МЕТОДИЧЕСКАЯ РАЗРАБОТКА

**«Основы программирования на языке Python 8.0»**

Автор - **Верещагин Александр Михайлович**,

учитель информатики и ИКТ

Муниципальное бюджетное общеобразовательное учреждение «Ильинская средняя общеобразовательная школа»

Адрес: 165680, Архангельская область, Вилегодский район, с. Ильинско-Подомское, ул. Ленина, 7.

2020-10-10

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Современному учителю приходится постоянно очень много работать по самообразованию – такова реальность. Учебники, даже самые хорошие, никак не успевают отслеживать новое. Стиль изложения в них тоже часто не самый понятный. Особенно это можно сказать об информатике, где обновление необходимых знаний столь стремительно, что очень трудно держать себя в форме.

Думаю, разработки, которые написаны опытными учителями, проверенные на ребятах, могут ускорить понимание нового материала. В моей работе я хочу дать одну из возможных форм для обучения с нуля основам Пайтона. Проверено: усвоение идёт быстро, так как подборка задач достаточно комфортная для учащихся 10 класса, знающего основы Паскаля. По ней можно начать преподавать новый язык и более младшим школьникам.

**1-я инструкция по языку Python 3.7.4-3.8.0**

### 0. История. Язык начал разрабатывать программист, Гвидо ван Россумом, в конце 1980-х. На тот момент он работал в центре математики и информатике в Нидерландах. **В феврале 1991 исходный код языка был опубликован на alt.sources**. **3 декабря 2008 в релиз вышла третья версия Python, которая является основной до сих пор**.

### Питон или Пайтон?

Будь то название британского телешоу или английское звучание слова “змея”, Python правильно произносить, как Пайтон. Однако, около 80% Российского сообщества привыкли использовать слово “Питон”.

Нельзя сказать, что однозначно правильно использовать один из вариантов, многие названия адаптируются под произношения конкретного языка, а изменить сложившиеся привычки общества очень сложно. Однако, вариант названия “Питон” уместно употреблять только в разговоре с русскоязычными собеседниками, потому что на любой международной конференции значение слова “Питон” просто не поймут, ведь в английском языке его нет, есть только “Python (Пайтон)”.

## Логотип

На логотипе изображены две змеи, образующие квадрат с выпуклым центром, это часто вводит в заблуждение пользователей, вынуждая ассоциировать название языка с рептилией.



Логотип создал брат автора, Юст ван Россум — программист и шрифтовой дизайнер. Он разработал как дизайн логотипа (две змеи), так и шрифт текста Flux Regular.

Питон используют во многих областях программирования, поэтому на нём можно написать что угодно.

Подробно об языке можно почитать в Интернете, в частности, по этому адресу:

[**https://all-python.ru/osnovy/yazyk-programmirovaniya.html**](https://all-python.ru/osnovy/yazyk-programmirovaniya.html)

**1. Установка Питона.**

**Шаг 1: Загрузка установочного файла Python 3.** Открываем окно браузера и переходим на **python.org**;

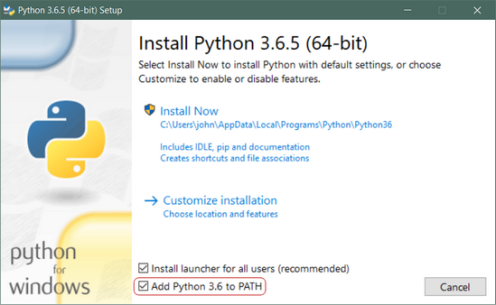
## Под верхним заголовком, где написано Python Releases for Windows, нажимаем на ссылку к последней версии Python 3.x.x.; листаем вниз и выбираем установочный файл Windows x86-64 для 64-разрядной версии операционной системы или файл Windows x86 для 32-разрядной.

## Шаг 2: Запускаем установочный файл

После того, как вы выбрали установочный файл и загрузили его, просто запустите его двойным нажатием Должен открыться диалог, который выглядит следующим образом:

**Важно!** Убедитесь в том, что вы установили флажок в разделе **Add Python 3.x to PATH**, чтобы интерпретатор был установлен в вашем пути выполнения…

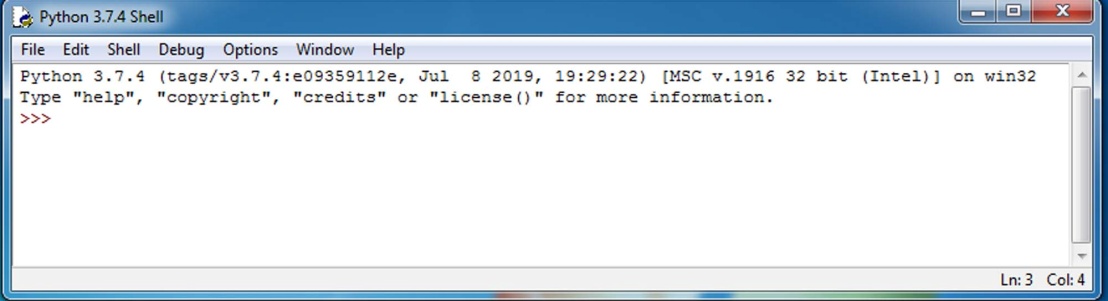
**Проблемы при установке бывают!** Прежде всего, на компьютере должно быть установлено обновление Windows 7 **Service Pack 1**. Ещё: возможно, не хватает библиотеки **api-ms-win-crt-runtime-l1-1-0.del** (в папке Windows/System32). Может помочь утилита **vc.redist.x86**… Интернет – к услугам!



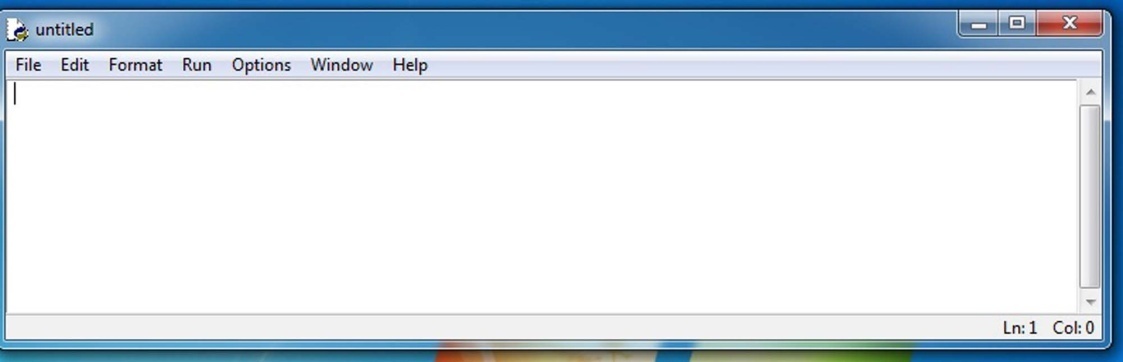
*Иногда, при автоматическом обновлении операционной системы, всё сразу идёт гладко! Всё программное обеспечение для домашнего компьютера можно взять в кабинете информатики. Кстати. Существуют и мобильные версии Python!*

# 2. Среда обработки IDLE. Заходим в меню Пуск – Все программы – Python 3.8… и выносим на Рабочий стол ярлык IDLE (Python 3.8 32 bit).

После [загрузки и установки Python](https://pythonworld.ru/osnovy/skachat-python.html) открываем IDLE (среда разработки на языке Python, поставляемая вместе с дистрибутивом). I**DLE** (Integrated Development and Learning Environment) — это интегрированная **среда разработки** и обучения на языке Python, созданная с помощью библиотеки Tkinter. I**DLE** (Integrated Development and Learning Environment[[2]](https://ru.wikipedia.org/wiki/IDLE#cite_note-2)) — это [интегрированная среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) и обучения на языке [Python](https://ru.wikipedia.org/wiki/Python), созданная с помощью библиотеки [Tkinter](https://ru.wikipedia.org/wiki/Tkinter). Официально — искажение [IDE](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8), но на самом деле названа в честь [Эрика Айдла](https://ru.wikipedia.org/wiki/%D0%90%D0%B9%D0%B4%D0%BB,_%D0%AD%D1%80%D0%B8%D0%BA) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Eric* ***Idle***) из [Монти Пайтон](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D1%82%D0%B8_%D0%9F%D0%B0%D0%B9%D1%82%D0%BE%D0%BD).



В меню выбираем File – New file и пишем построчно 1-ю программу. #означает «комментарий».



Набрали первые 3-4 строки. Затем File – Save as… (под **именем 01** сохраняем в папке C://Python). Затем F5 (исполнение программы). Откроется окно **Python 3.8 Shell.** Вводим значения переменных a и b через клавишу **Enter** и видим ответы. Пошагово добираем строки и исполняем. Так быстро можно понять основные операции с числами (многие отличаются от операций в Паскале).

**1-я программа. Основные функции для работы с числами.**

#int - целое число, float - действительное число.

a=float(input())

b=float(input())

print('сумма:',a+b)

print('разность:',a-b)

print('произведение:',a\*b)

print('частное:',a/b)

print('целочисленное деление:',a//b)

print('остаток от деления:',a%b)

print('округление с точностью до 2 знаков после запятой:',round(a/b,2))

print('квадрат числа:',a\*\*2)

print('степень числа:',a\*\*b)

print('степень числа иначе:',pow(a,b))

print('квадратный корень из неотрицательного числа',pow(a,0.5))

import math

print('модуль числа:',math.fabs(-a))

***Если всё у тебя получилось – молодец! Поздравляю с началом изучения нового языка!***

**2-я инструкция по языку Python 3.7.4-3.8.0**

**3.** **Числа.** Каждое значение в Python имеет **тип**. Поскольку всё в Python — объекты, типы являются классами, а значения — экземплярами (объектами) этих классов. Целые числа со знаком, числа с плавающей запятой и комплексные числа относятся к группе чисел. В Python они представлены классами int, float и complex. Мы можем использовать функцию type(), чтобы узнать класс переменной: **print(type(5))** **print(type(5.2))** – результатами соответственно будут **int** и **float**. О комплексных числах – позднее.

## Переменные Python не требуют явного объявления для резервирования пространства памяти. Объявление присваивается автоматически, когда вы присваиваете значение переменной. Знак равенства (=) используется для присвоения значений переменным. Множественное присвоение: Python позволяет одновременно назначать одно значение нескольким переменным.

Например, **a = b = c = 1**

Здесь создается целочисленный объект со значением 1, и все три переменные назначаются в одну и ту же ячейку памяти. Вы также можете назначить **несколько объектов нескольким переменным**. Например,

**a, b, c = 2, 5, "sky"**

Переменным **a** и **b** присваиваются целочисленные значения **2** и **5** соответственно, а переменной **c** присваивается строка **sky**.

**Распространенные операторы сравнения:**

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Значение** | **Пример** |
| **>** | Больше | 5 > 4 правильно (true) |
| **<** | Меньше | 4 < 5 правильно  (true) |
| **>=** | Больше или равно | 4 >= 4 правильно (true) |
| **<=** | Меньше или равно | 3 <= 4 правильно (true) |
| **==** | Равно | 1 == 1 правильно  (true) |
| **!=** | Неравно | 1 != 2 правильно (true) |
| **and** | И | a > 4 and a < 10 |
| **or** | Или | a == 1 or a == 4 |

**4**. **Организация ветвления.**

**2-я программа.** Вводится целое число **x**. В зависимости от его знака вычислить соответствующее значение переменной **y** с помощью оператора **if.** Таким способом можно организовать много проверок, хотя это и не будет рациональным:

x=int(input())

if x>0: y=x-2

if x==0: y=100

if x<0: y=x\*x

print('y=',y)

**3-я программа.** Оператор **if – else.** В случае 2-х «дорог» такой способ удобен для применения:

x=int(input())

if x>0: y=x-2

else:y=x\*x

print('y=',y)

**4-я программа. Множественное ветвление if-elif-else.** 2-я задача эффективно решается с помощью ниже описанного приёма – вложения условных операторов друг в друга.

x=int(input())

if x>0:

y=x-2

else:

if x==0:

y=100

else:

y=x\*x

print('y=',y)

Одна интересная особенность: аккуратно с табуляцией! 2-е написание if-else предполагает отступы в 4 пробела (именно 4). Иначе – ошибка при исполнении.

**5-я программа.** Ещё раз понять вложенные операторы поможет исполнение следующей программы, которую рекомендую набрать, не глядя в инструкцию. Сюжет можно разнообразить своими выдумками.

print('Ты школьник?')

old = int(input('Введи свой возраст: '))

if 3 <= old < 7:

print('Ты в садике')

else:

if 7 <= old < 11:

print('Ты в младших классах.')

else:

if 12 <= old <= 18:

print('Ты уже не маленький.')

else:

if 18 < old:

print('Ты выпускник!')

**6-я программа. Применение конструкции elif.**

Ввести любоецелое число. Сообщить, является ли оно положительным, нулём или отрицательным.

n = int(input('Введите число: '))

if n>0:

print('Положительное')

elif n==0:

print('0')

else:

print('Отрицательное')

**7-я программа. Целочисленное деление и остаток от деления. Ответ с проверкой.**

Ввести 2 числа **a** и **b**. Найти результат целочисленного деления первого числа на второе и остаток от деления первого на второе. Если **первое** число **а** делится нацело на 7, сообщить "да", иначе - "нет". Если **первое** число оканчивается цифрой 5, то также сообщить "да", иначе - "нет".

a=int(input())

b=int(input())

c=a//b

d=a%b

print('целочисленное деление:',c)

print('остаток от деления:',d)

if a/%7==0:print('da')

else:print('net')

if a%10==5:print('da')

else:print('net')

Исполним. Полученные ответы ожидаемые или что-то не так? В чём же дело? Поэкспериментируем…

**3-я инструкция по языку Python 3.7.4-3.8.0**

**5 блок.** **Цикл.**

**While** - один из самых универсальных циклов в Python. Выполняет тело цикла до тех пор, пока условие цикла истинно. Как правило, **цикл while** используется, когда невозможно определить точное значение количества проходов исполнения **цикла**.

**8 программа. Количество цифр в числе.**

Ввести с клавиатуры натуральное число **n**. Определить, сколько цифр оно содержит?

n = int(input())

length = 0 # Длина по-английски. Знак комментария мы уже знаем.

while n > 0: # После условия цикла ставится двоеточие.

n //= 10 # это эквивалентно n = n // 10. И 2 способа записи отсечения последней цифры.

length += 1 # А так делается накопление суммы. В Паскале мы бы записали length:=length + 1.

print(length)

Исполним программу, применив оба способа отсечения.

**9 программа. Выход из цикла с помощью оператора break.**

Вводятся целые числа и сообщаются на экран до тех пор, пока не введём 0. Если все введённые числа были положительными, появится соответствующая запись. При вводе отрицательного числа сразу происходит принудительный выход из цикла с сообщением.

a = int(input())

while a != 0:

if a < 0:

print('Встретилось отрицательное число', a)

break

a = int(input())

else:

print('Ни одного отрицательного числа не встретилось')

**10 программа. Сколько раз выполняется тело цикла?**

Запишите число, которое будет напечатано в результате выполнения следующей программы (кстати, это одна из ЕГЭ-задачек в 11-м классе, идущая под номером 8). Тело цикла наберём с отступом (хотя бы на 1 пробел, лучше – на 4 для лучшей узнаваемости именно тела цикла). На экзамене её надо уметь решать без компьютера: найти количество итераций и применить математические знания последовательностей.

Решим на бумаге (для тренировки) и проверим ответ, набрав программу на Питоне.

k = 5

s = 3

while k < 99:

s = s + k

k = k + 2

print(s)

**Заметим,** что в цикле while **может быть счётчик** (в 10-й задаче его роль выполняет переменная k), регламентирующий количество итераций (выполнений тела цикла). Но **наличие счётчика необязательно**: в 9-й задаче его нет, количество итераций зависит от вводимых чисел.

**11 программа**. Применение **цикла For.**

Английское **range** означает **диапазон**. Что сообщит при исполнении следующая программа? Здесь k «пробегает» целые (и только целые) значения от 6 до 10(!) с шагом 1 по умолчанию: верхний конец диапазона в него **не входит**.

s = 0

for k in range(6,11):

    s = s + 10

print (s)

Вы можете использовать аргумент **шаг** для увеличения в сторону больших чисел. Это называется **инкрементация**.

for i in range(3, 100, **25**):

    print(i)

Если ваш шаг отрицательный, то вы двигаетесь по ряду убывающих чисел, это называется **декрементация**. Это позволяет вам идти вспять. В следующем примере ваш шаг будет **-2**. Это значит, что декрементация будет равна 2 для каждого цикла:

for i in range(10, -6, **-2**):

print(i)

#### Все три конструкции объединим в одну программу, исполним её, поменяем числа

#### для полного понимания… Итак, функция range имеет следующие формы:

* range(stop): возвращает все целые числа от 0 до stop
* range(start, stop): возвращает все целые числа в промежутке от start (включая) до stop (не включая). Выше в программе факториала использована именно эта форма.
* range(start, stop, step): возвращает целые числа в промежутке от start (включая) до stop (не включая), которые увеличиваются на значение step.

**12 программа.** Вычислить **факториал для введённого с клавиатуры числа.**

number = int(input("Введите число: "))

factorial = 1

for i in range(1, number+1):

factorial \*= i # В привычной форме записали бы так: factorial= factorial\*i

print("Факториал числа", number, "равен", factorial)

***В Python строку можно заключать как в одиночные, так и в двойные кавычки.***

**13 программа.** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 2. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа по мо­ду­лю не пре­вы­ша­ют 30 000.

n=int(input())

k=0

for i in range(1,n+1):

a=int(input())

if a%3==0 and a%10==2:

k=k+1

print('k=',k)

Ограничения «1000» и «30000» указывают на важность выбора типа переменных (актуально для Паскаля). В Пайтоне-Питоне это несущественно: указываем тип **int**.

**4-я инструкция по языку Python 3.7.4-3.8.0**

**6 блок.** **Задачи на применение цикла и ветвления.**

*Тренироваться можно так: прочитать и запомнить условие задачи, а затем попробовать написать программу самому. При затруднении можно взглянуть на инструкцию.*

**14 программа.** Склавиатуры **вводится n** натуральных чисел, каждое не больше 1000. Подсчитать **сумму** введённых чисел, каждое из которых **делится нацело на 6** и **оканчивается цифрой 8**.

n=int(input())

s=0

for i in range(1,n+1):

a=int(input())

if a%6==0 and a%10==8:

s+=a

Если при исполнении будут появляться ошибки, проверьте отступы – смещение зависимых строк вправо (можно на 4 пробела).

print('s=',s)

**15 программа. Вводится** число n (n<=6). Затем вводятся n натуральных чисел, каждое из которых меньше 100. Найти **наименьшее** число, которое к тому же **оканчивается на 4**. Гарантировано, что хотя бы одно такое число есть.

n=int(input())

min=100

for i in range(1,n+1):

a=int(input())

if a%10==4 and a<min:

min=a

print('min=',min)

**16 программа**. **Вводится** число n (n<=6). Затем вводятся n натуральных чисел, каждое из которых меньше 1000. Найти **наибольшее** число, которое к тому же **делится на 3**. Гарантировано, что хотя бы одно такое число есть.

n=int(input())

max=0

for i in range(1,n+1):

a=int(input())

if a%3==0 and a>max:

max=a

print('max=',max)

**17 программа.** На вход подаются целые числа, по модулю <=30000. Их количество неизвестно (не более 1000), введение числа **0 – окончание ввода**. Найти произведение всех чисел.

p=1

a=int(input())

while a!=0:

p\*=a

a=int(input())

print('p=',p)

**18 программа.** На вход подаются целые числа, по модулю <=30000. Их количество неизвестно (не более 1000), введение числа **0 – окончание ввода**. Найти произведение чисел, которые являются чётными **или** оканчиваются на 5. Гарантировано, что хотя бы одно число с таким свойством имеется.

p=1

a=int(input())

while a!=0:

if (a%2 ==0 **or** a%10==5):

p\*=a

p=1

a=int(input())

while a!=0:

if (a%2 ==0 or a%10==5):p\*=a

a=int(input())

print('p=',p)

Пройдёт ли версия, предложенная сходу (см. слева)? Вообще, с отступами не всё ясно для начинающего. А эта версия? Оформим по-своему, проверим…

a=int(input())

print('p=',p)

**19 программа.** На вход подаются целые числа, по модулю <=30000. Их количество неизвестно (не более 1000), введение числа **0 – окончание ввода**. Найти произведение чисел, которые являются чётными **или** оканчиваются на 5. Если таких чисел совсем нет, сообщить «No».

p=1

a=int(input())

while a!=0:

if (a%2 ==0 or a%10==5):p\*=a

a=int(input())

if p!=1:print('p=',p)

else:print('No')

**20 программа**. Уче­ни­ки 9 клас­са вели днев­ни­ки на­блю­де­ния за по­го­дой и еже­днев­но за­пи­сы­ва­ли днев­ную температуру. Най­ди­те самую низ­кую тем­пе­ра­ту­ру за время наблюдения. Если тем­пе­ра­ту­ра опус­ка­лась ниже –33 градусов, вы­ве­ди­те YES, иначе вы­ве­ди­те NO. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство дней, в те­че­ние ко­то­рых про­во­ди­лось из­ме­ре­ние тем­пе­ра­ту­ры N (1 ≤ N ≤ 31), затем для каж­до­го дня вво­дит­ся температура.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 4  –5  12  –2  8 | –5  NO  min=100  flag=0  n=int(input())  for i in range(1,n+1):  a=int(input())  if a<min: min=a  if a<-33:flag=1    print('min=',min)  if flag==1:print('Yes')  else:print('No') |

**5-я инструкция по языку Python 3.7.4-3.8.0**

**7 блок.** **Знакомство со строками.**

Строка — базовый тип, представляющий из себя неизменяемую последовательность символов; str от «string» — «строка».

**21 программа. Основные действия со строками.**

s='informatika'

print(s)

s1='for'

print('s1=',s1)

s2='ma'

print('s2=',s2)

s3=s1+s2 # слияние строк.

print('s3=',s3)

s4=3\*s2 # оператор умножения строки на натуральное число.

print('s4=',s4) # если множитель - 0 или отриц. , будет пустая строка.

if s2 in s:print('true') # оператор принадлежности подстроки данной строке.

else:print('false') # печатается истина или ложь.

if s2 not in s:print('false') # наоборот -

else:print('true') # печатается ложь или истина.

# индексы строк в python начинаются с нуля!

print(s[2:7]) # вырезка - s[m:n] возвращает часть s, начинающуюся с позиции m,

# и до позиции n, но не включая позицию. Напечатается что?

print(s[:7]) # Если пропустить первый индекс, срез начинается с начала

# строки. Таким образом, s[:m] = s[0:m].

print(s[7:]) # Аналогично, если опустить второй индекс s[n:],

#срез длится от первого индекса до конца строки.

print(s[:7]+s[7:]) # Для любой строки s и любого целого n числа (0 ≤ n ≤ len(s)),

# s[:n] + s[n:]будет s.

t=s[7:7] # Если первый индекс в срезе больше или равен второму индексу,

print(t) # то генерируется пустая строка.

s5=s[0:8:2] #Добавление дополнительного : и третьего индекса означает

print('s5=',s5) #шаг, который #указывает, сколько символов следует пропустить

#после извлечения каждого символа в срезе.

print(len(s)) #Функция len(s) возвращает длину строки.

print(s[-1]) #индексы строк также могут быть указаны отрицательными числами. В этом случае

#индексирование начинается с конца строки: -1 относится к последнему символу,

#-2 к предпоследнему и так далее.

s6=s.replace('i',"") # удаление определённого символа во всей строке.

print('s6=',s6)

Наберём эту программу построчно: набрав несколько строк, запустим программу на исполнение, затем доберём пару строк и снова запустим… Так и познакомимся с основными действиями.

**22 программа. Коды символов и нахождение символов по их кодам.**

print(ord('0')) # У каждого символа есть код.

print(ord('A'))

print(chr(48)) # По коду можно найти соответствующий ему символ.

print(chr(65))

x=str(input()) # Вводится какой-то символ и находится его код.

print('код символа будет:',ord(x))

print('')

for i in range(65,91):

print(chr(i))

**ASCII** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) ***A****merican****s****tandard****c****ode for****i****nformation****i****nterchange*, [[ˈ](https://ru.wikipedia.org/wiki/%D0%97%D0%BD%D0%B0%D0%BA_%D1%83%D0%B4%D0%B0%D1%80%D0%B5%D0%BD%D0%B8%D1%8F_(%D1%81%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB_%D0%9C%D0%A4%D0%90))[æ](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BD%D0%B0%D0%BF%D1%80%D1%8F%D0%B6%D1%91%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BD%D0%B5%D0%BE%D0%B3%D1%83%D0%B1%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B3%D0%BB%D0%B0%D1%81%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D0%B5%D0%B3%D0%BE_%D1%80%D1%8F%D0%B4%D0%B0_%D0%BD%D0%B8%D0%B6%D0%BD%D0%B5%D0%B3%D0%BE_%D0%BF%D0%BE%D0%B4%D1%8A%D1%91%D0%BC%D0%B0)[s](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D1%83%D1%85%D0%BE%D0%B9_%D0%B0%D0%BB%D1%8C%D0%B2%D0%B5%D0%BE%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%B1%D0%B8%D0%BB%D1%8F%D0%BD%D1%82)[.](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D1%84%D0%BE%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82#%D0%A1%D1%83%D0%BF%D1%80%D0%B0%D1%81%D0%B5%D0%B3%D0%BC%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B5_%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0)[k](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D1%83%D1%85%D0%BE%D0%B9_%D0%B2%D0%B5%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B9_%D0%B2%D0%B7%D1%80%D1%8B%D0%B2%D0%BD%D0%BE%D0%B9_%D1%81%D0%BE%D0%B3%D0%BB%D0%B0%D1%81%D0%BD%D1%8B%D0%B9)[i](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BE%D0%B3%D1%83%D0%B1%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B3%D0%BB%D0%B0%D1%81%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D0%B5%D0%B3%D0%BE_%D1%80%D1%8F%D0%B4%D0%B0_%D0%B2%D0%B5%D1%80%D1%85%D0%BD%D0%B5%D0%B3%D0%BE_%D0%BF%D0%BE%D0%B4%D1%8A%D1%91%D0%BC%D0%B0)][[1]](https://ru.wikipedia.org/wiki/ASCII#cite_note-1)) — название таблицы (кодировки, набора), в которой некоторым распространённым печатным и непечатным символам сопоставлены числовые коды. Название «ASCII» по-русски часто произносится как [*аск(и)и́*]

## Базовая таблица кодировки ASCII

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 32 пробел | 48 0 | 64 @ | 80 P | 96 ` | 112 p |
| 33 **!** | 49 1 | 65 A | 81 Q | 97 a | 113 q |
| 34 **“** | 50 2 | 66 B | 82 R | 98 b | 114 r |
| 35 # | 51 3 | 67 C | 83 S | 99 c | 115 s |
| 36 $ | 52 4 | 68 D | 84 T | 100 d | 116 t |
| 37 % | 53 5 | 69 E | 85 U | 101 e | 117 u |
| 38 & | 54 6 | 70 F | 86 V | 102 f | 118 v |
| 39 ‘ | 55 7 | 71 G | 87 W | 103 g | 119 w |
| 40 ( | 56 8 | 72 H | 88 X | 104 h | 120 x |
| 41 ) | 57 9 | 73 I | 89 Y | 105 i | 121 y |
| 42 \* | 58 : | 74 J | 90 Z | 106 j | 122 z |
| 43 + | 59 ; | 75 K | 91 [ | 107 k | 123 { |
| 44 , | 60 < | 76 L | 92 \ | 108 l | 124 | |
| 45 — | 61 = | 77 M | 93 ] | 109 m | 125 } |
| 46 . | 62 > | 78 N | 94 ^ | 110 n | 126 ~ |
| 47 / | 63 ? | 79 O | 95 \_ | 111 o | 127 |

**Расширенная таблица кодов ASCII (128-255)** содержит национальные алфавиты, в частности, кириллицу. Например, заглавная буква А русская имеет код 128, значит, Б – 129 и т. д. Это можно узнать с помощью программы 22.

Заключение. Эта практическая разработка - 1-я часть задуманного. После апробации 2-й части на 3-м потоке обучающихся можно будет и её предложить коллегам.