Лабораторная работа 2.10 Функции с переменным числом параметров в Python

Цель работы: *приобретение навыков по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x.*

Ход работы

Как уже обсуждалось ранее, по умолчанию параметры передаются функции через стек. Поэтому, технически, нет ограничения на количество передаваемых параметров. Проблема в том, как потом функция будет разбирать переданные параметры. Функции с переменным числом параметров объявляются как обычные функции, с использованием конструкций `*args` и `**kwargs`.

Позиционные и именованные аргументы

Для того чтобы разобраться с `*args` и `**kwargs`, нам нужно освоить концепции позиционных (positional) и именованных (keyword) аргументов.

Сначала поговорим о том, чем они отличаются. В простейшей функции мы просто сопоставляем позиции аргументов и параметров. Аргумент №1 соответствует параметру №1, аргумент №2 — параметру №2 и так далее.

```
def print_these(a, b, c):
   print(a, "is stored in a")
   print(b, "is stored in b")
   print(c, "is stored in c")


print_these(1, 2, 3)
```

```
1 is stored in a
2 is stored in b
3 is stored in c
```

Для вызова функции необходимы все три аргумента. Если пропустить хотя бы один из них — будет выдано сообщение об ошибке.

```
def print_these(a, b, c):
   print(a, "is stored in a")
   print(b, "is stored in b")
   print(c, "is stored in c")


print_these(1, 2)
```

```
TypeError: printThese() missing 1 required positional argument: 'c'
```

Если при объявлении функции назначить параметру значение по умолчанию — указывать соответствующий аргумент при вызове функции уже необязательно. Параметр становится опциональным.

```
def print_these(a, b, c=None):
   print(a, "is stored in a")
   print(b, "is stored in b")
   print(c, "is stored in c")


printThese(1, 2)
```

```
1 is stored in a
2 is stored in b
None is stored in c
```

Опциональные параметры, кроме того, можно задавать при вызове функции, используя их имена.

В следующем примере установим три параметра в значение по умолчанию `None` и взглянем на то, как их можно назначать, используя их имена и не обращая внимания на порядок следования аргументов, применяемых при вызове функции.

```
def print_these(a=None, b=None, c=None):
   print(a, "is stored in a")
   print(b, "is stored in b")
   print(c, "is stored in c")


print_these(c=3, a=1)
```

```
1 is stored in a
None is stored in b
3 is stored in c
```

Оператор «звёздочка»

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]
```

Тут берётся содержимое списка `a`, распаковывается, и помещается в список `b`.

Как пользоваться `*args` и `**kwargs`?

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. А именно, `*args` — это сокращение от «arguments» (аргументы), а `**kwargs` — сокращение от «keyword arguments» (именованные аргументы).

Каждая из этих конструкций используется для распаковки аргументов соответствующего типа, позволяя вызывать функции со списком аргументов переменной длины. Например — создадим функцию, которая умеет выводить результаты, набранные учеником в тесте:

```
def print_scores(student, *scores):
   print(f"Student Name: {student}")
   for score in scores:
      print(score)


print_scores("Jonathan", 100, 95, 88, 92, 99)
```

```
Student Name: Jonathan
100
95
88
92
99
```

В предыдущем примере не использовалась конструкция `*args`. Вместо неё — `*scores`. Нет ли тут ошибки? Ошибки здесь нет. Дело в том, что «args» — это всего лишь набор символов, которым принято обозначать аргументы. Самое главное тут — это оператор `*`. А то, что именно идёт после него, особой роли не играет. Благодаря использованию `*` мы создали список позиционных аргументов на основе того, что было передано функции при вызове.

После того, как мы разобрались с `*args`, с пониманием `**kwargs` проблем быть уже не должно. Имя, опять же, значения не имеет. Главное — это два символа `**`. Благодаря им создаётся словарь, в котором содержатся именованные аргументы, переданные функции при её вызове.

```
def print_pet_names(owner, **pets):
   print(f"Owner Name: {owner}")
   for pet,name in pets.items():
      print(f"{pet}: {name}")


print_pet_names(
    "Jonathan",
    dog="Brock", fish=["Larry", "Curly", "Moe"],
    turtle="Shelldon"
)
```

```
Owner Name: Jonathan
dog: Brock
fish: ['Larry', 'Curly', 'Moe']
turtle: Shelldon
```

Пример 1. *Разработать функцию для определения медианы значений аргументов функции. Если функции передается пустой список аргументов, то она должна возвращать значение* `None`.

Медианой (серединой) набора чисел называется число стоящее посередине упорядоченного по возрастанию ряда чисел. Если количество чисел в ряду чётное, то медианой ряда является полусумма двух стоящих посередине чисел. Применяется в математической статистике — число, характеризующее выборку (например, набор чисел), также используется для вычисления медианной зарплаты.

Решение: Напишем программу для решения поставленной задачи.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

def median(*args):
    if args:
        values = [float(arg) for arg in args]
        values.sort()

        n = len(values)
        idx = n // 2
        if n % 2:
            return values[idx]
        else:
            return (values[idx - 1] + values[idx]) / 2

    else:
        return None


if __name__ == "__main__":
    print(median())
    print(median(3, 7, 1, 6, 9))
    print(median(1, 5, 8, 4, 3, 9))
```

Результаты выполнения программы:

```
None
6.0
4.5
```

Аппаратура и материалы

1. Компьютерный класс общего назначения с конфигурацией ПК не хуже рекомендованной для ОС Windows 10 с подключением к глобальной сети Интернет.
2. Операционная система Windows 10.
3. Система контроля версий Git.
4. Браузер для доступа к web-сервису GitHub, рекомендован к использованию Google Chrome.
5. Дистрибутив языка программирования Python, включающий набор популярных библиотек Anaconda.
6. Интегрированная среда разработки PyCharm Community Edition.

Указания по технике безопасности

При работе на ЭВМ без разрешения руководителя занятия запрещается:

- подавать (снимать) напряжение на ПЭВМ и электрические розетки с распределительного щита;
- включать и выключать блоки питания ПЭВМ и мониторы;
- извлекать ПЭВМ из защитного кожуха;
- устранять неисправности, возникшие в ходе выполнения лабораторной работы.

Методика и порядок выполнения работы

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл `.gitignore` необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.
6. Создайте проект PyCharm в папке репозитория.
7. Проработать примеры лабораторной работы.
8. Решить поставленную задачу: написать функцию, вычисляющую среднее геометрическое своих аргументов $a_1, a_2, \ldots a_n$

$$G = \sqrt[n]{\prod_{k=1}^{n} a_k}. \tag{1}$$

Если функции передается пустой список аргументов, то она должна возвращать значение `None`.

9. Решить поставленную задачу: написать функцию, вычисляющую среднее гармоническое своих аргументов $a_1, a_2, \ldots, a_n$

$$\frac{n}{H} = \sum_{k=1}^{n} \frac{1}{a_k}. \tag{2}$$

Если функции передается пустой список аргументов, то она должна возвращать значение `None`.

10. Приведите в отчете скриншоты результатов выполнения примера при различных исходных данных вводимых с клавиатуры.
11. Зафиксируйте изменения в репозитории.
12. Решите индивидуальное задание согласно своего варианта.
13. Самостоятельно подберите или придумайте задачу с переменным числом именованных аргументов. Приведите решение этой задачи.
14. Зафиксируйте изменения в репозитории.
15. Добавьте отчет по лабораторной работе в *формате PDF* в папку *doc* репозитория. Зафиксируйте изменения.
16. Выполните слияние ветки для разработки с веткой *master*/*main*.
17. Отправьте сделанные изменения на сервер GitHub.
18. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Индивидуальное задание

Напишите функцию, принимающую произвольное количество аргументов, и возвращающую требуемое значение. Если функции передается пустой список аргументов, то она должна возвращать значение `None`. Номер варианта определяется по согласованию с преподавателем. В процессе решения не использовать преобразования конструкции `*args` в список или иную структуру данных.

1. Произведение аргументов, расположенных между максимальным и минимальным аргументами.
2. Сумму аргументов, расположенных между первым и последним нулевыми аргументами.
3. Сумму аргументов, расположенных между первым и последним нулевыми аргументами.
4. Сумму аргументов, расположенных между первым и последним отрицательными аргументами.
5. Сумму аргументов, расположенных до последнего положительного аргумента.
6. Произведение аргументов, расположенных между первым и вторым нулевыми аргументами.
7. Сумму аргументов, расположенных между первым и вторым отрицательными аргументами.
8. Сумму аргументов, расположенных между первым и вторым положительными аргументами.
9. Сумму модулей аргументов, расположенных после первого аргумента, равного нулю.
10. Сумму модулей аргументов, расположенных после первого отрицательного аргумента.
11. Сумму аргументов, расположенных после первого положительного аргумента.
12. Сумму аргументов, расположенных после максимального аргумента.
13. Сумму аргументов, расположенных после минимального аргумента.
14. Произведение аргументов, расположенных после максимального по модулю аргумента.
15. Сумму модулей аргументов, расположенных после минимального по модулю аргумента.
16. Сумму аргументов, расположенных после последнего аргумента, равного нулю.
17. Сумму целых частей аргументов, расположенных после последнего отрицательного аргумента.
18. Сумму положительных аргументов, расположенных до максимального аргумента.

Содержание отчета и его форма

Отчет по лабораторной работе оформляется электронно в формате PDF, должен содержать ответы на контрольные вопросы, ссылку на репозиторий с которым выполнялась работа, скриншоты IDE PyCharm, скриншоты результатов работы программ.

Вопросы для защиты работы

1. Какие аргументы называются позиционными в Python?
2. Какие аргументы называются именованными в Python?
3. Для чего используется оператор `*`?
4. Каково назначение конструкций `*args` и `**kwargs`?