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 ПРАКТИЧЕСКОЙ РАБОТЕ №5
дисциплины «Программирование на Python»

Выполнил:
Лейс Алексей Вячеславович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем»

_____
(подпись)

Руководитель практики: кандидат тех. наук доцент кафедры инфокоммуникаций: Воронкин Р.А

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2022 г.

Тема: Основы языка Python

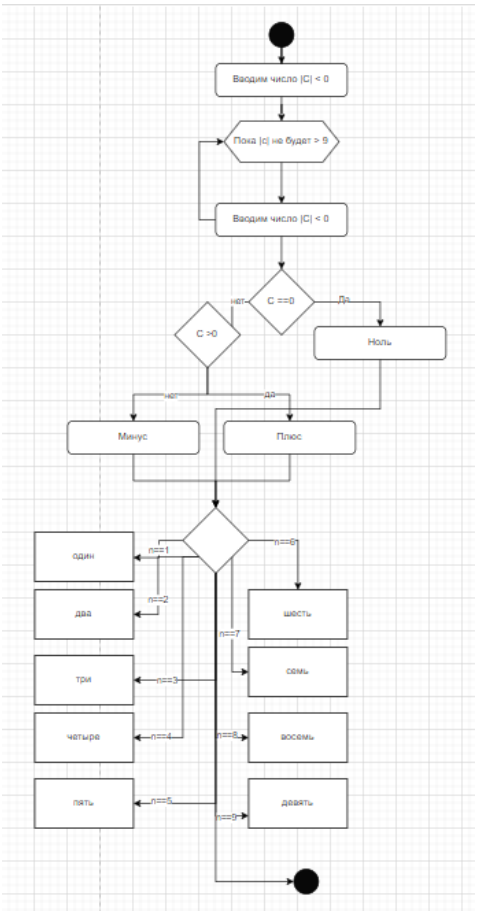
Цель работы: исследование процесса установки и базовых возможностей языка Python версии 3.x.

Порядок выполнения работы:

Дано целое число c такое, что $|c| < 9$. Вывести это число в словесной форме, учитывая его знак.

```python
print("Введите число |c|< 9 ")
c = int(input())
while abs(c) > 9:
    print("Введите число по правилу |c|< 9 ")
    c = int(input())

if c == 0:
    print("Ноль")
else:
    if c > 0 :
     print("Плюс")
    else:
     print("Минус")

if abs(c) ==1:
    print("Один")
elif abs(c) ==2:
    print("Два")
elif abs(c) ==3:
    print("Три")
elif abs(c) ==4:
    print("Четыре")
elif abs(c) ==5:
    print("Пять")
elif abs(c) ==6:
     print("Шесть")
elif abs(c) ==7:
    print("Семь")
elif abs(c) == 8:
    print("Восемь")
elif abs(c) ==9:
    print("Девять")
```

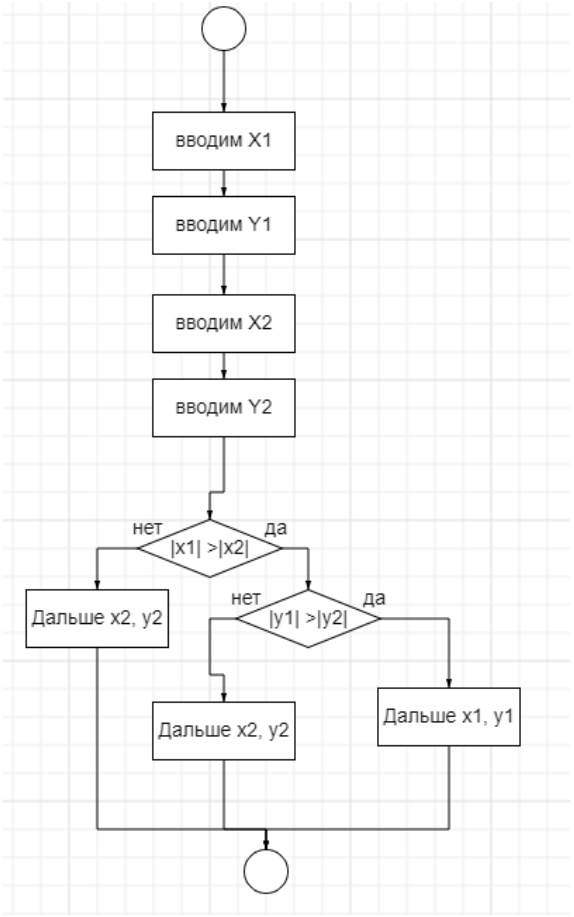
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

python.python-2023.18.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\lau
Введите число |c|< 9
-3
Минус
Три
PS I:\ИВТ-б-о-22-1\Прогграммирование на Python>  i:; cd 'i:\ИВТ-б-о-22-1\Проггра
python.python-2023.18.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\lau
Введите число |c|< 9
6
Плюс
Шесть
PS I:\ИВТ-б-о-22-1\Прогграммирование на Python>
```



Какая из точек $A(a_1, a_2)$ или $B(b_1, b_2)$ находится дальше от начала координат?

```python
print("Введитие A x1 координату:")
x1 = int(input())

print("Введитие A y1 координату:")
y1 = int(input())

print("Введитие B x2 координату:")
x2 = int(input())

print("Введитие B y2 координату:")
y2 = int(input())
if  abs(x1) > abs(x2):
    if abs(y1) > abs(y2):
        print("A дальше")
    else:
        print("B дальше")
else:
    print("B дальше")
```

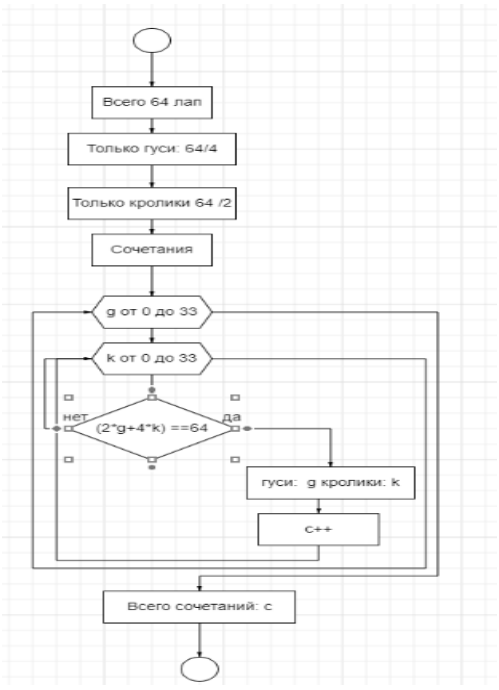
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
-300
500
Введитие x2 координату:
1000
Введитие y2 координату:
-800
PS I:\ИВТ-б-о-22-1\Прогграммирование на Python>  i:; cd 'i:\
python.python-2023.18.0\pythonFiles\lib\python\debugpy\adapt
Введитие x1 координату:
-60
PS I:\ИВТ-б-о-22-1\Прогграммирование на Python>  i:; cd 'i:\
Введитие x1 координату:
-600
Введитие y1 координату:
-300
Введитие x2 координату:
90
Введитие y2 координату:
5
A дальше
```



У гусей и кроликов вместе 64 лапы. Сколько могло быть кроликов и гусей (указать все сочетания, которые возможны).

```python
#4 у кроликов  и 2 у гусей #
print("Всего лап 64")
print("Только Гуси: ",64/4)
print("Только Кролики: ",64/2)
print("Гуси и Кролики пополам все сочетания:")
c =0
for g in range(0,33):
    for k in range(0,33):
        if (2*g+4*k ==64):
            print(f"Гуси: {g}, Кролики: {k}")
            c+=1
print("Гуси и Кролики пополам все сочетания: ",c)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Только Гуси:  16.0
Только Кролики:  32.0
Гуси и Кролики пополам все сочетания:
Гуси: 0, Кролики: 16
Гуси: 2, Кролики: 15
Гуси: 4, Кролики: 14
Гуси: 6, Кролики: 13
Гуси: 8, Кролики: 12
Гуси: 10, Кролики: 11
Гуси: 12, Кролики: 10
Гуси: 14, Кролики: 9
Гуси: 16, Кролики: 8
Гуси: 18, Кролики: 7
Гуси: 20, Кролики: 6
Гуси: 22, Кролики: 5
Гуси: 24, Кролики: 4
Гуси: 26, Кролики: 3
Гуси: 28, Кролики: 2
Гуси: 30, Кролики: 1
Гуси: 32, Кролики: 0
Гуси и Кролики пополам все сочетания:  17
```

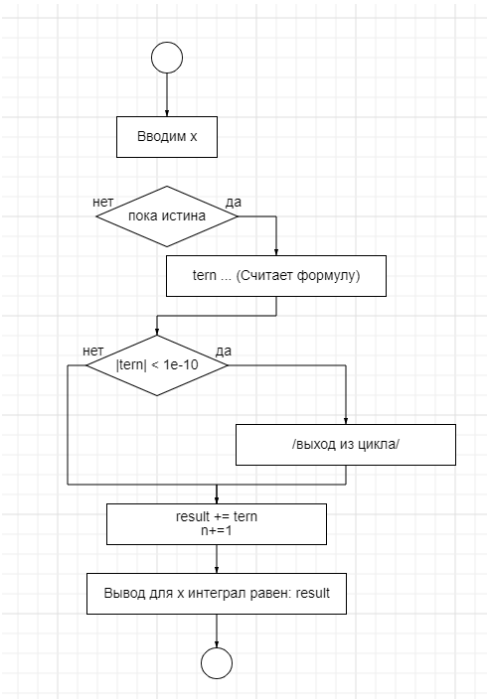


1. Интегральный синус:

$$\mathrm{Si}(x) = \int_0^x \frac{\sin t}{t}\,dt = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)(2n+1)!}. \tag{11}$$

```python
import math

x = float(input("Введите значение x: "))
n = 0
result = 0

while True:
    term = ((-1) ** n) * (x ** (2 * n + 1)) / ((2 * n + 1) * math.factorial(2 * n +
    if abs(term) < 1e-10:  # Проверяем точность
        break
    result += term
    n += 1

print(f"Значение интеграла для x = {x} равно {result}")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS I:\ИВТ-б-о-22-1\Прогграммирование на Python\5>  & 'D:\LAV\Files\Programs\VisualStudio\Shared\Pyt
de\extensions\ms-python.python-2023.18.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launc
аммирование на Python\5\4.py'
Введите значение x: 5
Значение интеграла для x = 5.0 равно 1.5499312449693072
```



Ответы на вопросы:

1. Для чего нужны диаграммы деятельности UML?
    - Диаграммы деятельности UML используются для визуализации и моделирования деятельности или процессов в системах. Они помогают описать последовательность шагов, роли, решения и потоки управления в системе.
2. Что такое состояние действия и состояние деятельности?
    - Состояние действия (action state) в диаграммах деятельности UML представляет мгновенное выполнение какой-либо операции или действия. Состояние деятельности (activity state) представляет долгосрочное выполнение действий и может содержать вложенные диаграммы деятельности.
3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?
    - Для обозначения переходов используется стрелка с меткой условия, а для ветвлений - ромбовидный символ с условием.
4. Какой алгоритм является алгоритмом разветвляющейся структуры?
    - Алгоритм разветвляющейся структуры представляет собой условное ветвление, такое как оператор if в Python.
5. Чем отличается разветвляющийся алгоритм от линейного?
    - Разветвляющийся алгоритм позволяет программе принимать разные пути выполнения в зависимости от условий, в то время как линейный алгоритм выполняется последовательно без ветвлений.
6. Что такое условный оператор? Какие существуют его формы?
    - Условный оператор позволяет выполнить определенный блок кода при определенных условиях. В Python, основные формы условных операторов - if, elif (else if), и else.
7. Какие операторы сравнения используются в Python?
    - В Python используются следующие операторы сравнения: == (равно), != (не равно), < (меньше), > (больше), <= (меньше или равно), >= (больше или равно).
8. Что называется простым условием? Приведите примеры.

- Простое условие - это условие, которое содержит только одну проверку. Например: **if x > 5:**.

9. Что такое составное условие? Приведите примеры.
    - Составное условие - это условие, которое содержит несколько проверок, объединенных логическими операторами. Например: **if x > 5 and y < 10:**.
10. Какие логические операторы допускаются при составлении сложных условий?
    - В Python используются логические операторы: and (и), or (или), not (не).
11. Может ли оператор ветвления содержать внутри себя другие ветвления?
    - Да, оператор ветвления может содержать внутри себя другие операторы ветвления, создавая вложенные условия.
12. Какой алгоритм является алгоритмом циклической структуры?
    - Алгоритм циклической структуры позволяет выполнить блок кода многократно. Примером является цикл for или цикл while в Python.
13. Типы циклов в языке Python.
    - В Python существуют два основных типа циклов: цикл for (для перебора элементов) и цикл while (для повторения операций до выполнения условия).
14. Назовите назначение и способы применения функции range.
    - Функция range используется для генерации последовательности чисел, которые могут быть использованы в циклах for. Синтаксис: **range(start, stop, step)**.
15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?
    - **for i in range(15, -1, -2):**
16. Могут ли быть циклы вложенными?
    - Да, циклы могут быть вложенными, то есть один цикл может находиться внутри другого.
17. Как образуется бесконечный цикл и как выйти из него?

- Бесконечный цикл образуется, если условие цикла всегда истинно. Чтобы выйти из бесконечного цикла, можно использовать комбинацию клавиш Ctrl+C или применить оператор break.

18. Для чего нужен оператор break?
    - Оператор break используется для прерывания выполнения цикла и выхода из него досрочно, если выполняется определенное условие.
19. Где употребляется оператор continue и для чего он используется?
    - Оператор continue используется внутри цикла для перехода к следующей итерации, игнорируя оставшуюся часть текущей итерации.
20. Для чего нужны стандартные потоки stdout и stderr?
    - Стандартные потоки stdout (стандартный вывод) и stderr (стандартный поток ошибок) используются для вывода информации и ошибок из программы.
21. Как в Python организовать вывод в стандартный поток stderr?
    - Можно использовать sys.stderr, например: **import sys; sys.stderr.write("Ошибка")**.
22. Каково назначение функции exit?
    - Функция **exit** используется для завершения выполнения программы. Она может быть использована для выхода из программы с определенным кодом завершения.

Вывод: в ходе работы был исследован процесс установки и базовых возможностей языка Python версии 3.x.