Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра инфокоммуникаций

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.10**

**дисциплины «Программирование на Python»**

**Вариант\_\_\_**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Иващенко Олег Андреевич  2 курс, группа ИВТ-б-о-22-1,  09.03.02 «Информационные и вычислительные машины», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Руководитель практики:  Воронкин Роман Александрович, доцент кафедры инфокоммуникаций  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2023 г.

**Тема**: «Функции с переменным числом параметров в Python»

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

Порядок выполнения работы

Таблица 1 – Код программы example.py

|  |
| --- |
| #!/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)) |



Рисунок 1 – Вывод программы example.py

Таблица 2 – Код программы general\_1.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  def average\_geo(\*args):  """  Функция для рассчёта среднего геометрического значения  введённых чисел. Если числа отсутствуют, то функция  вернёт None  """    if args:  list = [int(arg) for arg in args]  result = 1  for value in list:  result \*= value  return result\*\*(1 / len(list))  else:  return None  if \_\_name\_\_ == "\_\_main\_\_":  arg\_list = input("Введите несколько значений: ").split()  print(f"Результат вывода: {average\_geo(\*arg\_list)}") |



Рисунок 2.1 – Результат вывода программы general\_1.py при вводе значений



Рисунок 2.2 – Результат вывода программы general\_1.py при отсутствии введённых значений

Таблица 3 – Код программы general\_2.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  def average\_harmonic(\*args):  """  Функция вычисления среднего гармонического значения  введённых аргументов. Выводит None, если аргументов нет  """    if args:  harmonic = sum(1 / int(arg) for arg in args)  return len(args) / harmonic  else:  return None  if \_\_name\_\_ == "\_\_main\_\_":  arg\_list = input("Введите несколько значений: ").split()  print(f"Среднее гармоническое: {average\_harmonic(\*arg\_list)}") |



Рисунок 3.1 – Результат выполнения программы general\_2.py при вводе значений



Рисунок 3.2 – Результат выполнения программы general\_2.py при отсутствии ввода значений

Индивидуальное задание. Написать функцию, принимающую произвольное количество аргументов, и возвращающая требуемое значение. Если функции передаётся пустой список аргументов, то она должна возвращать значение None. В процессе решения не использовать преобразования конструкции \*args в список или иную структуру данных.

Сумму аргументов, расположенных после минимального аргумента.

Таблица 4 – Код программы individual.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  def args\_sum(\*args):  """  Функция находит минимальный элемент в ряде значений  и считает сумму элементов, находящихся после минимального.  В случае, если значений нет, функций возвращает None  """    if args:  min\_value = min(args)  min\_index = args.index(min\_value)  result = sum(args[min\_index + 1:])  return min\_index, min\_value, result    else:  return None  if \_\_name\_\_ == "\_\_main\_\_":  arg\_list = tuple(map(int, input("Введите несколько значений: ").split()))  result = args\_sum(\*arg\_list)  if result is None:  print("Ошибка ввода")  else:  print(f"Минимальное значение - [{result[0]}] {result[1]}")  print(f"Результат суммы - {result[2]}") |



Рисунок 4.1 – Результат выполнения программы individual.py при вводе значений



Рисунок 4.2 – Результат выполнения программы individual.py при отсутствии ввода значений

Контрольные вопросы

1. Какие аргументы называются позиционными в Python?

Позиционные аргументы – это аргументы, передаваемые в функцию в порядке их определения в сигнатуре функции. Значения этих аргументов соответствуют позициям, на которых они указаны при вызове функции.

1. Какие аргументы называются именованными в Python?

Именованные аргументы – это аргументы, передаваемые в функцию с явным указанием их имени. При использовании именованных аргументов порядок передачи не важен, поскольку каждый аргумент определяется по его имени. Например:

def example\_function(arg1, arg2):

…

example\_function(arg2 = 5, arg1 = 10)

1. Для чего используется оператор \*?

Оператор «\*» используется для распаковки последовательности (списка или кортежа) в аргументы функции или для определения переменного числа аргументов в сигнатуре функции. Например:

def example\_function(arg1, arg2, arg3):

…

arg\_list = [1, 2, 3]

example\_function(\*arg\_list)

1. Каково назначение конструкций \*args и \*\*kwarg?

«\*args» используется для передачи переменного числа позиционных аргументов в функцию. «args» - это кортеж, который содержит все переданные позиционные аргументы. Пример:

def example\_function(\*args):

….

example\_function(1, 2, 3)

«\*\*kwargs» используется для передачи переменного числа именованных аргументов в функцию. «kwargs» - это словарь, который содержит все переданные именованные аргументы. Пример:

def example\_function(\*\*kwargs):

…

example\_function(arg1 = 1, arg2 = 2, arg3 = 3)

**Выводы**: В процессе выполнения лабораторной работы были приобретены навыки по работе с функциями с переменным числом параметров при написании программ с помощью языка программирования Python версии 3.x, были написаны 4 программы: пример работы с такими функциями, 2 общих задания и индивидуальное задание.