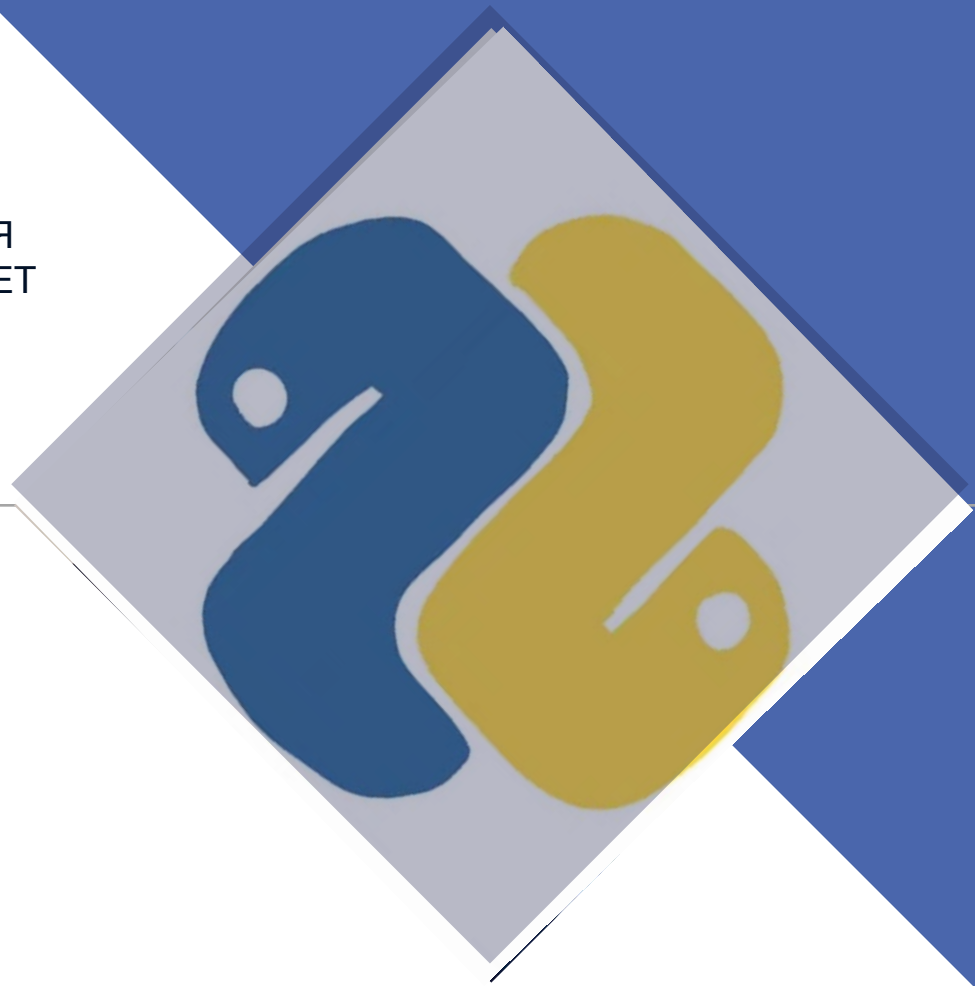
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КАБАРДИНО-БАЛКАРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. Х.М. БЕРБЕКОВА»

ИНСТИТУТ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И ЦИФРОВЫХ
ТЕХНОЛОГИЙ



Практическое занятие 2.1

ФУНКЦИИ, ОБЪЯВЛЕНИЕ И ВЫЗОВ

ПОНЯТИЕ ФУНКЦИИ

Функция — это фрагмент программного кода, который решает какую-либо задачу, объект, принимающий аргументы и возвращающий значение.

Она является фундаментальной частью любого языка программирования. 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.

Его можно вызывать в любом месте основной программы. При вызове происходит выполнение команд тела функции.

Функции в Python определяются с помощью ключевого слова def:

```
abs()
round ()
len()
int()
float()

print()
input()
sum()
max()
min()
…
```

```
def < имя функции>([список параметров]):
        оператор 1
        оператор 2
         …
```

Параметр — это переменная, которой будет присваиваться входящее в функцию значение
Аргумент — само это значение, которое передается в функцию при её вызове.
Параметры функции и аргументы функции необходимо указывать через запятую (может быть сколько угодно).

Именованные аргументы. Фактические и формальные параметры

Функции можно записывать в одну строку

Пример

```
def sum(x, y): print(x + y)
sum(5, 6)
```

В функции можно использовать неограниченное количество параметров, но число аргументов должно точно соответствовать параметрам. Эти параметры представляют собой позиционные аргументы.

Пример

```
def drawBox(a=2, b=3):
    c= a + b
    print(c)

drawBox(5,7)
```

Именованные аргументы. Фактические и формальные параметры

В Python есть возможность задать для аргументов значение по умолчанию. Если значение для такого аргумента при вызове функции не передаётся, то используется значение по умолчанию.

Часто в функции передаётся большое количество аргументов и вспомнить порядок их перечисления в функции бывает сложно. В Python есть возможность передать значение аргумента по его имени. В таком случае аргумент становится уже не позиционным, а именованным. Именованному аргументу присваивается значение при вызове функции.

Пример

```
def final_price(price, discount=1):
    return price - price * discount / 100

print(final_price(1000, discount=5))
print(final_price(discount=10, price=1000))
```

Вывод программы:

```
950.0
900.0
```

Именованные аргументы.
Фактические и формальные параметры

В Python есть возможность передать в функцию неограниченное количество позиционных и именованных аргументов.

Рассмотрим первый случай: функция принимает любое количество позиционных аргументов. Такой функцией, например, является стандартная функция print(), так как она может принимать любое количество выводимых строк-аргументов. Чтобы указать, что функция может принимать неограниченное количество позиционных аргументов, нужно при её объявлении поставить параметр со *. К примеру, *args. В функции этот аргумент будет кортежем, содержащим переданные значения позиционных аргументов. Модифицируем нашу функцию из примера про скидки так, чтобы мы могли передать в неё любое количество цен, а вернуть список цен со скидкой:

```
def final_price(*prices, discount=1):
    return [price - price * discount / 100 for price in prices]

print(final_price(100, 200, 300, discount=5))
```

Вывод программы: [95.0, 190.0, 285.0]

Именованные аргументы.
Фактические и формальные параметры

Добавим в нашу функцию возможность принимать границы диапазона цен, для которых рассчитывается скидка. Вернём список только рассчитанных со скидкой цен. Границы диапазона будут передаваться в аргументах price_low и price_high. Если нижняя или правая границы не переданы, то используем минимальную и максимальную стоимости из позиционных аргументов:

```
def final_price(*prices, discount=1, **kwargs):
    low = kwargs.get("price_low", min(prices))
    high = kwargs.get("price_high", max(prices))
    return [price - price * discount / 100 for price in prices if low <= price <= high]

print(final_price(100,     200, 300, 400, 500, discount=5))
print(final_price(100,  200,    300, 400, 500, discount=5, price_low=200))
print(final_price(100,     200, 300, 400, 500, discount=5,price_high=200))
print(final_price(100,     200, 300, 400, 500, discount=5, price_low=200, price_high=350))
```

Вывод программы:

```
[95.0, 190.0, 285.0, 380.0, 475.0]
[190.0, 285.0, 380.0, 475.0]
[95.0, 190.0]
[190.0, 285.0]
```

Именованные аргументы.
Фактические и формальные параметры

Чтобы функция могла принимать неограниченное количество именованных аргументов, нужно при её объявлении поставить параметр с **. Например, **kwargs (сокращение от "keyword arguments").

В Python аргументом функции может быть другая функция.

Рассмотрим пару полезных встроенных функций высшего порядка. Функция filter() позволяет отобрать элементы из итерируемого объекта по некоторому критерию. Результат работы функции — итератор. Критерий отбора — функция, возвращающая значения логического типа. Функция filter() последовательно проверяет значение функции-критерия для каждого элемента и при значении True элемент попадает в выходной итератор. Напишем программу, которая выберет из списка целых чисел только положительные:

Пример

```
def only_pos(x): return x > 0

result = filter(only_pos, [-1, 5, 6, -10, 0]) print(", ".join(str(x) for x in result))
```

Пример 1

```
a=abs(-7) #Встроенные функции
print(a)
7
b=max(4,5,7,4,3,2)
print(b)
7
b=max(4,abs(-90),5,7,4,min(100,200),3,2)
print(b)
100
```

Пример 2

```
def square(x): #Функция возводит число в квадрат
    print(x**2)
a= square(6)
print(a)
36
None
```

Пример 3

```
def square(x):
    print(x**2)
    return None
36
```

Пример 5

```
def square(x):
    return x**2
def example():
    print(1)
    print(2)
    return ‘hello’
    print(3)
    print(4)
example()
```

1
2

Пример 6

```
def example():
    return 1
    return 2
    return 3
print(example())
```

1

Пример 7

```
def sqr(x):
    return(x*x)
def print_sqr(a):
    print(“sqr=“,a)
y=5
print_sqr(sqr(y))
```

sqr=25

Пример 8

```
def ss(x,y):
    if x>y:
        return x,y
    else:
        return y,x
print(ss(3,8))
```

(8,3)

Пример 9

```
def even(x):
        return  x%2==0
for i in range (1,6):
    print(i,even(i))
```

Пример 10

```
def
sqAndPer(a,b):#нахожде
ние площади и
периметра
    return a*b,2*(a+b)
square,perimeter =
sqAndPer(2,5)
print(square,perimeter)
```

10 14

Пример 11

```
def
sqAndPer(a,b):#нахожде
ние площади и
периметра
    return a*b,2*(a+b)
print(sqAndPer(3,6))
```

(18,18)

Задания для самостоятельного решения

1. **Задайте функцию, которая не принимает никаких аргументов и просто выводит на экран строку:**

It's my first function

В конце программы вызовите эту функцию.

2. Запишите функцию без аргументов, которая считывает с клавиатуры имя и фамилию, записанные в одну строку через пробел, и выводит на экран сообщение (без кавычек):

" Уважаемый, <имя> <фамилия>! Вы верно выполнили это задание! "

В конце программы вызовите эту функцию.

Формат входных данных	**Формат выходных данных**
Рогов Иван	Уважаемый, Рогов Иван! Вы верно выполнили это задание!

Задания для самостоятельного решения

3. Объявите функцию, которая имеет один параметр - вес предмета и выводит на экран сообщение (без кавычек):

" Предмет имеет вес: X кг. "

где x - переданное значение функции. После объявления функции прочитайте (с помощью функции input) вещественное число и вызовите функцию с этим значением.

Формат входных данных	Формат выходных данных
12.67	Предмет имеет вес: 12.67 кг.

Задания для самостоятельного решения

4. Объявите функцию, которая принимает список, находит максимальное, минимальное и сумму значений этого списка и выводит результат в виде строки:

Min = v_min, max = v_max, sum = v_sum

где v_min, v_max, v_sum - вычисленные значения минимального, максимального и суммы значений списка.

После объявления функции прочитайте (с помощью функции input) список целых чисел, записанных в одну строку через пробел, и вызовите функцию с этим списком.

Формат входных данных	Формат выходных данных
8 11 5 -10 12 0	Min = -10, max = 12, sum = 26

Задания для самостоятельного решения

5. Объявите функцию с двумя параметрами width и height (ширина и высота прямоугольника), которая выводит сообщение :

Периметр прямоугольника, равен x

где x - вычисленный периметр прямоугольника. После объявления функции прочитайте (с помощью функции input) два целых числа, записанных в одну строку через пробел, и вызовите функцию с этими значениями.

Формат входных данных	Формат выходных данных
8 11	Периметр прямоугольника, равен 38

Задания для самостоятельного решения

6. Напишите функцию, которая проверяет корректность переданного ей email-адреса в виде строки. Будем полагать, что адрес верен, если он обязательно содержит символы '@' и '.', а все остальные символы могут принимать значения: 'a-z', 'A-Z', '0-9' и '_'. Если email верен, то функция выводит **ДА**, иначе - **НЕТ**.

После объявления функции прочитайте (с помощью функции input) строку с email-адресом и вызовите функцию с этим аргументом.

Формат входных данных	Формат выходных данных
sc_lib@list.ru	ДА

СПАСИБО ЗА ВНИМАНИЕ!