ШКОЛА ОБЛАЧНЫХ ТЕХНОЛОГИЙ

# Процедуры и функции. Lambda function. *args, **kwargs в Python.

# Функции

Функции представляют блок кода, который выполняет определенную задачу и который можно повторно использовать в других частях программы. В большинстве случаев с функцией связывается идентификатор, но допускаются и безымянные функции.

# Функции

Создание функции

```
def имя_функции(параметры):
    инструкции
```

```
def calc(a, b):
    c = a + b
    print(f"{a} + {b} = {c}")
```

Вызов функции

```
имя_функции(параметры)
```

```
calc(5, 4)
```

```
def main():
    say_hello()  # Вызов функции say_hello
    say_goodbye()  # Вызов функции say_goodbye


def say_hello():
    print("Привет")


def say_goodbye():
    print("Пока!")


# Вызов функции main
main()
```

# Необязательные аргументы

```
def print_person(name, age=18):
    print(f"Имя: {name}, возраст: {age}")


print_person("Bob")
print_person("Tom", 37)
```

# *args, **kwargs

Оператор *

Оператор * чаще всего ассоциируется у людей с операцией умножения, но в Python он имеет и другой смысл. Этот оператор позволяет «распаковывать» объекты, внутри которых хранятся некие элементы. Вот пример:

```
a = [1,2,3]

b = [a,4,5,6]
print(b) # [[1, 2, 3], 4, 5, 6]

b = [*a,4,5,6]
print(b) # [1, 2 ,3, 4, 5, 6]
```

Итак, мы знаем о том, что оператор «звёздочка» в Python способен «вытаскивать» из объектов составляющие их элементы. Знаем мы и о том, что существует два вида параметров функций. *args — это сокращение от «arguments» (аргументы), а **kwargs — сокращение от «keyword arguments» (именованные аргументы).

```
def printScores(student, *scores):
    print(f"Имя студента: {student}")
    for score in scores:
        print(score)


printScores("Михаил", 100, 95, 88, 92, 99)

"""
Имя студента: Михаил
100
95
88
92
99
"""
```

Благодаря ** создаётся словарь, в котором содержатся именованные аргументы, переданные функции при её вызове.

```
def printPetNames(owner, **pets):
    print(f"Имя владельца: {owner}")
    for pet,name in pets.items():
        print(f"{pet}: {name}")


printPetNames("Михаил", dog="Брок", fish=["Ларри", "Дорри", "Мо"], turtle="Шелдон")

"""
Имя владельца: Михаил
dog: Брок
fish: ['Ларри', 'Дорри', 'Мо']
turtle: Шелдон
"""
```

# Возвращение результата

```
def имя_функции(параметры):
    инструкции
    return возвращаемое_значение
```

```
def get_message():
    text = "Привет!"
    return text
```

```
msg = get_message()
print(msg)
# Привет!
```

# Lambda функции

В языке Python представляют небольшие анонимные функции, которые определяются с помощью оператора lambda.

```
lambda параметры: иструкции
```

```
message = lambda: print("hello")

message()  # hello
```

```
square = lambda n: n * n

print(square(4))  # 16
print(square(5))  # 25
```

# Задания для выполнения

1) Создать процедуру hello3. При её вызове в консоль должно три раза печататься «Hello». Вызовите процедуру 2 раза, а затем напечатайте «Bye».

2) Создать процедуру helloUser, принимающий один аргумент – имя пользователя.
При вызове этой процедуры в консоль должно печататься приветствие, включающие в себя переданное имя пользователя. Вызовите функцию два раза с разными именами.

3) Дополнить процедуру из предыдущего задания. Сделать возможность вызова процедуры без указания имени пользователя – в этом случае оно должно заменяться на «Гость». Проверить работоспособность процедуры, вызвав её.

4) Создать процедуру printMax, принимающую в качестве аргументов два числа. При вызове этой процедуры в консоль должно печататься, сообщение с сравнением чисел. Вызовите функцию три раза, чтобы получить разные результаты.

# Домашняя работа

- Доделать задания которые не успели сделать во время лекции
- Повторить пройденный материал