PYTHON
Функции (введение)
И
СИМВОЛЬНЫЕ СТРОКИ
(часть 2)

Замена букв в строке; Сравнение строк; Кодировка UNICODE; Коды символов; Вывод таблицы символов; Методы: upper(); lower(); count(); find() rfind(); index(); isalpha(); split();

split()

(Лекция 7)

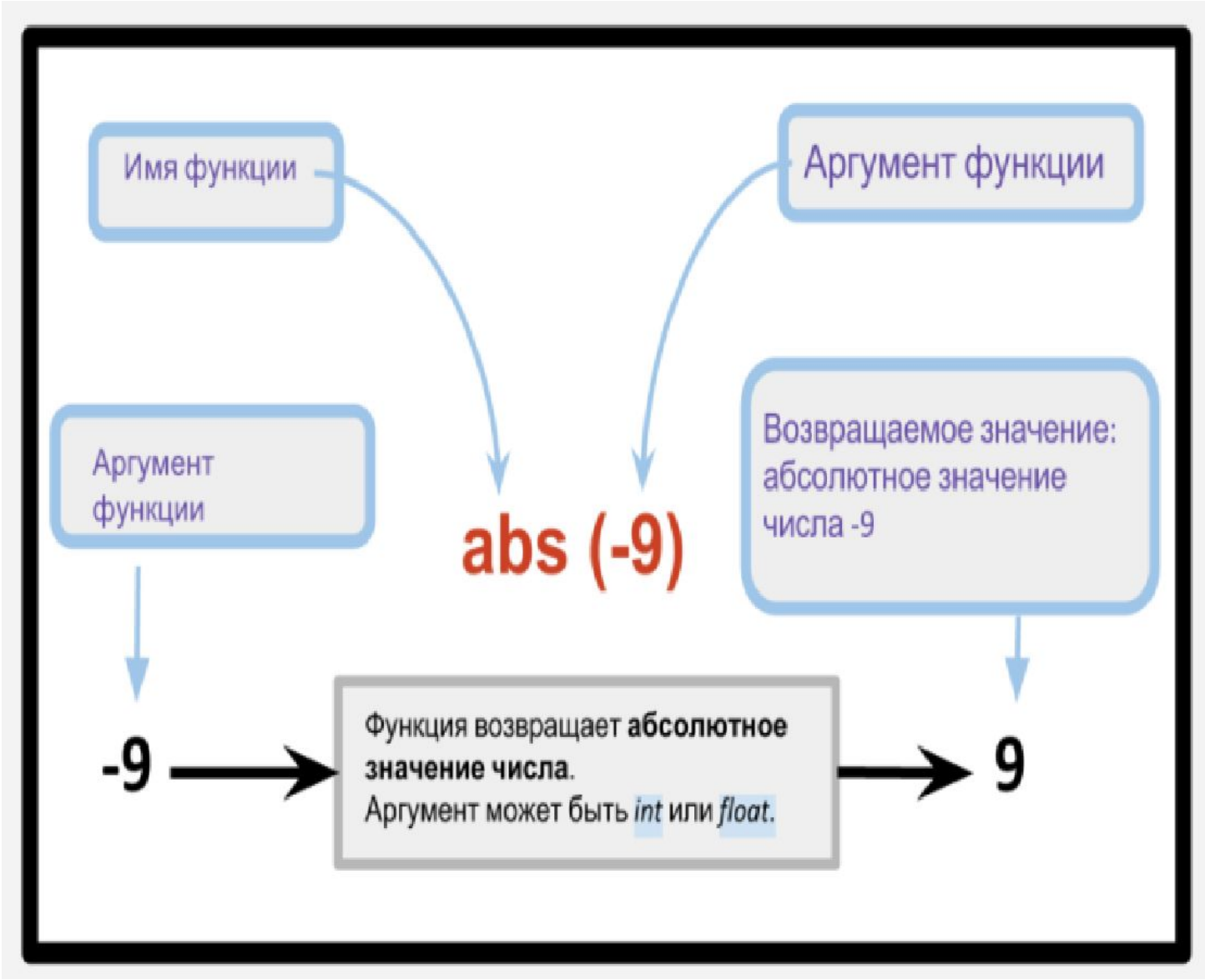
Что такое функция

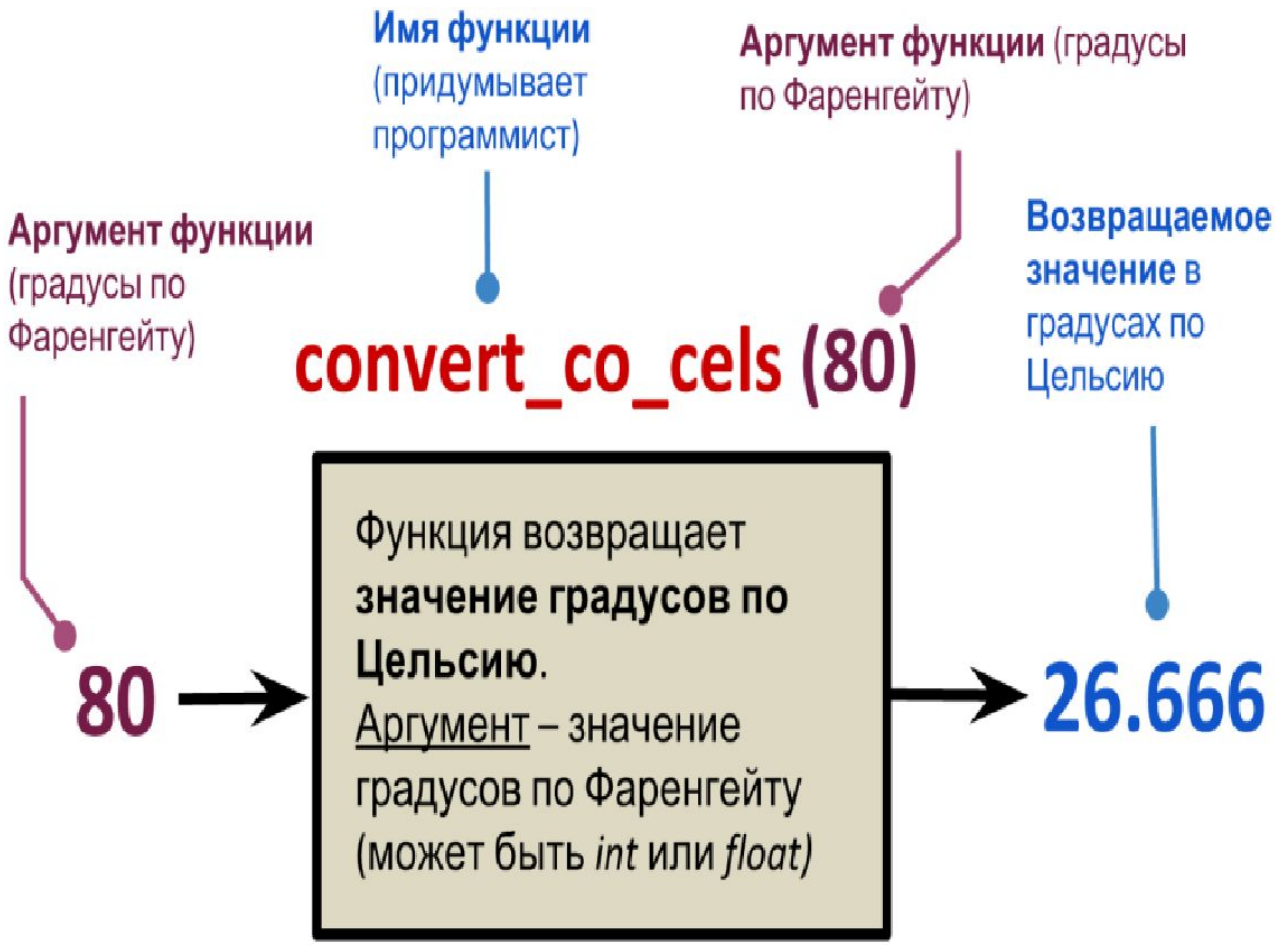
Функция — это блок кода, который можно многократно вызывать на выполнение. Она является фундаментальной частью любого языка программирования.

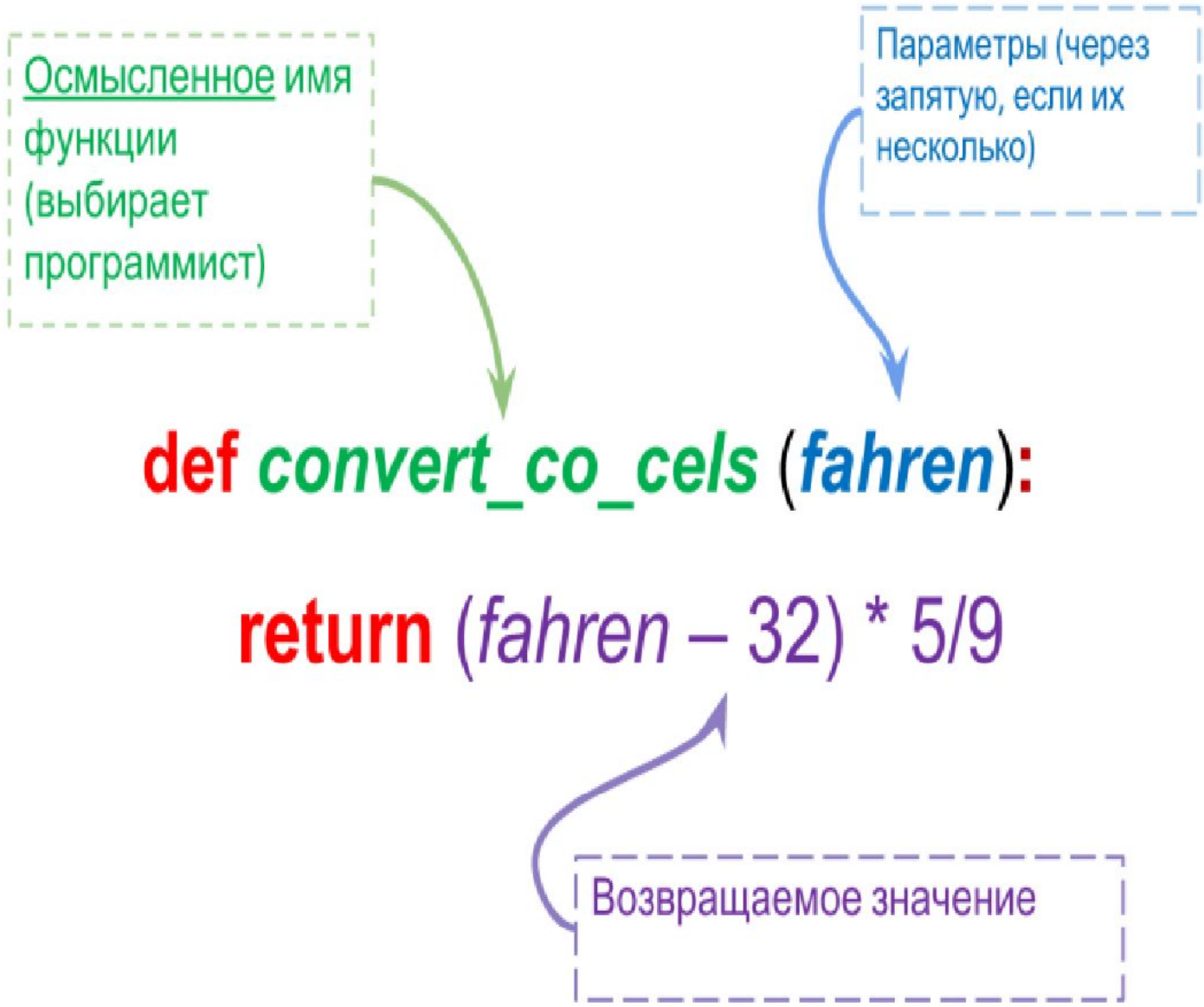
Функция позволяет разделять программу на самостоятельные, но связанные части. Программисты используют функции, чтобы сделать программу модульной и избежать повторения кода.

Функция может использоваться для обработки данных, она получает на вход значения, обрабатывает его и возвращает результат в программу. Также она может не возвращать значение, а выводить его на экран или записывать в файл.

Программист может написать собственную функцию или использовать готовые решения языка, если они есть, конечно. Например, лучше самому не написать функцию для определения максимального числа, а воспользоваться стандартной `max()`.


Имя функции
Аргумент функции
Аргумент функции
abs (-9)
Возвращаемое значение: абсолютное значение числа -9
-9
Функция возвращает **абсолютное значение числа**.
Аргумент может быть *int* или *float*.
9


Имя функции (придумывает программист)
Аргумент функции (градусы по Фаренгейту)
Аргумент функции (градусы по Фаренгейту)
Возвращаемое значение в градусах по Цельсию
convert_co_cels (80)
80
Функция возвращает значение градусов по Цельсию.
Аргумент – значение градусов по Фаренгейту (может быть int или float)
26.666


Осмысленное имя функции (выбирает программист)
Параметры (через запятую, если их несколько)
def convert_co_cels (fahren):
return (fahren – 32) * 5/9
Возвращаемое значение

- Ключевое слово **def** для Python означает, что дальше идет описание функции.
- После def указывается им**я функции** convert_co_cels, затем в скобках указывается параметр, которому будет присваиваться значение при вызове функции.
- **Параметры функции** – обычные переменные, которыми функция пользуется для внутренних вычислений.
- Переменные, объявленные внутри функции, называются локальными и не видны вне функции.
- После символа «:» начинается тело функции.
- Выражение, стоящее после ключевого слова return будет возвращаться в качестве

```
# Бывают случаи, когда
# функция ничего не принимает
# на вход и ничего не возвращает
# (не используется ключевое слово return)
def print_hello():
    print('Привет')
    print('Hello')
    print('Hi')

print_hello()
```

```
def f(x):
    x = 2 * x
    return x


print(f(4))
print(f(56))
```

```
# Площадь прямоугольника
def s(a,b):
    s_p=a*b
    return s_p

x=int(input("x="))
y=int(input("y="))
s_pr=s(x,y)
print("x=",x,"y=",y,"S=",s_pr)
```

```
import math
def circle(r):
    return math.pi * r ** 2

def rectangle(a, b):
    return a * b

def triangle(a, b, c):
    p = (a + b + c) / 2
    return math.sqrt(p * (p - a) * (p - b) * (p - c))

choice = input("Круг(к), прямоугольник(п) или треугольник(т): ")
if choice == 'к':
    rad = float(input("Радиус: "))
    print("Площадь круга: %.2f" % circle(rad))
elif choice == 'п':
    l = float(input("Длина: "))
    w = float(input("Ширина: "))
    print("Площадь прямоугольника: %.2f" % rectangle(l, w))
elif choice == 'т':
    AB = float(input("Первая сторона: "))
    BC = float(input("Вторая сторона: "))
    CA = float(input("Третья сторона: "))
    print("Площадь треугольника: %.2f" % triangle(AB, BC, CA))
```

```
import math

def circle(r):
    return math.pi * r ** 2


def rectangle(a, b):
    return a * b


def triangle(a, b, c):
    p = (a + b + c) / 2
    return math.sqrt(p * (p - a) * (p - b) * (p - c))
```

```
choice = input("Круг(к), прямоугольник(п) или треугольник(т): ")
if choice == 'к':
    rad = float(input("Радиус: "))
    print("Площадь круга: %.2f" % circle(rad))
elif choice == 'п':
    l = float(input("Длина: "))
    w = float(input("Ширина: "))
    print("Площадь прямоугольника: %.2f" % rectangle(l, w))
elif choice == 'т':
    AB = float(input("Первая сторона: "))
    BC = float(input("Вторая сторона: "))
    CA = float(input("Третья сторона: "))
    print("Площадь треугольника: %.2f" % triangle(AB, BC, CA))
```

#1 строка - неизменяемый объект

```
s='vasya'
print(s,id(s))#58419968
#s[0]='V'#TypeError: 'str' object does not support item assignment
s_new='V'+s[1:]
print(s_new,id(s_new))#Vasya 62155104
```

#2 строка - неизменяемый объект

```
s='vasya'
print(s,id(s))#58419968
#s[0]='V'#TypeError: 'str' object does not support item assignment
s='V'+s[1:]
print(s,id(s))#Vasya 62155104
```

```
# 3
"""
```

перебираются все символы, входящие в строку s.
В теле цикла проверяем значение переменной c
(это очередной символ исход-ной строки):
если оно совпадает с буквой «э»,
то заменяем его на букву «е»

```
if c == "э": c = "е"
"""
s="Привэт"
sNew = ""
print("sNew",sNew,"id",id(sNew))
for c in s:
    if c == "э": c = "е"
    sNew += c
    print("sNew",sNew,"id",id(sNew))
print( sNew )
```

```
sNew  id 22551264
sNew П id 49870672
sNew Пр id 51270240
sNew При id 51270240
sNew Прив id 51270240
sNew Приве id 51270240
sNew Привет id 49841112
Привет
```

```
# 4
"""
```

Сравнение строк
Строки можно сравнивать
между собой так же, как числа.
Например, можно проверить
равенство двух строк:

```
"""
password = input( "Введите пароль:" )
if password == "Sergey":
    print( "Слушаюсь и повинуюсь!" )
else:
    print( "No pasaran!" )
```

```
# 5 Введите пароль:
psw = "pass"
in_psw = ""
while psw != in_psw:
   in_psw = input("Введите пароль: ")
print("Вход в систему разрешен")
```

```
# -*- coding: utf-8 -*-
"""
 Программа к учебнику информатики для 10 класса
 К.Ю. Полякова и Е.А. Еремина.
 Глава 8.
 Программа № 3. Операторы ввода, вывода, присваивания
 Вход:
   5
   7
 Результат:
   12
"""
a = int ( input() )
b = int ( input() )
c = a + b
print ( c )
```

Сравнение строк

Пар ? пар ? парк

Сравнение по кодам символов:

	0	1	...	8	9
CP-1251	48	49	...	56	57
UNCODE	48	49	...	56	57
	A	**B**	...	**Y**	**Z**
CP-1251	65	66	...	89	90
UNCODE	65	66	...	89	90
	a	**b**	...	**y**	**z**
CP-1251	97	98	...	121	122
UNCODE	97	98	...	121	122

Сравнение строк

	А	Б	...	Ё	...	Ю	Я
CP-1251	192	193	...	168	...	222	223
UNCODE	1040	1041	...	1025	...	1070	1071

	а	б	...	ё	...	ю	я
CP-1251	224	225	...	184	...	254	255
UNCODE	1072	1073	...	1105	...	1102	1103

5STEAM < STEAM < Steam < steam

steam < ПАР < Пар < пАр < пар < парк

Windows-1251

Windows-**1251** — набор символов и кодировка, являющаяся стандартной 8-битной кодировкой для русских версий Microsoft Windows до 10-й версии. В прошлом пользовалась довольно большой популярностью.

Windows-1251 — Википедия

ru.wikipedia.org›Windows-1251

Кодировка UNICODE

Юникод (Unicode) — стандарт кодирования символов, позволяющий представить знаки практически всех письменных языков.

Стандарт предложен в 1991 году некоммерческой организацией «Консорциум Юникода».

В Unicode используются 16-битовые (2-байтовые) коды, что позволяет представить 65536 символов.

Кодировка UNICODE (продолжение)

Применение стандарта Unicode позволяет закодировать очень большое число символов из разных письменностей: в документах Unicode могут соседствовать китайские иероглифы, математические символы, буквы греческого алфавита, латиницы и кириллицы, при этом становится ненужным переключение кодовых страниц.

UTF-8 и Unicode не могут сравниваться.
UTF-8 является кодировкой используется для перевода чисел в двоичные данные.
Unicode - это набор символов используется для перевода символов в числа.

Python 3: всё на Юникоде

Python 3 полностью реализован на Юникоде, а точнее на UTF-8. Вот что это означает:

По умолчанию предполагается, что исходный код Python 3 написан с помощью UTF-8. Это значит, что вам не нужно использовать определение

```
# -*- coding: UTF-8 -*
```

- в начале файлов .py в этой версии языка.

Пример 6 Коды символов
"""
Функция ord() позволяет получить номер символа по таблице Unicode. Соответственно, принимает она в качестве аргумента одиночный символ, заключенный в кавычки:
"""

```
print("ord(a)=",ord("a"))#ord(a)= 97
print("ord(A)=",ord("A"))#ord(A)= 65
print("ord(z)=",ord("z"))#ord(z)= 122
print("ord(ф)=",ord("ф"))#ord(ф)= 1092
print("ord(Ф)=",ord("Ф"))#ord(Ф)= 1060
print("ord(в)=",ord("в"))#ord(в)= 1074
print("ord(х)=",ord("х"))#ord(х)= 1093
```

#Пример 7. Вывод таблицы символов

```
for i in range(32, 128):
    print("i=",i,"<->", chr(i), end=' ')

    if (i - 1) % 10 == 0:
        print()

print()
```

i= 32 <-> i= 33 <-> ! i= 34 <-> " i= 35 <-> # i= 36 <-> $ i= 37 <-> % i= 38 <-> & i= 39 <-> ' i= 40 <-> (i= 41 <->)
i= 42 <-> * i= 43 <-> + i= 44 <-> , i= 45 <-> - i= 46 <-> . i= 47 <-> / i= 48 <-> 0 i= 49 <-> 1 i= 50 <-> 2 i= 51 <-> 3
i= 52 <-> 4 i= 53 <-> 5 i= 54 <-> 6 i= 55 <-> 7 i= 56 <-> 8 i= 57 <-> 9 i= 58 <-> : i= 59 <-> ; i= 60 <-> < i= 61 <-> =
i= 62 <-> > i= 63 <-> ? i= 64 <-> @ i= 65 <-> A i= 66 <-> B i= 67 <-> C i= 68 <-> D i= 69 <-> E i= 70 <-> F i= 71 <-> G
i= 72 <-> H i= 73 <-> I i= 74 <-> J i= 75 <-> K i= 76 <-> L i= 77 <-> M i= 78 <-> N i= 79 <-> O i= 80 <-> P i= 81 <-> Q
i= 82 <-> R i= 83 <-> S i= 84 <-> T i= 85 <-> U i= 86 <-> V i= 87 <-> W i= 88 <-> X i= 89 <-> Y i= 90 <-> Z i= 91 <-> [
i= 92 <-> \ i= 93 <->] i= 94 <-> ^ i= 95 <-> _ i= 96 <-> ` i= 97 <-> a i= 98 <-> b i= 99 <-> c i= 100 <-> d i= 101 <-> e
i= 102 <-> f i= 103 <-> g i= 104 <-> h i= 105 <-> i i= 106 <-> j i= 107 <-> k i= 108 <-> l i= 109 <-> m i= 110 <-> n i= 111 <-> o
i= 112 <-> p i= 113 <-> q i= 114 <-> r i= 115 <-> s i= 116 <-> t i= 117 <-> u i= 118 <-> v i= 119 <-> w i= 120 <-> x i= 121 <-> y
i= 122 <-> z i= 123 <-> { i= 124 <-> | i= 125 <-> } i= 126 <-> ~ i= 127 <->

Функция chr()
позволяет получить символ по его номеру

```
"""
print(87, chr(87)) #87 W
```

```
#Посчитать количество строчных (маленьких)
# и прописных (больших) букв в введенной строке.
# Учитывать только английские буквы.
s = input("Введите строку на английском ")
let_s = 0
let_b = 0
for i in s:
    if 'a' <= i <= 'z':
        let_s += 1
    else:
        if 'A' <= i <= 'Z':
            let_b += 1
print(let_s)
print(let_b)
```

#8 Сравнение строк

```
print('AAA'=='AAA') #True
print('abc'>'r',ord('a'),ord('r'))#False 97 114

print(ord('A'), ord('a'))#65 97
print('ASDFG'<'dfghjj')#True
print('abc'<'abcd')#True
```

```
# Пример 9
"""
«паровоз» будет «меньше»,
чем слово «пароход»:
они отличаются в пятой букве и «в» < «х».
"""
s1 = "паровоз"
s2 = "пароход"
if s1 < s2:
    print( s1, "<", s2 )
elif s1 == s2:
    print( s1, "=", s2 )
else:
    print( s1, ">", s2 )
```

Стандартные функции

Верхний/нижний регистр:

```
s = "aAbBcC"
s1 = s.upper()   # "AABBCC"
s2 = s.lower()   # "aabbcc"
```

Проверка на цифры:

```
s = "abc"
print( s.isdigit() )   # False
s1 = "123"
print( s1.isdigit() )   # True
```

… и много других.

#10 строки метод upper()

```
s='qwerty'
print(s,id(s))#qwerty 69954304
print(s.upper(),id(s))#QWERTY 69954304
print(s,id(s))#qwerty 69954304
s=s.upper()#
print(s,id(s))#QWERTY 10875552
```

#11 строки метод lower()

```
z='ZXCV'
print(z,id(z))#ZXCV 60386016
z=z.lower()
print(z,id(z))#zxcv 50655872
```

#12 строки метод count()

```
s='hello worolodo'
print(s.count('o'))#5
#print(s.count())#TypeError: count() takes at least 1 argument (0 given)
print(s.count('o',5))#4
print(s.count('o',3,10))#3
print(s.count('o',3,12))#4
```

String.count(sub[, start[, end]])

возвращает число повторений подстроки sub в строке String.
Два необязательных аргумента:

start – индекс, с которого начинается поиск;
end – индекс, которым заканчивается поиск.

13 строки метод find() rfind()

```
s='hello worolodo'
print(s.find('o'))#4
print(s.find('wo'))#6
print(s.find('k'))#-1
print(s.rfind('o'))#13
```

String.find(sub[, start[, end]])

возвращает индекс первого найденного вхождения подстроки sub в строке String.
А аргументы start и end работают также как и в методе count.

Если требуется делать поиск в обратном направлении: справа-налево, то для этого используется метод
String.rfind(sub[, start[, end]])

```
# 14 строки метод index()
s='hello worolodo'
print(s.index('o'))#4
#print(s.index('k'))#
'''
    print(s.index('k'))#
ValueError: substring not found
'''
```

String.index(sub[, start[, end]])

Он работает абсолютно также как find, но с одним отличием: если указанная подстрока sub не находится в строке String, то метод приводит к ошибке
В действительности такие ошибки можно обрабатывать как исключения и это бывает полезно для сохранения архитектуры программы: когда неожиданные ситуации обрабатываются единым образом в блоке исключений.

```
# 15 строки метод replace()
s='hello worolodo'
print(s,id(s))#hello worolodo 30886512
print(s.replace('o','OO'), id(s) )#hellOO wOOrOOlOOdOO 30886512
```

String.replace(old, new, count=-1)

Выполняет замену подстрок old на строку new и возвращает измененную строку
Третий необязательный аргумент задает максимальное количество замен. Например:

msg.replace("a", 'o', 2)
Заменит только первые две буквы a: «msg.replace("a", 'o', 2)». При значении -1 количество замен неограниченно.

```
# 16 строки метод isalpha()
s='hello world'
print(s.isalpha())#False
#False, т.к. имеется символ пробела

s='helloworld'
print(s.isalpha())#True
# вернет True, т.к. наша строка содержит
# только буквенные символы
```

String.isalpha()

возвращает True, если строка целиком состоит из букв и False в противном случае.

```
# 18 строки метод isdigit ()
s='12345'
print(s.isdigit())#True
print("5.6".isdigit())#False
#т.к. имеется символ точки
print("56".isdigit())#True
```

String.isdigit()

возвращает True, если строка целиком состоит из цифр и False в противном случа

```
# 19 строки метод isdigit ()
s='12345'
print(type(s),s.isdigit())#<class 'str'> True
s=int(s)
print(type(s),s)#<class 'int'> 12345
s='12 345'
print(s.isdigit())#False
# s=int(s) #ValueError: invalid literal for int() with base 10: '12 345'
```

```
# 20 Такая проверка полезна, например,
# перед преобразованием строки
# в целое число:
dig = input("Введите число: ")
if(dig.isdigit()):
    dig = int(dig)
    print(dig)
else:
   print("Число введено неверно")
```

```
# 21 Контроль ввода метод isdigit ()
# подсчет суммы целых чисел
sum=0
while True:
    print('Введите число или Enter для выхода из программы')
    x=input('x=')
    if x=='':
        print('Good Bye')
        break
    if x.isdigit():
        x=int(x)
        sum+=x
    else:
        print('это не число')
print("sum=",sum)
```

#22 Мы здесь разбиваем строку по пробелам

split()

```
fio="Иванов Иван Иванович"
print(type(fio),fio,id(fio))
#<class 'str'> Иванов Иван Иванович 53036080
z=fio.split()
print(z)#['Иванов', 'Иван', 'Иванович']
print(type(z),z,id(z))
#<class 'list'> ['Иванов', 'Иван', 'Иванович'] 51568360
print("F=",z[0])#F= Иванов
print("I=",z[1])#I= Иван
print("O=",z[2])#O= Иванович
```

```
# 23 split()
w='ivanov ivan ivanovich'
v=w.split('n')
print(type(v),v)
#<class 'list'> ['iva', 'ov iva', ' iva', 'ovich']
```

```
# 24 split()
z='23,45,4,555,66,433,23,56,45'
print(type(z),z)
#<class 'str'> 23,45,4,555,66,433,23,56,45
y=z.split(',')
print(type(y),y)
#['23', '45', '4', '555', '66', '433', '23', '56', '45']
```

```
## 25 join
# Метод join () является строковым методом и возвращает строку,
# в которой элементы последовательности были объединены
# разделителем str.
y=['23', '45', '4', '555', '66', '433', '23', '56', '45']
print(type(y),y)
#<class 'list'> ['23', '45', '4', '555', '66', '433', '23', '56', '45']

s=''.join(y)
print(type(s),s)#
#<class 'str'> 234545556643323 5645
s='!'.join(y)
print(type(s),s)#
#<class 'str'> 23!45!4!555!66!433!23!56!45
```

```
#26 split() Ввод списка целых чисел через пробел
A = input('Введите целые числа через пробел. Конец ввода- Enter: ').split()
print()
print(type(A),A)
#<class 'list'> ['1', '2', '3']
```