МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Кафедра инфокоммуникаций

Отчет

по лабораторной работе №1

«Условные операторы и циклы в языке Python»

по дисциплине:

«Введение в системы искусственного интеллекта»

Вариант 7

Выполнил: студент группы ИВТ-б-о-18-1 (2)
Криворучко Ярослав Евгеньевич
_____(подпись)
Проверил:
Воронкин Роман Александрович
_____(подпись)

Ставрополь, 2022 г.

Цель работы: приобретение навыков программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоить операторы языка Python версии 3.x if , while , for , break и continue , позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

Задание №1

7. С клавиатуры вводится цифра m (от 1 до 12). Вывести на экран название месяца, соответствующего цифре.

Ввод [9]:
```
m = int(input("Введите номер месяца: "))
if m == 1 or m == 2 or m == 12:
    print("Зима")
elif m >= 3 and m <= 5:
    print("Весна")
elif m >= 6 and m <= 8:
    print("Лето")
elif m >= 9 and m <= 11:
    print("Осень")
else:
    print("Ошибка")
```

```
Введите номер месяца: 10
Осень
```

Рисунок 1 – Код программы

Задание №2

7. Провести исследование биквадратного уравнения $a\,x^4 + b\,x^2 + c = 0$ ($a \neq 0$), где a, b и c - действительные числа. Если действительных корней нет, то об этом должно быть выдано сообщение, иначе должны быть выданы 2 или 4 действительных корня.

Ввод [63]:

```
import math
a = float(input())
b = float(input())
c = float(input())
D = pow(b,2)- (4 * a * c)

if a == 0:
    if b == 0 and (c == 0 or c != 0):
        print(3)
    if b != 0 and (c != 0 or c == 0):
        x = -(c / b)

else:
    if D > 0:
        x1= ((-b) - (math.sqrt(D)))/ (2 * a)
        print(x1)
        if(x1>0):
            x1 = math.sqrt(x1)
        else:
            x1 = 0
        x2= ((-b) + (math.sqrt(D)))/ (2 * a)
        print(x)
        if(x2>0):
            x2 = math.sqrt(x2)
        else:
            x2 = 0
        if(x1==0 and x2==0):
            print("Корней нет")
        elif (x1==0 and x2>0):
            print("2 корня: ",x2,",",x2*-1)
        elif (x1>0 and x2==0):
            print("2 корня: ",x1,",",x1*-1)
        else:
            print("4 корня: ",x2,",",x2*-1, ",",x1,",", x1*-1)
    elif D == 0:
        x = ((-b) / (2 * a))**0.5
        print("2 корня: ",x,",",x*-1)
    elif D < 0:
        print("Корней нет")
```

```
1
-10
9
1.0
0.5773502691896257
4 корня:  3.0 , -3.0 , 1.0 , -1.0
```

Рисунок 2 – Код программы

Задание №3

7. Определить среди всех двузначных чисел те, которые делятся на сумму своих цифр.

Ввод [5]:

```
for i in range(10,100):
    sum = i / 10 + i % 10
    if (i % int(sum) == 0):
        print(i)
```

```
10
12
18
20
21
24
27
30
36
40
42
45
48
50
54
60
63
70
72
80
81
84
90
```

Рисунок 4 – Код программы

Задание №4

7. Функция Бесселя первого рода $I_n(x)$, значение $n = 0, 1, 2, \ldots$ также должно вводиться с клавиатуры

$$I_n(x) = \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} \frac{(x^2/4)^k}{k!(k+n)!}. \qquad (17)$$

Ввод [1]:
```
import math
import sys

#Точность вычислений
EPS = 1e-10

x = float(input("Введите x: "))
n = int(input("Введите n: "))
if x == 0:
   print("Недопустимое значение x", file=sys.stderr)
   exit(1)
a = x
S, k = a, 0

while math.fabs(a) > EPS:

   a *= ((x**2/4)**math.factorial(k)*(math.factorial(k)*math.factorial(k+n)))/((k+1)*math.factorial(k+1+n)*(x**2/4)**math.factorial(k+1))

   S += a
   k += 1

print(f"In({x}) = {a+(x/2)**n}")
```

```
Введите x: 5
Введите n: 5
In(5.0) = 97.65625
```

Рисунок 5 – Код программы

Файл 2.2.ipynb с решением задач находится на **Github**: https://github.com/YaroStavr/LR1_Artificial-Intelligence.git

Ответы на вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности — это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов поведения системы. Диаграмма деятельности — это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, однако, по сравнению с последней, у ней есть явные преимущества: поддержка многопоточности и объектно-ориентированного проектирования.

Диаграмма деятельности (Activity diagram) показывает поток переходов от одной деятельности к другой. Деятельность (Activity) – это продолжающийся во времени неатомарный шаг вычислений в автомате. Деятельности в конечном счете приводят к выполнению некоего действия (Action), составленного из выполняемых атомарных вычислений, каждое из которых либо изменяет состояние системы, либо возвращает какое-то значение. Действие может заключаться в вызове другой операции, посылке сигнала, создании или уничтожении объекта либо в простом вычислении - скажем, значения выражения. Графически диаграмма деятельности

представляется в виде графа, имеющего вершины и ребра.

2. Что такое состояние действия и состояние деятельности?

Состояния действия не могут быть подвергнуты декомпозиции. Кроме того, они атомарны. Это значит, что внутри них могут происходить различные события, но выполняемая в состоянии действия работа не может быть прервана. Обычно предполагается, что длительность одного состояния действия занимает неощутимо малое время.

В противоположность этому состояния деятельности могут быть подвергнуты дальнейшей декомпозиции, вследствие чего выполняемую деятельность можно представить с помощью других диаграмм деятельности. Состояния деятельности не являются атомарными, то есть могут быть прерваны. Предполагается, что для их завершения требуется заметное время. Можно считать, что состояние действия — это частный вид состояния деятельности, а конкретнее – такое состояние, которое не может быть подвергнуто дальнейшей декомпозиции. А состояние деятельности можно представлять себе как составное состояние, поток управления которого включает только другие состояния деятельности и действий.

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

Переходы. Когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока используются переходы, показывающие путь из одного состояния действия или деятельности в другое. В UML переход представляется простой линией со стрелкой.

Ветвление. Простые последовательные переходы встречаются наиболее часто, но их одних недостаточно для моделирования любого потока управления. Как и в блок-схеме, вы можете включить в модель ветвление, которое описывает различные пути выполнения в зависимости от значения некоторого булевского выражения. В точку ветвления может входить ровно

один переход, а выходить - два

или более. Для каждого исходящего перехода задается булевское выражение, которое вычисляется только один раз при входе в точку ветвления. Ни для каких двух исходящих переходов эти сторожевые условия не должны одновременно принимать значение «истина», иначе поток управления окажется неоднозначным. Но эти условия должны покрывать все возможные варианты, иначе поток остановится.

Для удобства разрешается использовать ключевое слово else для пометки того из исходящих переходов, который должен быть выбран в случае, если условия, заданные для всех остальных переходов, не выполнены.

Реализовать итерацию можно, если ввести два состояния действия - в первом устанавливается значение счетчика, во втором оно увеличивается - и точку ветвления, вычисление в которой показывает, следует ли прекратить итерации.

4. 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. Программа разветвляющейся структуры реализует такой алгоритм.

5. Чем отличается разветвляющийся алгоритм от линейного?

В программе разветвляющейся структуры имеется один или несколько условных операторов. Для программной реализации условия используется логическое выражение. В сложных структурах с большим числом ветвей применяют оператор выбора.

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

Оператор ветвления if позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Возможны следующие варианты использования.

Синтаксис оператора if выглядит так.

```
if выражение:
    инструкция_1
    инструкция_2
    ...
    инструкция_n
```

После оператора if записывается выражение. Если это выражение истинно, то выполняются инструкции, определяемые данным оператором. Выражение является истинным, если его результатом является число не равное нулю, непустой объект, либо логическое True. После выражения нужно поставить двоеточие “:”.

Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. Т. е. при истинном условии нужно выполнить один набор инструкций, при ложном –другой. Для этого используется конструкция if – else.

```
if выражение:
    инструкция_1
    инструкция_2
    ...
    инструкция_n
else:
    инструкция_a
    инструкция_b
    ...
    инструкция_x
```

Для реализации выбора из нескольких альтернатив можно использовать конструкцию if – elif – else.

```
if выражение_1:
    инструкции_(блок_1)
elif выражение_2:
    инструкции_(блок_2)
elif выражение_3:
    инструкции_(блок_3)
else:
    инструкции_(блок_4)
```

7. Какие операторы сравнения используются в Python?

Операторы сравнения в Python используются для сравнения двух объектов. Возвращаемый результат – логическое значение – True или False.

В Python есть 6 типов операторов сравнения:

== Возвращает True, если два операнда равны, в противном случае –

False. Пример: a == b

!= Возвращает True, если два операнда не равны, в противном случае – False. Пример: a != b

> Возвращает True, если левый операнд больше правого, в противном случае – False. Пример: a > b

< Возвращает True, если левый операнд меньше правого, в противном случае – False. Пример: a < b

> = Возвращает True, если левый операнд больше или равен правому операнду, в противном случае – False. Пример: a > b

<= Возвращает True, если левый операнд меньше или равен правому операнду, в противном случае – False. Пример: a < b

8. Что называется простым условием? Приведите примеры.

Простейшие условия состоят из одного отношения (больше, меньше и т.п.)

```
a = 3
if a > 1:
   print("hello 3")
```

9. Что такое составное условие? Приведите примеры.

Простейшие условия состоят из одного отношения (больше, меньше и т.п.) Но иногда необходимо объединение простых условий в более сложные, Например, на улице холодно и идет дождь. Два простых условия (на улице холодно), (на улице идет дождь) здесь связаны связкой И.

СЛОЖНОЕ УСЛОВИЕ – состоит из двух или нескольких простых отношений (условий), которые объединяются с помощью логических операций:

И - логическое умножение - на языке Python записывается как and,

ИЛИ - логическое сложение - на языке Python записывается как or,

НЕ - логическое отрицание -на языке Python записывается как not.

```
>>> x = 8
>>> y = 13
>>> y < 15 and x > 8
False
```

10. Какие логические операторы допускаются при составлении сложных условий?

И - логическое умножение - на языке Python записывается как and,

ИЛИ - логическое сложение - на языке Python записывается как or,

НЕ - логическое отрицание -на языке Python записывается как not.

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

Да может. Оператор ветвления if позволяет выполнить определенный набор инструкций в зависимости от некоторого условия.

12. 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.

13. Типы циклов в языке Python.

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

14. Назовите назначение и способы применения функции range.

Функция range возвращает неизменяемую последовательность чисел в виде объекта range. Функция range хранит только информацию о значениях start, stop и step и вычисляет значения по мере необходимости. Это значит, что независимо от размера диапазона, который описывает функция range, она всегда будет занимать фиксированный объем памяти.

15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?

List(gange(15, 0, -2))

16. Могул ли быть циклы вложенными?

Да могут

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

While True:

break

18. Для чего нужен оператор break ?

Оператор break предназначен для досрочного прерывания работы цикла while.

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

Оператор continue запускает цикл заново, при этом код, расположенный после данного оператора, не выполняется.

20. Для чего нужны стандартные потоки stdout и stderr?

В операционной системе по умолчанию присутствуют стандартных потока вывода на консоль:

буферизованный поток stdout для вывода данных и информационных сообщений, а также небуферизованный поток stderr для вывода сообщений об ошибках. По умолчанию функция print использует поток stdout. Для того, чтобы использовать поток stderr необходимо передать его в параметре file функции print. Само же определение потоков stdout и stderr находится в стандартном пакете Python sys. Хорошим стилем программирования является наличие вывода ошибок в стандартный поток stderr поскольку вывод в потоки stdout и stderr может обрабатываться как операционной системой, так и сценариями пользователя по разному.

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

Для того, чтобы использовать поток stderr необходимо передать его в параметре file функции print.

print("Ошибка!", file=sys.stderr)

22. Каково назначение функции exit ?

В Python завершить программу и передать операционной системе заданный код возврата можно посредством функции exit.