будет рассмотрено в отдельном модуле.

Атрибут - предназначен для хранения данных внутри объекта. Проще говоря - это переменная, которая существует только вместе с объектом.

Метод - это функция, которая принадлежит объекту определенного типа данных. Вызов метода начинается с имени объекта, потом ставится точка и указывается имя метода с последующим вызовом при помощи круглых скобок.

Метод	Результат
"Ivan petrov".title()	"Ivan Petrov"
"Hello".upper()	"HELLO"
"PC".lower()	"pc"
"hello world".capitalize()	"Hello world"



Основные строковые методы

```
s = "hello world!"  
city = " moscow "
```

Функции и методы	Описание	Результат
s.upper()	В верхний регистр (сделать прописными)	"HELLO WORLD!"
s.lower()	В нижний регистр (сделать строчными)	"hello world!"
s.capitalize()	Сделать первую букву прописной	"Hello world!"
city.strip()	Удаление пробельных символов в начале и конце строки	"moscow"
s.title()	Сделать каждое слово с заглавной буквы	"Hello World!"
len(s)	Найти длину строки	12
s.isdigit(s)	Является ли строка числом	False
s..isalpha(s)	Не является ли числом	True



Задачи

1. Напишите программу на Python для вычисления суммы трех заданных чисел, если значения равны, верните трехкратную их сумму, то сумму увеличенную в 3-и раза
2. Напишите программу Python для суммы трех заданных целых чисел x, y, z. Однако, если два значения равны, сумма будет равна нулю, например, если x == y или x == z, то сумма равна нулю.
3. Напишите программу на Python для преобразования расстояния в метрах в следующие величины: в дюймы, ярды, мили и футы. На входе программы предложите пользователю выбрать величину, в которую будет выполнено преобразование.

1м = 39,3701 дюйма = 1,093614 ярда = 0,0006214 мили = 3,281 фута

4. Напишите программу на Python, чтобы проверить, является ли число положительным, отрицательным или нулевым.
5. Напишите программу Python для нахождения суммы двух заданных целых чисел. Однако, если сумма попадает в диапазон от 15 до 20, она должна вернуть 20.
6. Напишите программу, которая проверяет, является ли введеное занчение строкой с символами алфавита или числом. Если строка содержит число, определите какой тип данных имеет число и отобразите пользователю.



Последовательность

Последовательность (sequence) - это упорядоченный набор элементов доступ к которым можно получить при помощи индекса. Индексация последовательности в Python начинается с нуля.

Последовательности могут быть как изменяемыми, так и неизменяемыми. Размер и содержимое однажды созданной неизменяемой последовательности нельзя изменить, вместо этого можно только создать новую на основе другой и добавив в нее новые данные.

Последовательность может быть однородной или неоднородной. В однородной последовательности все элементы имеют один и тот же тип. Например, строки - это однородные последовательности, в которых каждый элемент относится к одному типу.

Примеры последовательностей в стандартной библиотеке:

Последовательность	
Список (list)	Изменяемая, неоднородная
Кортеж (tuple)	Неизменяемая, неоднородная
Диапазон (range)	Неизменяемая, однородная
Строка (str, unicode)	Неизменяемая, однородная
Массив (array.array)	Изменяемая, однородная



Список

Список (list) - объект имеющий тип данных list и хранящий другие объекты любого типа в определенном порядке. Список является изменяемым типом данных.

```
names = ["ivan", "petr", "bob", "john"]
```

0	1	2	3
"ivan"	"petr"	"bob"	"john"

```
print( names[0] ) # "ivan"
print( names[1] ) # "petr"
print( names[2] ) # "bob"
print( names[3] ) # "john"
```



Понятие объект, метод и свойство

Python — объектно-ориентированный язык программирования.

Почти все в Python — это объект с его атрибутами и методами.

Объект - это способ организации данных и кода в процессе работы программы, а также способ разделения сложных задач на более простые, что облегчает их решение. Более подробно понятие объекта будет рассмотрено в отдельном модуле.

Атрибут - предназначен для хранения данных внутри объекта. Проще говоря - это переменная, которая существует только вместе с объектом.

Метод - это функция, которая принадлежит объекту определенного типа данных. Вызов метода начинается с имени объекта, потом ставится точка и указывается имя метода с последующим вызовом при помощи круглых скобок.



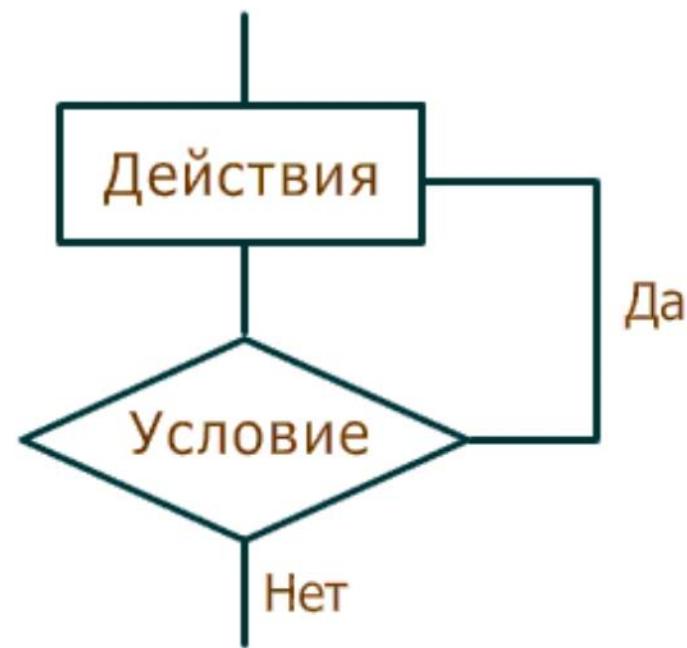
Циклы с предусловием и постусловием

- многократное исполнение действия до тех пор, пока выполняется заданное условие

Цикл "Пока"
(цикл с предусловием)



Цикл с постусловием



Арифметический цикл (с параметром)

- многократное исполнение действия для всех значений переменной-параметра от начального до конечного с заданным шагом

Арифметический цикл





Цикл, итерация, итератор

Цикл: фрагмент кода, непрерывно выполняющий инструкции (блок кода), пока удовлетворено определенное в коде условие.

Итерация - в широком смысле — организация обработки данных, при которой действия повторяются многократно, не приводя при этом к вызовам самих себя (в отличие от рекурсии). В узком смысле — один шаг итерационного, циклического процесса, например, повторение одного шага цикла.

Итератор (iterator) - это объект, который возвращает свои элементы по одному за раз.

Итерируемый объект (iterable) - это объект, который способен последовательно возвращать элементы по одному. Кроме того, это объект, из которого можно получить специальный объект итератор. Пример итерируемых объектов: список, кортеж, строка и другие.

Итерирование (перебор): использование цикла для получения доступа к каждому элементу итерируемого объекта.

Цикл for

Цикл for - это цикл, перебирающий итерируемый объект — например, строку, список, кортеж или словарь.

Цикл for используется, чтобы создать блок кода, который будет выполняться один раз для каждого элемента в итерируемом объекте. Внутри блока кода можно получить доступ к этим элементам и осуществить операции с этими элементами.





Синтаксис цикла for

Синтаксис — это набор правил, принципов и процессов, которые определяют структуру предложений на определенном языке, в частности, порядок слов. У русского языка есть синтаксис, и у Python есть синтаксис.

for имя_переменной in имя_итерируемого_объекта:
 блок кода (инструкции)

имя_переменной — выбранное вами имя переменной, которая назначается значению каждого элемента в итерируемом объекте, а блок кода — код, который выполняется при каждом прохождении цикла



Перебор элементов строки

При помощи цикла `for` можно перебрать по очереди все символы строки.

```
name = "Иван"  
for character in name:  
    print(character)
```

Результат работы цикла `for`:

И
в
а
н



Перебор элементов списка

При помощи цикла for можно перебрать по очереди все элементы списка.

```
names = ["ivan", "petr", "bob", "john"]
```

```
for name in names:  
    print(name.capitalize())
```

Результат работы цикла for:

Ivan
Petr
Bob
John

```
name = "Иван"  
for character in name:  
    print(character)
```



Функция range()

При помощи функции **range()** можно создать последовательность целых чисел и цикл `for`, чтобы выполнить ее перебор.

range(start, stop, step)

`start` - число с которого начинается последовательность

`stop` - число на котором заканчивается (не включая это число)

`step` - шаг

```
for i in range(5):    for i in range(1, 5):    for i in range(5, 1, -1):  
    ... print(i)        ... print(i)        ... print(i)
```

Результат:

```
0  
1  
2  
3  
4
```

Результат:

```
1  
2  
3  
4
```

Результат:

```
5  
4  
3  
2
```



Практика Черепаха

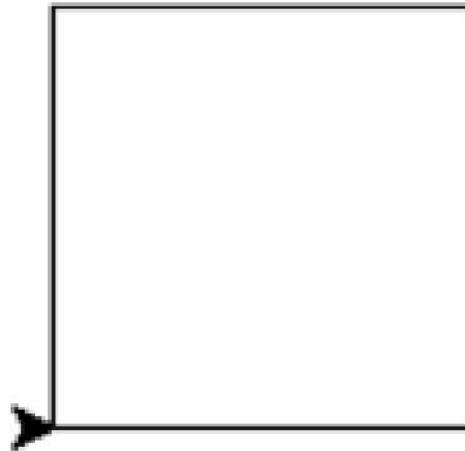
Стандартная библиотека Python содержит модуль `turtle`, предназначенный для обучения программированию. Этот модуль содержит набор функций, позволяющих управлять черепахой. Черепаха умеет выполнять небольшой набор команд, а именно:

Команда	Значение
<code>forward(X)</code>	Пройти вперёд X пикселей
<code>backward(X)</code>	Пройти назад X пикселей
<code>left(X)</code>	Повернуться налево на X градусов
<code>right(X)</code>	Повернуться направо на X градусов
<code>penup()</code>	Не оставлять след при движении
<code>pendown()</code>	Оставлять след при движении
<code>shape(X)</code>	Изменить значок черепахи ("arrow", "turtle", "circle", "square", "triangle", "classic")
<code>stamp()</code>	Нарисовать копию черепахи в текущем месте
<code>color()</code>	Установить цвет
<code>begin_fill()</code>	Необходимо вызвать перед рисованием фигуры, которую надо закрасить
<code>end_fill()</code>	Вызвать после окончания рисования фигуры
<code>width()</code>	Установить толщину линии
<code>goto(x, y)</code>	Переместить черепашку в точку (x, y)



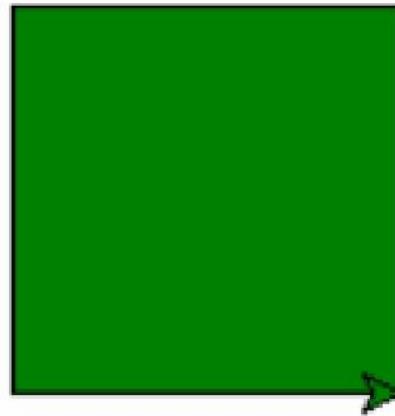
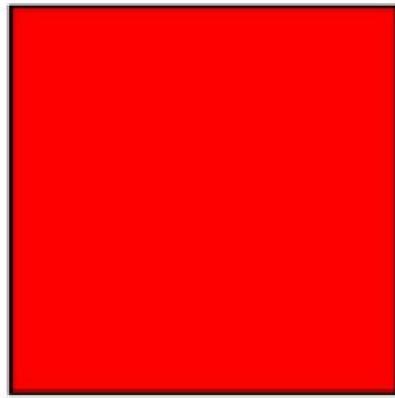
Задание 1. Квадрат

```
import turtle  
  
for i in range(0, 4):  
    ....turtle.forward(100)  
    ....turtle.left(90)  
  
turtle.exitonclick()
```

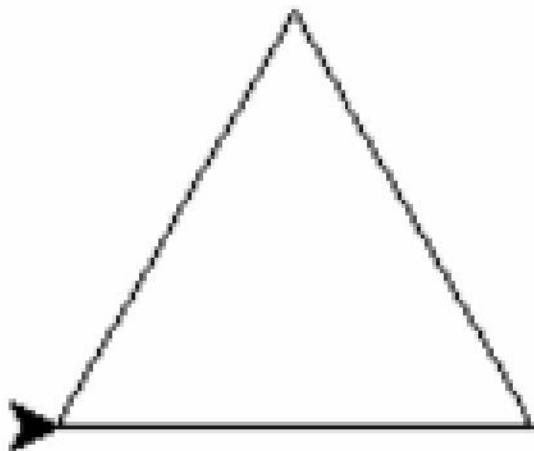




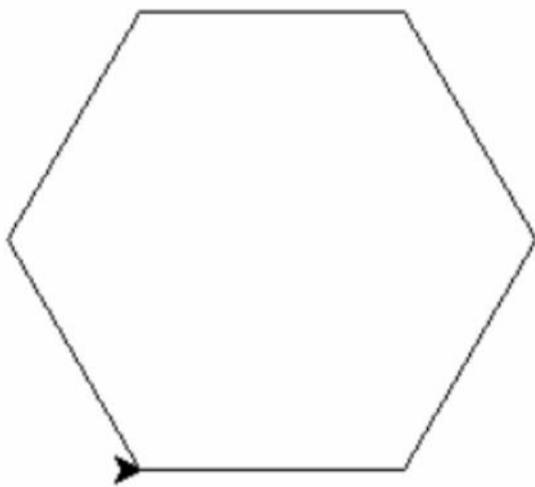
Задание 2. Квадраты



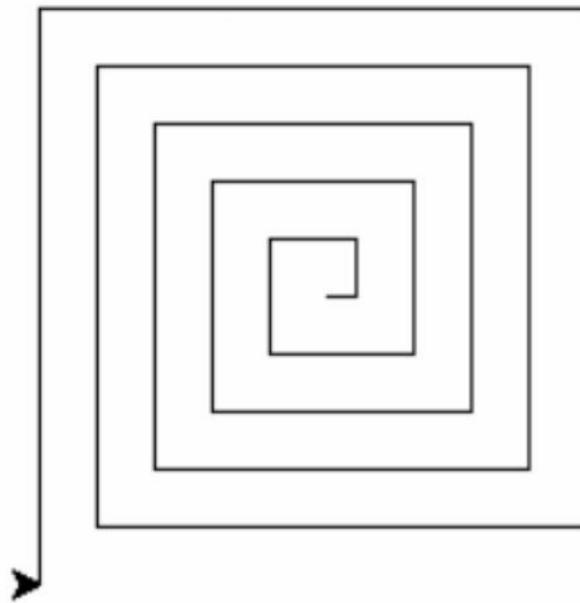
Задание 3. Треугольник



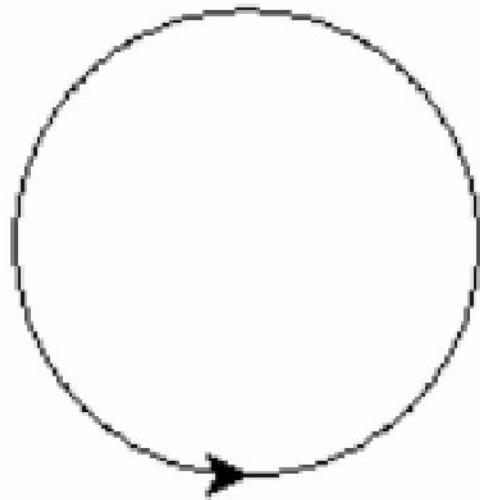
Задание 4. Шестиугольник



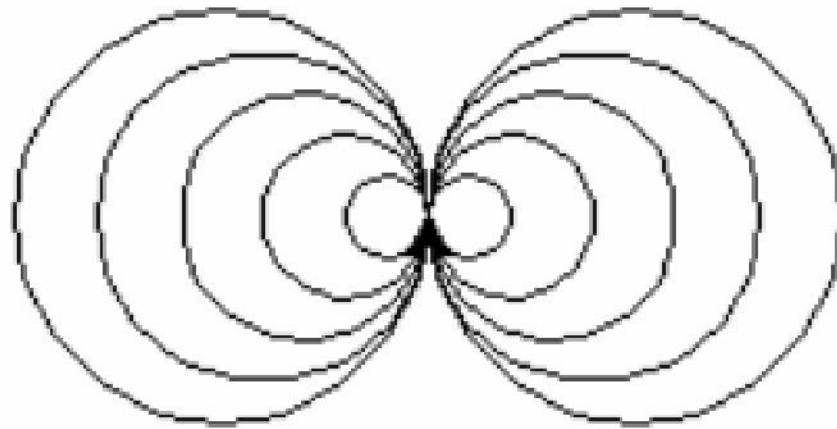
Задание 5. Квадратная спираль



Задание 5. Круг



Задание 6. Бабочка





Вложенный цикл

Вы можете поместить один цикл `for` внутри другого. Нет никаких ограничений по количеству циклов, которые можно помещать внутрь других циклов, но большой глубины вложенности лучше избегать. Когда цикл находится внутри другого цикла, второй цикл является вложенным в первый. В этом случае цикл, содержащий внутри другой цикл, называется **внешним**, а вложенный цикл — **внутренним**. Когда у вас есть вложенный цикл, внутренний цикл выполняет перебор своего итерируемого объекта один раз за итерацию внешнего цикла.

```
for i in range(1, 3):
    print('Внешний цикл. i == ', i)
    for letter in ["а", "б", "в"]:
        print(letter)
```

Внешний цикл. i == 1

а

б

в

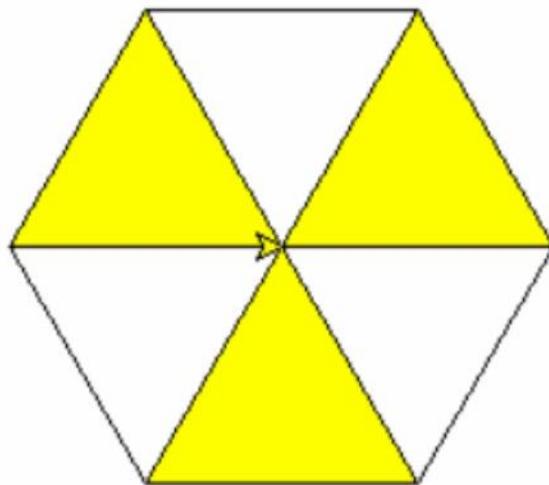
Внешний цикл. i == 2

а

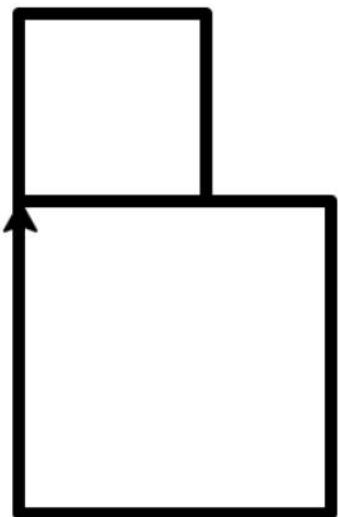
б

в

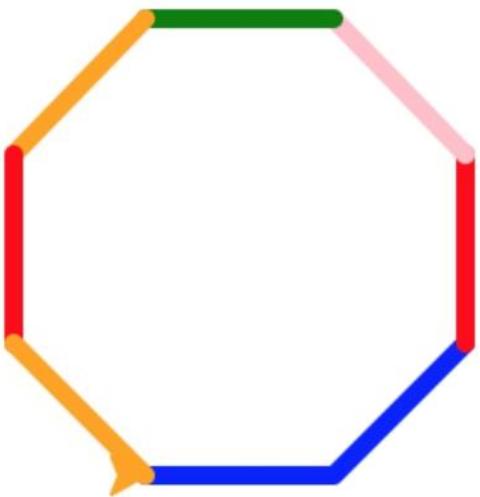
Задание 7. Бабочка



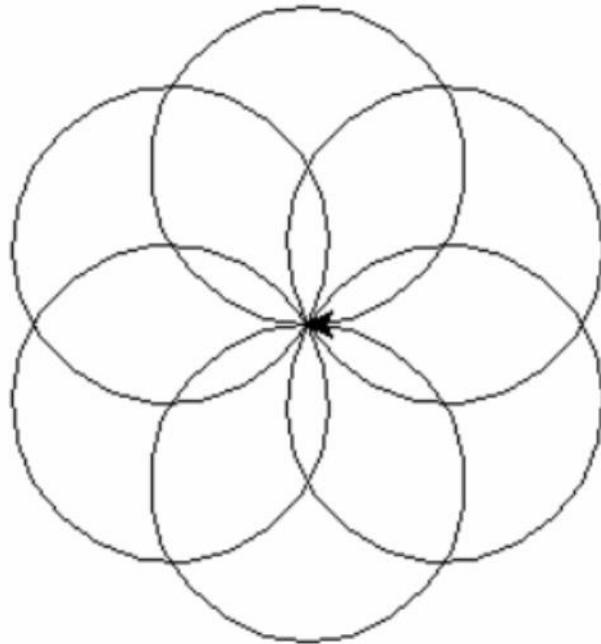
ДЗ 1. Буква В



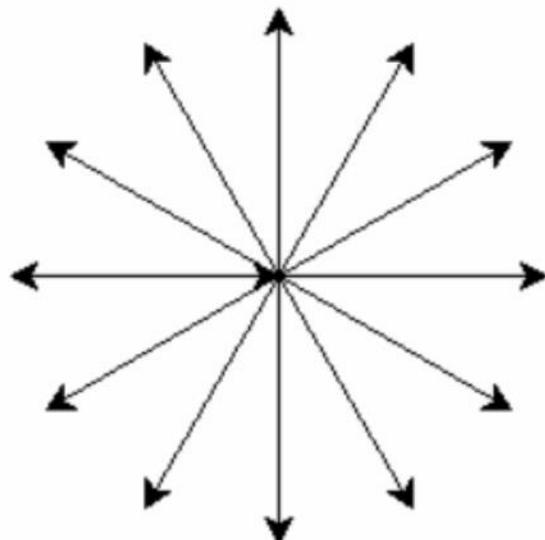
ДЗ 2. Восьмиугольник



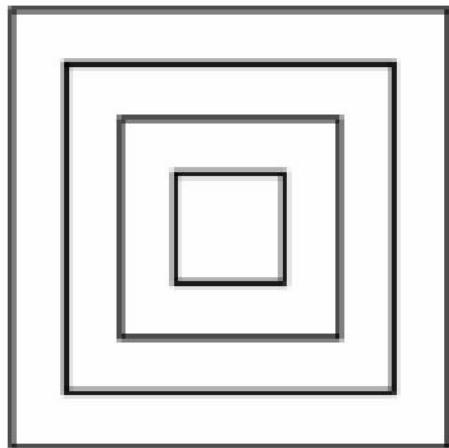
ДЗ 3. Цветок



ДЗ 4. Паук

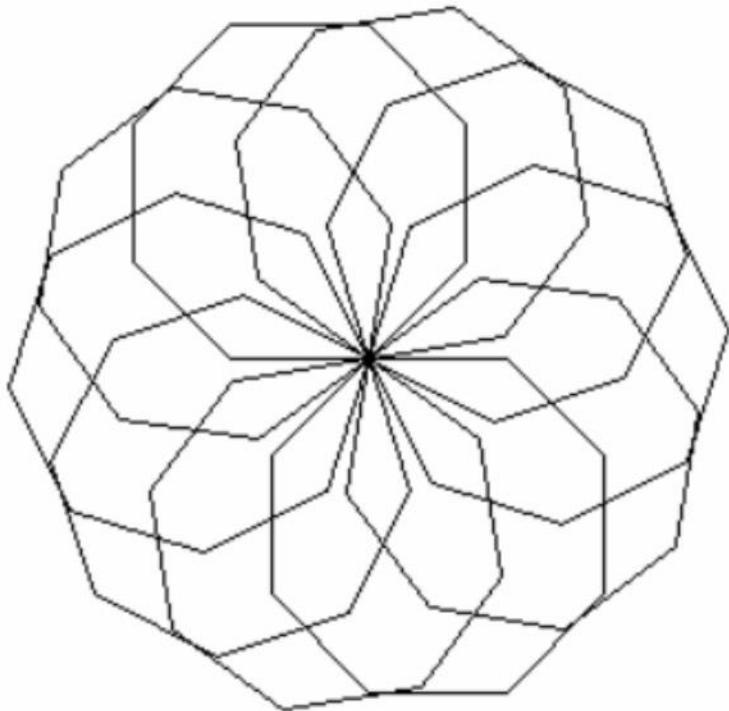


ДЗ 5. Квадрат в квадрате



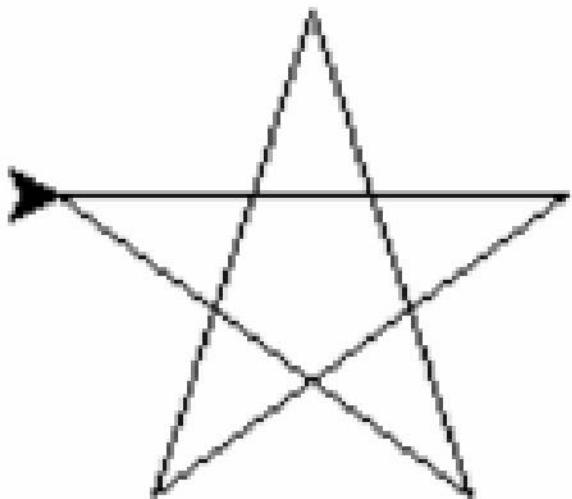


ДЗ 6. Узор





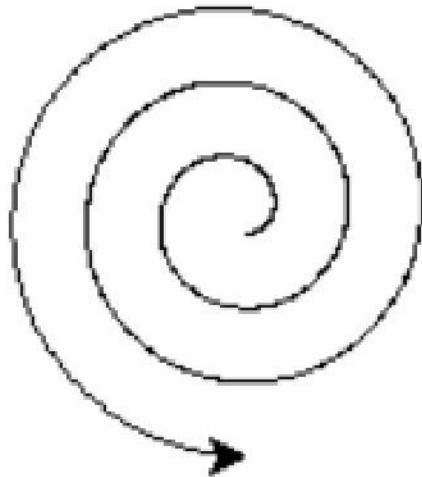
*ДЗ 7. Звезда



* - для тех, кто справится с другими заданиями



*ДЗ 8. Спираль



* - для тех, кто справится с другими заданиями



Входит в ГК Аплана



АКАДЕМИЯ АЙТИ

Основана в 1995 г.

E-learning
и очное
обучение



Ежегодные награды
Microsoft,
Huawei, Cisco и
другие

Направления обучения:

Информационные технологии

Информационная безопасность

ИТ-менеджмент и управление проектами

Разработка и тестирование ПО

Гос. и муниципальное управление

Филиалы:

Санкт-Петербург, Казань, Уфа, Челябинск,
Хабаровск, Красноярск, Тюмень, Нижний
Новгород, Краснодар, Волгоград, Ростов-на-Дону

Головной офис
в Москве

Ресурсы более 400
высококлассных
экспертов и
преподавателей

Разработка
программного
обеспечения и
информационных
систем

Программы по
импортозамещению

Сеть региональных учебных центров
по всей России

Крупные заказчики



100+
сотрудников



АКАДЕМИЯ АЙТИ

Спасибо за внимание!

Центральный офис:

Москва, Варшавское шоссе 47, корп. 4, 7 этаж

Тел: +7 (495) 150-96-00

academy@it.ru

academyit.ru