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 №2.3**

**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**: «Работа со строками в языке Python»

**Цель**: Приобретение навыков по работе со строками при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы

Таблица 1 – Код программы example\_1.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  if \_\_name\_\_ == '\_\_main\_\_':  s = input("Введите предложение: ")  r = s.replace(' ', '\_')  print(f"Предложение после замены: {r}") |



Рисунок 1 – Вывод программы example\_1.py

Таблица 2 – Код программы example\_2.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  if \_\_name\_\_ == '\_\_main\_\_':  word = input("Введите слово: ")  idx = len(word) // 2  if len(word) % 2 == 1:  # Длина слова нечетная.  r = word[:idx] + word[idx+1:]  else:  # Длина слова четная.  r = word[:idx-1] + word[idx+1:]  print(r) |



Рисунок 2.1 – Вывод программы example\_2.py при нечётном количестве символов



Рисунок 2.2 – Вывод программы example\_2.py при чётком количестве символов

Таблица 3 – Код программы example\_3.py

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding: utf-8 -\*-  import sys  if \_\_name\_\_ == '\_\_main\_\_':  s = input("Введите предложение: ")  n = int(input("Введите длину: "))  # Проверить требуюемую длину.  if len(s) >= n:  print(  "Заданная длина должна быть больше длины предложения",  file=sys.stderr  )  exit(1)    # Разделить предложение на слова.  words = s.split(' ')  # Проверить количество слов в предложении.  if len(words) < 2:  print(  "Предложение должно содержать несколько слов",  file=sys.stderr  )  exit(1)    # Количество пробелов для добавления.  delta = n  for word in words:  delta -= len(word)    # Количество пробелов на каждое слово.  w, r = delta // (len(words) - 1), delta % (len(words) - 1)  # Сформировать список для хранения слов и пробелов.  lst = []  # Пронумеровать все слова в списке и перебрать их.  for i, word in enumerate(words):  lst.append(word)  # Если слово не является последним, добавить пробелы.  if i < len(words) - 1:  # Определить количество пробелов.  width = w  if r > 0:  width += 1  r -= 1  # Добавить заданное количество пробелов в список.  if width > 0:  lst.append(' ' \* width)    # Вывести новое предложение, объединив все элементы списка lst.  print(''.join(lst)) |



Рисунок 3 – Вывод программы example\_3.py

Индивидуальное задание 1. Вывести «столбиком» все его буквы «и», стоящие на чётных местах.

Таблица 4 – Код программы individual\_1.py

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":  sentence = input("Введите предложение: ")  index = 0  for i in sentence:  if i == 'и' and index % 2 == 0:  print(f"[{index}] {i}")    index += 1 |



Рисунок 4 – Вывод программы individual\_1.py

Индивидуальное задание 2. Дано предложение. Определить, какая из букв – «н» или «к» - встречается в нём раньше при просмотре слева направо (принять, что указанные буквы в строке есть).

Таблица 5 – Код программы individual\_2.py

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":  sentence = input("Введите предложение: ")  n\_count = 0  k\_count = 0  for i in sentence:  if i == 'н' or i == 'Н':  n\_count += 1  elif i == 'к' or i == 'К':  k\_count += 1  if n\_count == 0 or k\_count == 0:  print("В строке нет букв 'н' и/или 'к'")  exit(1)  for i in sentence:  if i == 'н' or i == 'Н':  print("Буква 'н' встречается раньше")  break  elif i == 'к' or i == 'К':  print("Буква 'к' встречается раньше")  break |



Рисунок 5.1 – Вывод программы individual\_2.py в случае, если одной из букв «н» или «к» нет в слове



Рисунок 5.2 – Вывод программы individual\_2.py в случае, если буква «к» встречается раньше буквы «н»



Рисунок 5.3 – Вывод программы individual\_3.py в случае, если буква «н» встречается раньше буквы «к»

Индивидуальное задание 3. Дано ошибочно написанное слово «рпроцессо». Путём перемещения его букв получить слово процессор.

Таблица 6 – Код программы individual\_3.py

|  |
| --- |
|  |

Контрольные вопросы

1. Для чего нужны диаграммы деятельности UML?

Диаграмма деятельности показывает поток переходов от одной деятельности к другой. Деятельность – это продолжающийся во времени неатомарный шаг вычислений в автомате. Деятельность в конечном итоге приводит к выполнению некоего действия, составленного из выполняемых атомарный вычислений, каждое из которых либо изменяет состояние системы, либо возвращает какое-то значение.

1. Что такое состояние действия и состояние деятельности?

Состояние действия представляет собой точку в процессе, где выполняется какое-то действие или операция. Используется для обозначения конкретного действия или операции, которая выполняется в данном этапе процесса.

Состояние деятельности представляет собой более общую концепцию и обозначает фазу, в которой система находится в процессе выполнения какой-то деятельности. Используется для обозначения более широких блоков выполнения, включающих в себя несколько действий. Состояние деятельности может охватывать последовательность действий или вложенных подсостояния.

1. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?

В диаграммах деятельности для обозначения переходов и ветвлений применяются следующие нотации:

* Стрелка, указывающая направление перехода между состояниями или действиями;
* Ветвления – линия, разделяющаяся на две или более линии, представляющие параллельные пути выполнения;
* Слияния – линии, сходящиеся от двух или более линий, представляющих собой объединение параллельных путей выполнения.

1. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры – это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия.

1. Чем отличается разветвляющийся алгоритм от линейного?

Линейных алгоритм выполняется последовательно без разветвлений, тогда как разветвляющийся алгоритм содержит условия или ветвления, позволяющие изменять ход выполнения в зависимости от условий или входных данных.

1. Что такое условный оператор? Какие существуют его формы?

Условный оператор в языке программирования Python используется для выполнения блока кода в зависимости от истинности или ложности некоторого условия. Основной формой считается «if-else», однако можно использовать также формы «if», «if-elif» и «if-elif-else».

1. Какие операторы сравнения используются в Python?

В языке Python используются следующие операторы сравнения:

* «==» (равно) - проверяет, равны ли два значения;
* «!=» (не равно) - проверяет, не равны ли два значения;
* «<» (меньше) – проверяет, меньше ли левое значение правого;
* «<=» (меньше или равно) – проверяет, меньше или равно ли левое значение правого;
* «>» (больше) – проверяет, больше ли левое значение правого;
* «>=» (больше или равно) – проверяет, больше или равно ли левое значение правого;

1. Что называется простым условием? Приведите примеры.

Простое условие – это условие, которое проверяет истинность или ложность одного выражения. Пример:

*x = 5*

*if x == 5:*

1. Что такое составное условие? Приведите примеры.

Составное условие – это комбинация нескольких простых условий, объединённых логическими операторами. Пример:

*age = 25*

*if age >= 18 and age <= 30:*

1. Какие логические операторы допускаются при составлении сложных условий?

При составлении сложных условий допускаются следующие операторы:

* «and» (логическое «И») – возвращает true, если условия истинны, или false, если одно из условий ложно;
* «or» (логическое «ИЛИ») – возвращает true, если хотя бы одно из условий истинно;
* «not» (логическое «НЕ») – инвертирует значение условия.

1. Может ли оператор ветвления содержать внутри себя другие ветвления?

Оператор ветвления может содержать внутри себя другие ветвления, создавая вложенные условные конструкции.

1. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры – это алгоритм, в котором происходит многократное повторение одного и того же участка программы. Такие повторяемые участки вычислительного процесса называются циклами.

1. Типы циклов в языке Python.

В языке программирования Python существует несколько типов циклов:

* Цикл «for» - перебирает элементы в итерируемом объекте (списке, строке). Пример использования – *for i in range (5):*
* Цикл «while» - выполняет блок кода, пока условие остаётся истинным. Пример использования – *while i < 5:*
* Цикл «for..in» - перебирает элементы в итерируемом объекте, таком как список или кортеж. Пример использования:

*fruits = [“яблоко”, “груша”, “апельсин”]*

*for fruit in fruits:*

1. Назовите назначение и способы применения функции range.

Функция range() в языке программирования Python используется для создания последовательности чисел в указанном диапазоне. Она облегчает создание итерируемых объектов, таких как списки, используемых в циклах.

Способы применения:

* Создание последовательности чисел от начального (включительно) до конечного (исключая) значения с определённым шагом – *numbers = list(range(1, 10, 2))*
* Использование в цикле *for* – *for i in range(5):*
* Генерация последовательности для создания списка – *numbers = list(range(5))*

1. Как с помощью функции range организовывать перебор значения от 15 до 0 с шагом 2?

*range(15, -1, -2)*

1. Могут ли быть циклы вложенными?

Циклы могут быть вложенными в языке программирования Python.

1. Как образуется бесконечный цикл и как выйти из него?

Бесконечный цикл образуется, когда условие цикла всего истинно. Это может произойти, например, при использовании цикла «while» без явного изменения переменной условия внутри блока цикла.

Чтобы выйти из бесконечного цикла, нужно использовать оператор break внутри блока.

1. Для чего нужен оператор break?

Оператор «break» позволяет прервать выполнения цикла и перейти к следующему за блоком цикла оператору.

1. Где употребляется оператор continue и для чего он используется?

Оператор «continue» используется в циклах в языке программирования Python и служит для пропуска текущей итерации цикла и перехода к следующей.

Встречая оператор «continue» внутри цикла, выполнения последующего кода в блоке цикла прерывается и начинается заново с следующей итерацией цикла.

1. Для чего нужны стандартные потоки stdout и stderr?

Стандартный вывод (stdout) используется для вывода обычных данных, результатов работы программы, сообщений и т.д. По умолчанию, вывод «stdout» направляется на консоль.

Стандартный поток ошибок (stderr) используется для вывода сообщений об ошибках и предупреждений. Этот поток также направляется на консоль, но может быть перенаправлен в другие места для логирования и обработок ошибок.

1. Как в Python организовать вывод в стандартный поток stderr?

В Python для организации вывода в стандартный поток ошибок (stderr) можно воспользоваться библиотекой «sys». Пример использования:

*sys.stderr.write(“Сообщение, выводимое в stderr”)*

1. Каково назначение функции exit?

Функция exit() в языке программирования Python используется для немедленного завершения выполнения программы. Код возврата 0 указывает на успешное завершение программы, а другие значения могут использоваться для указания на различные ошибочные ситуации.

**Выводы**: В процессе выполнения лабораторной работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры, освоены операторы языка Python версии 3.x if, while, for, break и continue, написаны 9 программ (5 примеров, 3 индивидуальных задания и 1 задание повышенной сложности), а так же построены 6 ULM-диаграмм деятельности.