МБОУ СОШ №9 г. Нерчинск

Знакомство с языком программирования

линейные и циклические структуры, модуль Turtle

Пособие для 6-7 классов подготовил для занятий на кружках и факультативах
учитель информатики высшей категории Курилов И.А.

Нерчинск, 2022 год

1.Введение

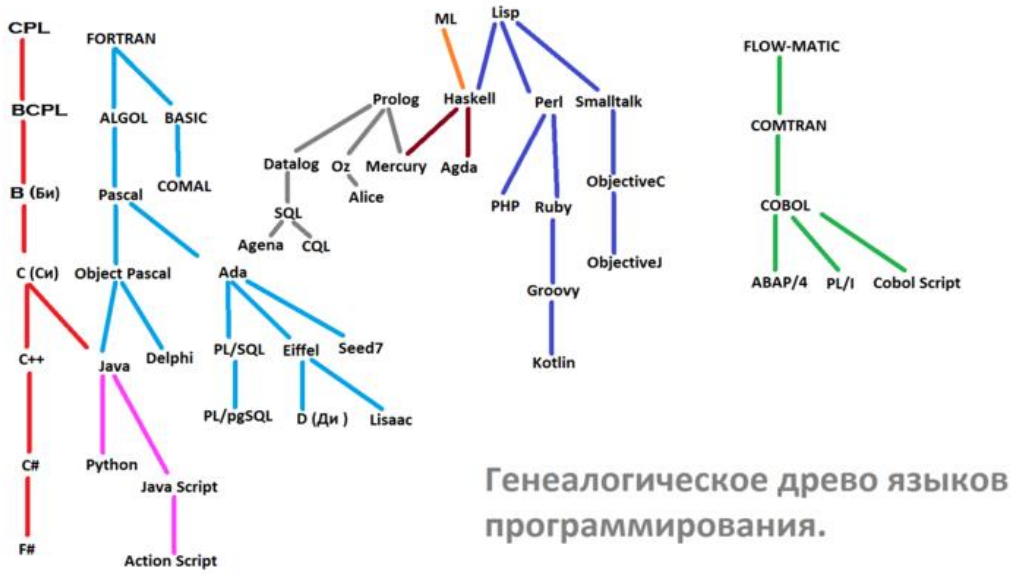
Что такое Python и почему так назвали?

Python (в честь шоу Monty Python) – язык программирования, разработанный Гвидо Ван Россумом в 1991 году!

Отличительная черта Python - **использование отступов для выделения блоков кода и управляющих структур, отсутствие описание переменных, простота работы с длинными числами.**

Язык крайне простой и скромный на выразительные средства сравнительно с Ruby или Perl . Использование описаний переменных, как в Pascal или C++ не нужно!

Место в структуре языков

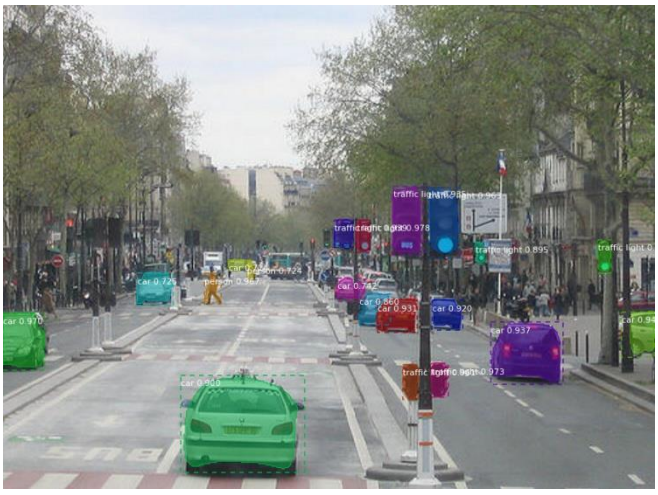


Почему надо изучать Python?

- Высокая популярность языка и использование в большом количестве проектов;
- Сравнительно простой, но в то же время строгий синтаксис;
- Множество доступных сред разработки, сервисов и фреймворков;
- Средства для работы с электронной почтой, протоколами Интернета, базами данных и другие инструменты в стандартной библиотеке языка;
- Востребованность Python-разработчиков на рынке труда.

*Примеры **проектов на Python***

1.YouTube
2.Google Search
3. DropBox (Облачные хранилища).
4.Instagram
5.Yahoo Maps
6.Mask R-CNN
7.Face Recognition
8.МАШИННОЕ ОБУЧЕНИЕ …



Рейтинг PYPL (по поиску Google)

Во всем мире, сентябрь 2022 по сравнению с прошлым годом:

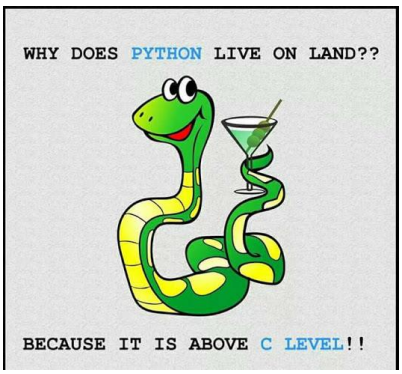
Рейтинг	Изменение	Язык	Поделиться	Тенденция
1		Python	28.29 %	-1.8 %
2		Java	17.31 %	-0.7 %
3		JavaScript	9.44 %	-0.1 %
4		C#	7.04 %	-0.1 %
5		C/C++	6.27 %	-0.4 %
6		PHP	5.34 %	-1.0 %
7		R	4.18 %	+0.3 %
8	↑↑↑	TypeScript	3.05 %	+1.5 %
9	↑↑↑	Вперед	2.16 %	+0.6 %
10		Swift	2.11 %	+0.5 %
11	↓↓↓	Objective-C	1.93 %	-0.0 %
12	↓↓↓	Kotlin	1.88 %	+0.0 %
13		Matlab	1.55 %	+0.1 %
14	↑↑	Ржавчина	1.5 %	+0.7 %
15		Ruby	1.13 %	+0.1 %
16	↓↓	VBA	0.94 %	-0.3 %
17	↑	Dart	0.83 %	+0.2 %
18	↑	Ada	0.79 %	+0.2 %

2.Основные сведения

Ввод данных

Ввод данных осуществляется без описания типа переменных:
a = input() – в этом случае строковая переменная (по умолчанию) или с описанием типа строкового типа переменных **a =str(input()).** В скобках можем писать пояснения.

Если предполагается, что вводится число (а не набор символов), придется преобразовать входную строку к числовому типу:
b = int(input()) # для целых
c = float(input()) # для дробных чисел.



Если в строке вводится не одно число, а сразу несколько (например, четыре), придется воспользоваться функцией **map**:
a, b, c, d = map (int, input().split())
map - применить
int - эту операцию (преобразование в число) к каждой части
input().split() - разделить строку на части по пробелам.

Обработка данных

Рассмотрим основные функции обработки чисел.
x + y # Складывает число x и число y
x - y # Вычитает число y из числа x
x * y # Умножает x на y
x / y # Делит x на y - результатом всегда является значение типа float
x // y # Возвращает целочисленное частное от деления x на y
x % y # Возвращает модуль (остаток) от деления x на y
x ** y # Возводит x в степень y
-x # Изменяет знак числа x
abs(x) # Возвращает абсолютное значение x (модуль числа)
round(x, n) # Возвращает значение типа int, соответствующее значению x типа float,

округленному до ближайшего целого числа (или значение типа float, округленное до n-го знака после запятой, если задан аргумент n)

Вывод данных и первая программа

Для вывода данных используется команда **print().**

Решим следующую задачу: с клавиатуры вводятся два числа, каждое в отдельной строке, вывести их среднее арифметическое.

Решение:

```
a = int(input())
b = int(input())
c = a + b
c = c / 2
print (c)
```

То же самое, но без промежуточных переменных:

```
print ((int(input()) + int(input())) / 2)
```

Математическая библиотека*

Питон, как и все современные языки, имеет богатую *библиотеку* — набор готовых "решений" некоторых задач, написанных программистами — авторами языка Питон. Вы можете использовать эти решения в своих программах.

Библиотека языка Питон разбита на части, называемые модулями. В этом листке мы рассмотрим модуль math, содержащий множество математических функций. Для их использования необходимо в начале своей программы написать инструкцию

```
import math
```

Далеко не полный список математических функций выглядит так:

floor(x)- округляет число x вниз. Результатом является наибольшее целое число, не превосходящее x, представленное в виде действительного числа.

ceil(x)- округляет число x вверх. Результатом является наименьшее целое число, не меньше x, представленное в виде действительного числа.

fabs(x)- вычисляет модуль (абсолютную величину) действительного числа x

Пример:

```
x = int(input())
print "Модуль числа=", math.fabs(x)
```

**Помимо этих функций в модуле math содержатся функции вычисления логарифмов, степеней, гиперболические функции и многое другое.*

Также в модуле math определены две константы (постоянные), одна из них - **math.pi, равное числу π!**

Практическое занятие №1 «Линейные алгоритмы. Вычисляем по формулам»

Вы изучали, что такое линейный алгоритм, как выглядит блок-схема линейного алгоритма, а выше была приведена программа линейного алгоритма.

Давайте с вами посмотрим на несколько простых алгоритмов, а потом попробуем составить сами.

Пример №1.

Составить программу, зная стороны прямоугольника a и b, найти площадь прямоугольника s.

```
a=float(input("Введите 1-ую сторону прямоугольника:"))
b=float(input("Введите 2-ую сторону прямоугольника:"))
s= a * b
print (s)
```

Пример №2.
Составить программу нахождения площади круга s, зная радиус этого круга r.

```
import  math #Подключение математического модуля
r=float(input("Введите радиус круга:"))
s= math.pi*r**2
print (s)
```

Пример №3.
Кузнечик прыгал влево и вправо на координатной прямой. Сначала кузнечик находился в точке А на координатной прямой, он прыгнул на x1 влево, а затем 2 раза на x2 шагов вправо. В какой точке В окажется кузнечик после прыжков.

```
a=int(input("Введите начальное положение кузнечика A:"))
x1=int(input("Введите размер прыжков влево x1:"))
x2=int(input("Введите размер прыжков вправо x2:"))
b= a-x1+2*x2
print (b)
```

Пример №4.
Составить программу, зная стороны комнаты (a – ширина, b – длина, c – высота), найти, сколько рулонов обоев нужно для поклейки данной комнаты. Будем считать, что клеить нужно 3 стены (2 маленьких и 1 большую сторону, на 4 стене находятся окна).
Найдем площадь, которую нужно поклеить: s=2*(a*c)+b*c
Все знаем, что обычные обои 0,5м*10м=5м^2, поэтому количество обоев определяется: k=s/5 и усовершенствуем последнюю формулу, округлив до целого числа обоев: k=round(s/5+0.5).

```
a=float(input("Введите a-ширину:"))
b=float(input("Введите b-длину:"))
b=float(input("Введите c-длину:"))
s=2*(a*c)+b*c
k=round(s/5+0.5)
print (k)
```

Задачи:
1.Зная стороны прямоугольника a и b, найти периметр прямоугольника.
2.Зная радиус окружности, найти длину окружности.
3.Даны две точки на координатной прямой с координатами A и B. Найти расстояние между 2-мя точками s.
4.Помогите Звездочке найти внутреннюю площадь кубического бака стороной n.
5.Дано начальное положение автомобиля на координатной прямой – A (одна клетка – 1 км.). Затем автомобиль начал движение. Сначала он двигался со скоростью v1 км/ч вправо n1 часа, затем v2 км/ч влево n2 часа. Найти конечное положение автомобиля.

Практическое занятие №2 «Линейные алгоритмы. Находим целую часть и остаток»

В первой работе мы считали по геометрическим формулам, а также выводили формулы. В этой работе мы решим задачи с нахождением остатка и целой части при делении.

Пример №5.
Составить программу, найти последнюю цифру числа x.

```
x=int(input('Введите число:'))
pos_cif= x % 10
print (pos_cif)
```

Пример №6.1. Найдем сумму первой и второй цифры заданного 2-хзначного числа.

```
x=int(input('Введите число:'))
a= x//10
b=x%10
s=a+b
print (s)
```

Пример №7.1. Найдем центральную цифру 3-значного числа.

```
x=int(input('Введите 3-хзначное число:'))
x= x//10
b=x%10
print (b)
```

Задачи:
1.Даны три трехзначных числа. Найти произведение последних цифр этих чисел.
2.Дано трехзначное число. Найти первую цифру этого числа.
3.В классе n человек, помогите Незнайке найти, сколько команд из 5 человек может сформировать учитель физкультуры и сколько человек еще останется лишних учеников?
4.«Игра в цифры». Петя и Ваня решили поиграть в одну придуманную ими игру. Один из мальчиков говорит 4-хзначное число, а другой должен быстро найти сумму 1-ой и 3-ей цифры этого числа. Помогите автоматизировать процесс игры.

Практическое занятие №3 «Линейные алгоритмы. Символьная строка»

Вы уже знаете, что по умолчанию вводится символьная строка, чтобы ввести число вы пишите вначале **int** (смотрите «Основные сведения»). А что если этим воспользоваться! Для этого надо знать немного из темы «Символьные переменные»:
len(s) – функция определяет длину символьной строки
s[0], s[1], … - первый, второй, … символ в символьной строке
s[-1], … - последний, … символ в символьной строке

Например, из практического занятия №2 все примеры и задачи №1,2,4 можно решить этим способом. Рассмотрим ниже данный способ.

Пример №8. Дано слово. Вывести 1-ую и последнюю букву.

```
x=input('Введите слово:')
print (x[0],x[-1])
```

Теперь применим символьную строку к задачам с числами. Всего-то надо не забыть переводить в цифры (числа) операцией **int**.

Пример №6.2. Найдем сумму первой и второй цифры заданного 2-хзначного числа.

```
x=input('Введите число:')
print (int(x[0])+int(x[1]))
```

Пример №7.2. Найдем центральную цифру 3-значного числа.

```
x=input('Введите 3-хзначное число:')
print (int(x[1])
```

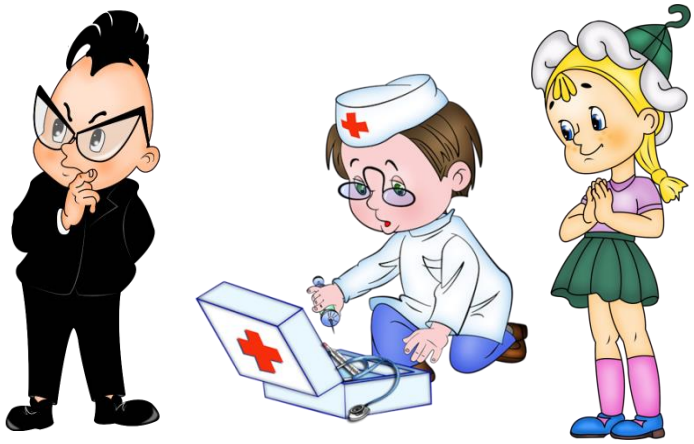
Пример №9. Найдем длину введенного числа, выведем первую и последнюю цифру этого числа, сложим эти цифры (с использованием символьных переменных).

```
x=input('Введите n-хзначное число:')
n=len(x)
print (n)
print (x[0],x[n-1],int(x[0])+int(x[n-1]))
```

Как мы видим использование символьные переменных упрощает задачу.

Задачи:

1.Введите своё имя и имена 2 друзей. Результат работы вашей программы – вывод первых букв имен.
2.Дано четырехзначное число. Найти сумму первой и последней цифры этого числа.
3.Даны три числа разной значности. Найти произведение последних цифр этих чисел.
4.Знайка, Пилюлькин и Поночка решили научить Незнайку умножению. После урока они задали ему задачу: нужно в году рождения автора данной книги из библиотеки сложить все цифры. Помогите Незнайке составить такую программу.



Практическое занятие №4 «Линейные алгоритмы. Графика на Python»

В Python есть специальный модуль под названием **turtle**, которым мы и воспользуемся, чтобы изучить основы создания изображений на экране. С помощью модуля **turtle** можно программировать векторную графику, то есть составлять рисунки из линий, точек и кривых.

1.Подключения модуля turtle и другие основные установки
В первой строке пишем:
а)import turtle
или
б)from turtle import * (чтобы каждый раз не подключать модуль **turtle)**

Примеры программ[1]:
1а) «Квадрат»

```
import turtle
t=turtle.Pen() # использовуем букву t в качестве обозначения ручки черепашки
t.forward(150)
t.left(90)
t.forward(150)
t.left(90)
t.forward(150)
t.left(90)
t.forward(150)
t.left(90)
```

1б) «Квадрат»

```
from turtle import*
shape("turtle") # использовуем внешний вид – черепашку*
forward(150)
left(90)
forward(150)
left(90)
forward(150)
left(90)
forward(150)
left(90)
```

или с полной установкой окна

```
# Черепашья графика
from turtle import *
# Настройка окна
Width, Height = 600, 400
Window = Screen()
Window.title("Черепашка")
Window.setup(width=Width, height=Height)
# Рисование
shape("turtle")
forward(150)
left(90)
…
```

([1]Ниже в таблице приведены основные команды)

Примечание:
а) Параметр «**Style**» в функции **shape()** изменяет внешний вид исполнителя. Поддерживаются следующие стили:
• **arrow**
• **turtle**
• **circle**
• **square**
• **triangle**
• **classic**
б) Существует несколько команд определяющих внешний вид исполнителя:
1. **turtle.shape(«Style»)** - изменяет внешний вид
2. **turtle.shapesize(m,n)** — управляет размерами исполнителя
3**.turtle.tilt(alpha)** — изменяет ориентацию исполнителя
4. **turtle.hideturtle()** - скрывает черепаху
5. **turtle.showturtle()** - показывает черепаху и др.
в) Чтобы программа с модулем **turtle** на Python работала корректно, в самом конце программы всегда нужно прописывать две команды:
1. t.screen.exitonclick()
2. t.screen.mainloop()
С помощью команды **t.screen.exitonclick()** программа на Python реагирует на нажатие кнопки мыши после исполнения программы. Если пользователь нажмёт на левую кнопку мыши, пока курсор находится в окне для графики модуля **turtle**, то окно закроется.
С помощью команды **t.screen.mainloop()** останавливает выполнение программы.

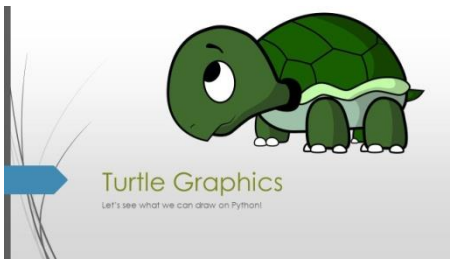
2.Список основных команд черепашки

Команды перемещения черепашки	
forward(n)/fd(n)	Проползти вперед n шагов (пикселей)
backward(n)	Проползти назад n шагов (пикселей)
left(angle)	Повернуться налево на angle градусов
right(angle)	Повернуться направо на angle градусов
circle(r)	Нарисовать окружность радиуса |r|, центр которой находится слева от черепашки, если r>0 и справа, если r<0
circle(r,angle)	Нарисовать дугу радиуса |r| и градусной мерой angle. Дуга рисуется против часовой стрелки, если r>0 и по часовой стрелке, если r<0
goto(x,y)	Переместить черепашку в точку с координатами (x,y)
Команды рисования	
down()	Опустить перо. После этой команды черепашка начнет оставлять след при любом своем передвижении
up()	Поднять перо
width(n)	Установить ширину следа черепашки в n пикселей
color(s)	Установить цвет следа черепашки в s. s должно быть текстовой строкой, заключенной в кавычки, с названием цвета (по-английски), например: "red", "yellow", "green" и т.д.
Bgcolor(s)	Устанавливает цвет фона в s. s должно быть текстовой строкой, заключенной в кавычки, с названием цвета (по-английски), например: "red", "yellow", "green" и т.д.
fill(f)	Используется для рисования закрашенных областей. Начиная рисовать закрашенную область, дайте команду turtle.fill(1), а закончив рисование области — turtle.fill(0)
stamp()	После выполнения этой команды в окне для графики в месте, на котором была черепашка, останется рисунок этой черепашки**.**
setheading(x)	где x – угол поворота в градусах относительно начального положения черепашки при запуске программы
Прочие команды	

reset()	Возврат черепашки в исходное состояние: очищается экран, сбрасываются все параметры, черепашка устанавливается в начало координат, глядя вправо
write(s)	Вывести текстовую строку s в точке нахождения черепашки
dot(r, color)	Команда ставит точку, где r – радиус точки и color – цвет точки
radians()	Установить меру измерения углов (во всех командах черепашки) в радианы
degrees()	Установить меру измерения углов (во всех командах черепашки) в градусы. Этот режим включен по умолчанию
tracer(f)	Включить режим отладки (трассировки) программы черепашки, если значение f равно 1. Если значение f равно 0, отключить режим отладки. В режиме отладки черепашка перемещается медленнее и на экране нарисована сама черепашка. По умолчанию режим отладки включен
speed()	устанавливает скорость движения черепахи

2) «Квадрат-2»
Добавим цвета!

```
from turtle import*
reset() # черепаха в начальное положение (сбрасывает цвет), работает по умолчанию
down() # опускаем перо, работает по умолчанию
shape("turtle")
bgcolor("yellow") # фон желтый
color("red") # чертит красным
forward(150)
left(90)
forward(150)
left(90)
forward(150)
left(90)
forward(150)
up()
forward(100)
mainloop() #задержать окно на экране
```



Задачи:

1.Составить программу построения лестницы со ступеньками (4-5 штук).
2. Составить программу построения движение черепашки прямоугольной змейкой (4-5 штук).
3. Составить программу построения шестиугольника.
4. Составить программу построения трех соприкасающихся окружностей (4-5 штук).
5. Составить программу построения творческого рисунка (например, домика и используем краски).

Практическое занятие №5 «Алгоритмы с ветвлением»

Что такеое ветвление?

Ветвление – это алгоритм, в котором в зависимости от условия выполняется одна из последовательности действий.

Как организовать ветвление на Python?

В Python, в отличие от других языков, сдвиги операторов относительно левой границы (отступы) *влияют* на работу программы.

Полное и неполное ветвление

Пример 1: Рассмотрим запись ветвления на примере нахождения **максимального** значения из 2-х чисел.

Вариант 1:

```
if a > b: m = a
else: m = b
```

Вариант 2:

```
m = a if a > b else b
```

Вариант 1 и 2 используют **полную** форму ветвления, а можно обойтись **неполной** формой.

Вариант 3:

```
m = a
if b > a: m = b
```

Вариант 4:

Кроме всего в Python есть функция max (и min) и можно обойтись вовсе без ветвления:

```
m = max( a, b )
```

Аналогично можно найти max (min) из 3, 4-х и т.д чисел.

Составной оператор

Часто при выполнении какого-то условия нужно выполнить сразу несколько действий.

Пример 1: **Обмен значений двух переменных (ячеек памяти):**

```
if a > b:
  dop = a
  a = b
  b = dop
```

Пример 2: **Определение максимума и минимума**

Вариант 1:

```
if a > b:
    ma=a
    mi=b
else:
    ma=b
    mi=a
```

Вариант 2:

Заметим, что в Python, в отличие от многих других языков программирования, есть множественное присваивание, которое позволяет выполнить такой обмен значительно проще (пример 1): **a, b = b, a**

Аналогично можно производить более сложные перестановки, например:

a, b, c = b, c, a

Вложенные условные операторы
Условный оператор находящийся внутри блока «иначе» (**else**) называется *вложенным условным оператором*. Использование вложенных условных операторов позволяет выбрать один из *нескольких* (а не только из двух) вариантов.
Пример 1: А возраст Александра, В возраст Владимира. Нужно определить, кто из них старше или оба одного возраста.

```
…
if A > B:
    print( "Александр старше" )
 else:
    if A == B:
      print( "Одного возраста" )
    else:
      print( "Владимир старше" )
```

Обратите внимание на отступы: слова **if**, **elif** и **else** находятся на одном уровне.

Пример 2:Скидка при 1500 рублей.
В цепочке операторов **if-elif-elif-**… срабатывает первое истинное условие.

```
cost = 1500
if cost < 1000:
  print( "Скидок нет." )
elif cost < 2000:
  print( "Скидка 2%." )
elif cost < 5000:
  print( "Скидка 5%." )
else:
  print( "Скидка 10%." )
```

Выводит «Скидка 2%.», хотя условие *cost* < 5000 тоже выполняется.

Сложные условия
В сложных условиях используется одна из логических операций: «И» (and), «ИЛИ» (or), «НЕ» (not).
Пример 1: Прием на работу

```
…
if age >= 20 and age <= 50:
  print( "подходит" )
else:
 print( "не подходит" )
```

В программе *на языке Python можно сразу проверить выполнение двойного неравенства:*

```
…
if  20 <= age <= 50:
   print("подходит")
else:
   print("не подходит")
```

Пример 2: Сложное условие с операцией «И»
Предположим, что нам надо убедиться, что значение целой переменной *a* – двухзначное число, которое делится на 5. Для этого нужно, чтобы одновременно выполнились три условия:
1) число не меньше 10;

2) число меньше 100;
3) число делится на 5, то есть остаток от его деления на 5 равен нулю.
В условном операторе эти три простых условия должны быть
связаны с помощью двух операций «И»:

```
...
if 10 <= a and a < 100 and a % 5 == 0:
  print( "Да!" )
else:
 print( "Нет." )
```

Пример 3: Сложное условие с операцией «ИЛИ»
Узнать, полетит ли самолет в этот день недели. Самолеты летают по вторникам и пятницам

```
…
if day == 2 or day == 5:
  print( "Полетит!" )
else:
  print( "Нет рейса." )
```

Пример 4: Сложное условие с операцией «НЕ»
Узнать, есть ли уроки физической культуры в этот день недели. Урок физической культуры есть во вторник, четверг и субботу.

```
…
if not( day == 2 or day == 4 or day==6 ):
  print( "Есть урок" )
else:
  print( "Урока нет" )
```

Если в сложном условии встречается несколько разных операций, они выполняются в следующем порядке:
1) операции в скобках;
2) операции «НЕ»;
3) операции «И»;
4) операции «ИЛИ».

Задачи:
1. Напишите программу, которая получает с клавиатуры три целых числа и находит наибольшее и наименьшее из них.
2. Напишите программу, которая получает четыре числа – рост 4-х школьников, и выводит сообщение «По росту», если числа введены по возрастанию (неубыванию), или сообщение «Не по росту!», если они введены в другом порядке.
3. Напишите программу, которая получает с клавиатуры целое число и выводит ответ на вопрос: «Верно ли, что было получено трёхзначное число?».
4. Напишите программу, которая определяет, принадлежит ли число x отрезку $[a; b]$. Все числа вещественные, значения x, a и b вводятся с клавиатуры. Разработайте два варианта программы: с использованием вложенных условных операторов и со сложным условием.
5. Напишите программу, определяющую, какое из двух введенных слов с клавиатуры больше.

Практическое занятие №6 «Циклические алгоритмы»

Что такеое циклы?

Цикл – это алгоритм, в котором несколько раз выполняется последовательности действий в зависимости от условия (или заранее известно количество повторяемых действий). Сейчас мы познакомимся только с простыми программами, которые используют цикл с переменной (или с параметром, или со счетчиком).

Циклы с переменной

Цикл, в котором заранее задается число повторений тела цикла (действия внутри цикла).

Пример 1: 10 раз выводит слово привет

```
for i in range(10):
    print( "привет" )
```

Здесь слово *for* означает «для», переменная *i* (её называют *переменной цикла*) изменяется в диапазоне (**in range**) от 0 до 10, не включая 10 (то есть от 0 до 9 включительно). Таким образом, цикл выполняется для *i* = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 – ровно 10 раз.
Можно было записать этот цикл и по-другому:

```
for i in [0,1,2,3,4,5,6,7,8,9]:
    print( "привет" )
```

В квадратных скобках через запятую перечислены все значения переменной, при которых выполняется цикл. Если их много, такой способ неудобен, лучше использовать встроенную функцию **range**.

Пример 2: Вывод чисел 2 в степени от 0 до 10

```
for i in range(0,11):
   print(2**i)
```

Пример 3: Вывод квадратов чисел от 10 до 0 (в два столбика)

```
for k in range(10,0,-1):
    print( k, k*k )
```

В этом примере шаг равен –1, но шаг может быть любым целым числом.

Пример 4: Находим сумму всех нечетных чисел от 1 до N.

```
n=int(input("n:"))
s=0
for k in range(1,n+1,2):
     s=s+k
print( s )
```

Задачи:

1.Составить программу вывода на экран звездочек («*») в шахматном порядке.
2. Составьте программу, выводящую на экран квадраты чисел от 10 до 20.
3. Составьте программу, которая вычисляет сумму чисел от 1 до 100.
4. Составьте программу, которая вычисляет сумму чисел от 1 до N. Значение N вводится с клавиатуры.
5. С клавиатуры вводится натуральное число *N*. Программа должна найти факториал этого числа (обозначается как *N*!) – произведение всех натуральных чисел от 1 до *N*.

Практическое занятие №7 «Циклические алгоритмы. Графика на Python»

А теперь давайте зациклим две графические команды **t.forward(x)** и **t.left(90)**, где x будет меняться от 1 до 100.

Примеры программ:

1а) «Спираль»

```
import turtle
t=turtle.Pen() # использовуем букву t в качестве обозначения ручки черепашки
for x in range(100):
    t.forward(x)
    t.left(90)
```

или

1б) «Спираль»

```
from turtle import*
shape("turtle") # использовуем внешний вид – черепашку*
for x in range(100):
    forward(x)
    ritght(90)
```

Сделаем нашу графику красочной! В строке **shape("classic")** будет теперь бегать не черепашка, а простая галочка. **color("red")** или **pencolor("red")** задаем цвет рисования, а **bgcolor("black")** задаем цвет фона. А посмотрите что будет если изменить угол поворота всего на 1 градус - **left(91)**.

2) «Спираль-2»

```
from turtle import*
shape("classic")
bgcolor("black")
color("red") #pencolor("red")
for x in range(150):
    forward(x)
    left(91)
```

3) «Спираль-3» А теперь сделаем список цветов!*

```
from turtle import*
shape("classic")
bgcolor("black")
colors=["red","yellow", "blue","green"]
for x in range(150):
    pencolor(colors[x%4])
    forward(x)
    left(91)
```

Задачи:

1.а) Составить программу движения черепашки влево и вправо; б) Составить программу движения черепашки влево и вправо, которая рисует разными красками (несколько раз с помощью цикла).

2. Составить программу рисования нескольких окружностей или прямоугольников* в разных местах экрана.

3. Составить программу построения нескольких окружностей разного радиуса и разными красками.

4. Составить программу построения «звездного неба»*.

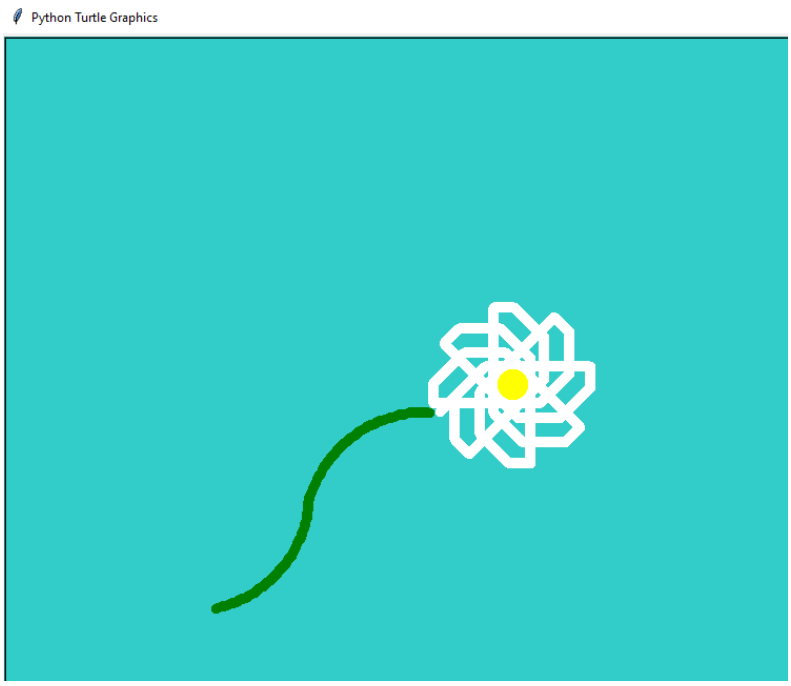
5. Составить программу построения творческого рисунка (используем цикл и краски).

Практическое занятие №8 «Творческий проект»

Задание: Используя команды модуля Turtle языка Python создайте творческую программу.

Пример программы ***«Цветок»***

```
from turtle import*
bgcolor("#33CDC9") #[2]
width(10)
color("white")
speed(13)
for i in range(10):
    forward(20)
    left(90)
    forward(100)
    left(45)
    forward(40)
    left(45)
    forward(40)
    left(45)
forward(20)
left(90)
forward(100)
left(45)
forward(40)
forward(20)
color("green")
right(40)
up()
forward(10)
down()
for i in range(15):
      forward(5)
      right(10)
      forward(5)
      left(15)
for i in range(15):
      forward(5)
      left(10)
      forward(5)
      right(15)
up()
right(100)
forward(200)
right(82)
forward(265)
down()
color("yellow")
dot(30,"yellow")
```



([2]цвет подбираем с помощью редактора Paint и Калькулятора – переводим в 16-ую систему счисления каждый цвет)

Список литературы:

1. Бриггс, Джейсон Python для детей. Самоучитель по программированию / Джейсон Бриггс; пер. с англ. Станислава Ломакина ; [науч. ред. Д. Абрамова]. — М. : Манн, Иванов и Фербер, 2017. — 320 с.
2. Шуман Х.-Г. Python для детей / пер. с нем. М. А. Райтман. – М.: ДМК Пресс, 2019. – 344 с.: ил.
3. Пэйн, Брайсон Python для детей и родителей / Пэйн, Брайсон; пер. с англ. Райтмана; - Москва: Издательство «Э», 2017. – 352 с.: ил.
4. http://server.179.ru/tasks/python/2017b1/pgm12.5_Turtle.html
5. http://itrobo.ru/programmirovanie/python/grafika-turtle-cherepashka-v-piton.html
6. http://31415.ru/blog/python/
7. https://kpolyakov.spb.ru//download/8-3_python.pdf