PYTHON
Функции часть 1
(Лекция 8)

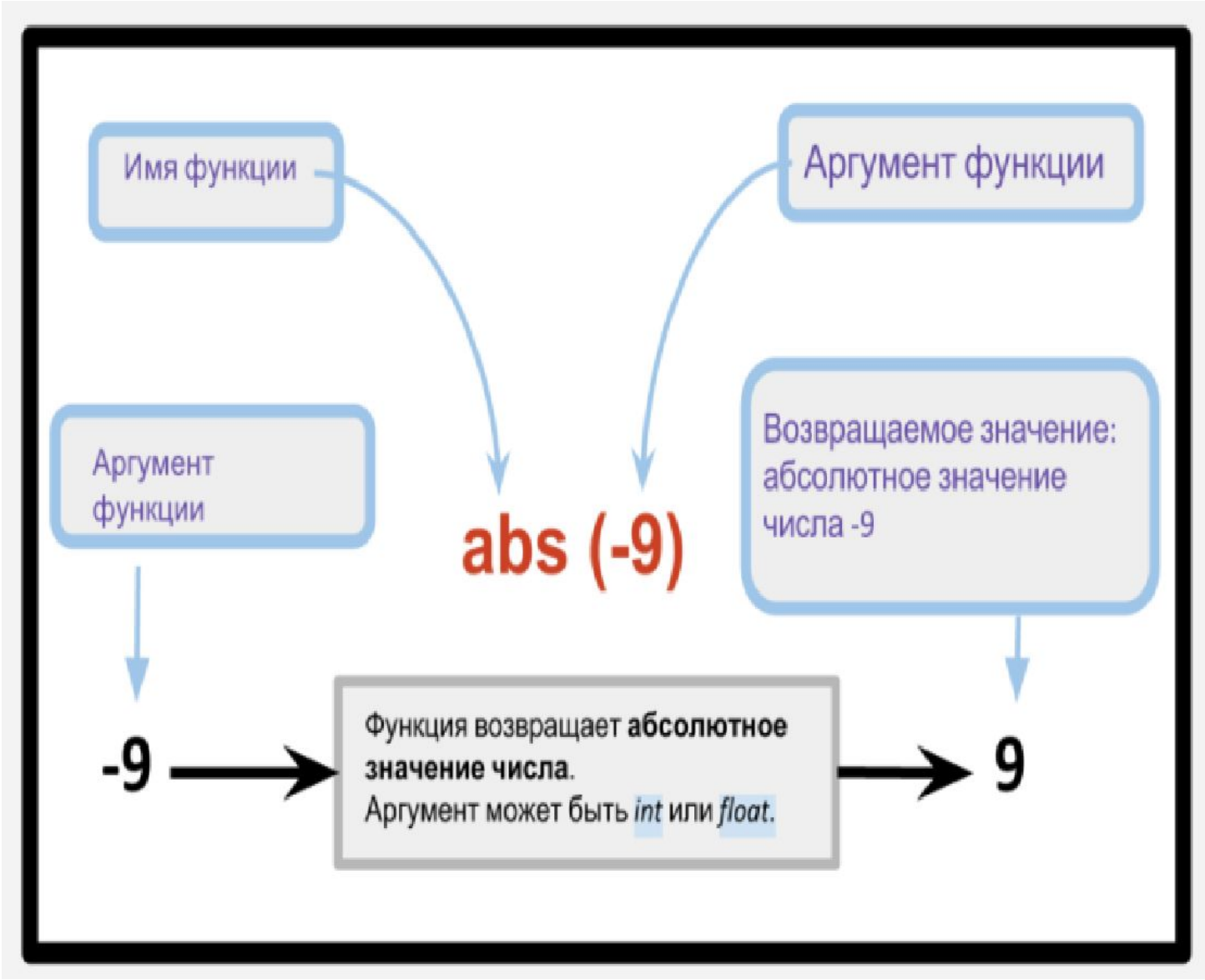
Что такое функция

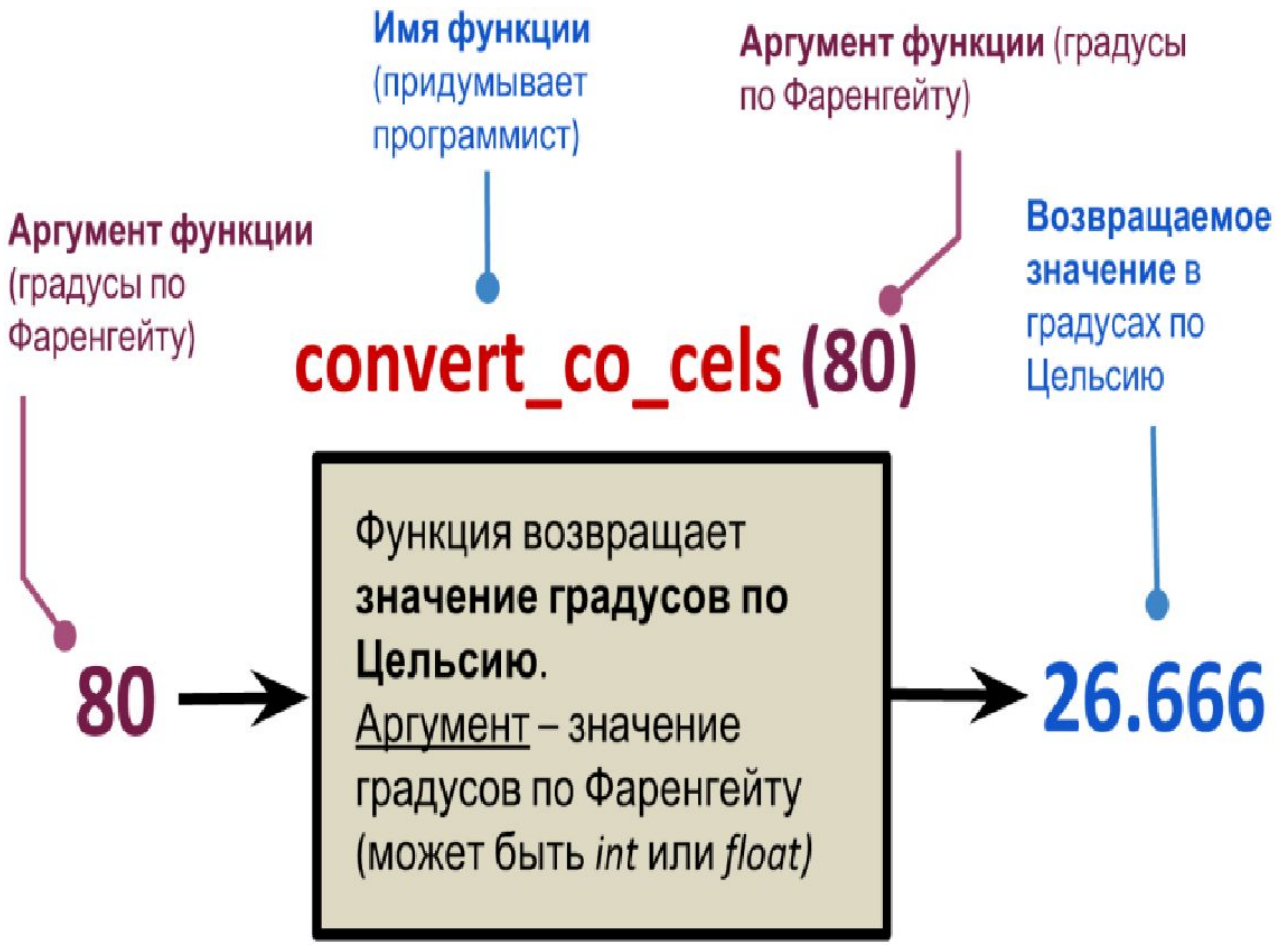
Функция — это блок кода, который можно многократно вызывать на выполнение. Она является фундаментальной частью любого языка программирования.

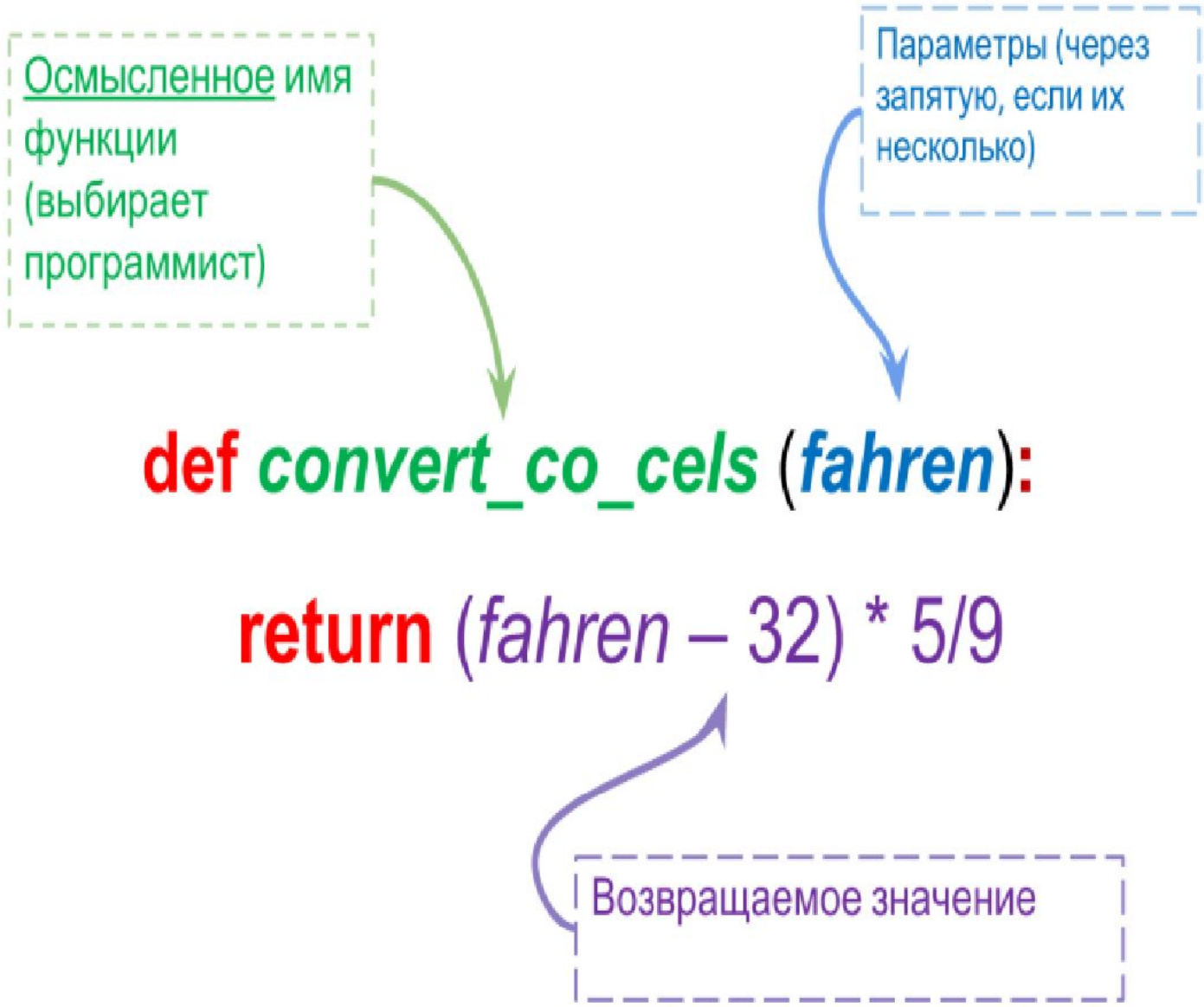
Функция позволяет разделять программу на самостоятельные, но связанные части. Программисты используют функции, чтобы сделать программу модульной и избежать повторения кода.

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

```
# 1 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()
```

Режимы сопоставления аргументов в Python. Классификация

1. Сопоставление по позиции: Fn(value). Пример

Это есть классический способ передачи аргумента. При этом режиме значения, которые передаются, присваиваются именам в функции в порядке слева направо.

```
# Аргументы. Режимы сопоставления.
# Передача аргумента в функцию - сопоставление по позиции

# Функция, получающая 4 аргумента
def Fn(a, b, c, d):
    # в теле функции выводятся значения аргументов с именами
    print('Fn.a = ', a)
    print('Fn.b = ', b)
    print('Fn.c = ', c)
    print('Fn.d = ', d)

# Передача объектов в функцию Fn()
Fn(1, 2, 3, 4) # Fn.a=1, Fn.b=2, Fn.c=3, Fn.d=4
```

#2 Использование аргумента

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


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

```
# 3Площадь 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)
```

```
#4
def arithmetic (a, b, c):
    if c == "+":
        return a + b
    elif c == "-":
        return a - b
    elif c == "*":
        return a * b
    elif c == "/":
        return a / b
    else:
        return "Неизвестная операция!"

result = arithmetic (int(input("Введите число a=")),
                     int(input("Введите число b=")),
                     input("Введите операцию:"))
print("Результат: ", result)
```

2. Сопоставление по именам – это режим, в котором при вызове функции аргументы (объекты) содержат имена, которые описываются параметрами в теле функции (в инструкции def). Имя параметра в инструкции def получает значение, которое присваивается такому же имени в вызове функции.

```
# Аргументы. Режимы сопоставления.
# Передача аргумента в функцию - сопоставление за именем

# Функция, получающая 4 аргумента
def Fn(a, b, c, d):
    # в теле функции выводятся значения аргументов с именами
    print('Fn.a = ', a)
    print('Fn.b = ', b)
    print('Fn.c = ', c)
    print('Fn.d = ', d)

# Передача объектов в функцию Fn() - задаются конкретные имена
Fn(d=4, a=1, c=3, b=2) # Fn.a=1, Fn.b=2, Fn.c=3, Fn.d=4
```

Если при вызове функции Fn(), указать несуществующий параметр, то интерпретатор выдаст ошибку. Например, если в вызове функции фрагмент c=3 заменить на f=3

Fn(d=4, a=1, f=3, b=2)
то интерпретатор выдаст следующую ошибку

TypeError: Fn() got an unexpected keyword argument 'f'

3. Режим передачи аргументов по умолчанию: def Fn(name=value).

В этом режиме, при объявлении функции (в инструкции def) допускается устанавливать параметрам значение по умолчанию. Это значит, что если параметр со значением по умолчанию не получит значения в вызывающем коде, то этому параметру будет установлено именно это значение по умолчанию.

```
# Аргументы. Режимы сопоставления.
# Передача аргумента в функцию - указание значений по умолчанию

# Функция, которая получает 4 параметра.
# Параметры c, d получают значение по умолчанию, соответственно 10 и 100
def Fn(a, b, c=10, d=100):
    # в теле функции выводятся значения параметров с именами
    print('Fn.a = ', a)
    print('Fn.b = ', b)
    print('Fn.c = ', c)
    print('Fn.d = ', d)
# Передача объектов в функцию Fn() - задаются значения по умолчанию
Fn(1,2,3,4) # Fn.a=1, Fn.b=2, Fn.c=3, Fn.d=4
Fn(1,2,3) # Fn.a=1, Fn.b=2, Fn.c=3, Fn.d=100
Fn(1,2) # Fn.a=1, Fn.b=2, Fn.c=10, Fn.d=100
```

4. **Режим передачи аргументов по умолчанию. Порядок следования аргументов по умолчанию в инструкции def**

Если в описании функции в инструкции def именам аргументов присваиваются значения по умолчанию, то в этом случае нужно соблюдать определенный порядок, который определяется следующим правилом:

первыми определяются имена аргументов, которые не получают значения по умолчанию. И уже после этого определяются имена аргументов, которые получают значения по умолчанию.
Например. Пусть задана функция Fn(), которая получает три аргумента a, b, c. Если при описании функции, для аргументов a, b задать значения по умолчанию следующим образом

```
def Fn(a=1, b=2, c): # это есть ошибочный код
    # ...
    # ...
    return
```

то при запуске программы на выполнение, интерпретатор Python выдаст синтаксическую ошибку со следующим сообщением

non default argument follows default argument

что означает «аргумент не по умолчанию следует за аргументом по умолчанию». Итак, аргументы по умолчанию должны определяться последними в перечне аргументов функции в инструкции def.

Чтобы поправить ситуацию, можно написать следующий код объявления функции Fn()

```
def Fn(c, a=1, b=2): # это есть корректный код
    # тело функции Fn()
    # ...
```

В вышеприведенном объявлении функции Fn(), имена аргументов по умолчанию **a**, **b** следуют после аргумента не по умолчанию **c**.

```
# Печать списков в столбик
# Функция. Печать списка группы - передаем список
#Вариант 1 for
spis_gr_1=["Иванов","Петров","Сидоров","Аверин","Куценко"]
spis_gr_2=["Пугачева","Киркоров","Маменко","Булитко"]

def print_spisok(lst):
    n=0
    for st in lst:
        print(n,st)
        n+=1

print_spisok(spis_gr_1)
```

```
#Варианты вызова функции
def f(a,b,c):
    print(a,b,c)

#1. позиционный
#f(1,2,3)
#f(1,2) #TypeError: f() missing 1 required positional argument: 'c'
#f(1,2,3,4) #TypeError: f() takes 3 positional arguments but 4 were given

#2. по имени
#f(b=10,c=45,a=98)
#f(b=10,c=45,d=98)#TypeError: f() got an unexpected keyword argument 'd'

#3. Комбинированный вариант
#f(10,c=45,b=98)#10 98 45 # начала идут позиционные аргументы
#f(c=45,b=9,10)
```

```
# Использование параметров по умолчанию
def f(a,b='ASDF',c='ghjk'):
    print(a,b,c)

f(56)#56 ASDF ghjk
f(44,66,88)#44 66 88
```

```
def g(a=2,b='ASDF',c='ghjk'):
    print(a,b,c)
g()#2 ASDF ghjk
g(1,2,3)#1 2 3 #значения передаютя позиционно
g(b=99999)#2 99999 ghjk
```

######## так нельзя. Обязательные не м.б. в середине

```
'''
def r(a=45,b,c='tghj'):
    print(a,b,c)
r(1,2,3)
'''
```

9. Тренажер работы со списками

```
#Тренажер работы со списками
spis_gr=["Иванов","Петров","Сидоров","Аверин",
        "Куценко","Пугачева",
        "Киркоров","Маменко","Булитко"]
ist_0001=["Иванов","Петров"]
def print_spisok(lst):
    i=0
    while i < len(lst):
        print(i+1, lst[i])
        i += 1

menu='''
МЕНЮ ПРОГРАММЫ "Список Группы":
1 - вывести на печать текущий список в строку;
2 - вывод списка в столбик
3- добавить в список;
4 - отсортировать список по возрастанию;
5 - отвортировать список по убыванию;
6 - число элементов в списке;
Введите число или пробел - для выхода из программы:
'''
```

```
def work_with_list(lst):
    print(menu)
    while True:
        z = input('Введите режим: ')
        if z=='1':
            print(lst)
        elif z=='2':
            print_spisok(lst)
        elif z=='3':
            name= input("Введите фамилию: ")
            lst.append(name)
        elif z=='4':
            lst.sort()
        elif z=='5':
            lst.reverse()
        elif z=='6':
            print(len(lst))
        elif z=='':
            break
        else:
            print("Нет такой цифры")
    print("Работа программы закончена!")

work_with_list(spis_gr)
```