**МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Институт цифрового развития**

**Кафедра информационных систем и технологий**

Отчет по лабораторной работе №5.

Дисциплина: **«**Основы программной инженерии**»**

**Выполнил:**

Студент группы ПИЖ-б-о-22-1, направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Гуртовой Ярослав Д

**Проверил:**

Воронкин Р. А.

Ставрополь 2022

Тема: Лабораторная работа 2.1 Основы языка Python

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

Выполнение работы:

1. Изучил теоретический материал работы.
2. Создал репозиторий на git.hub.
3. Клонировал репозиторий.

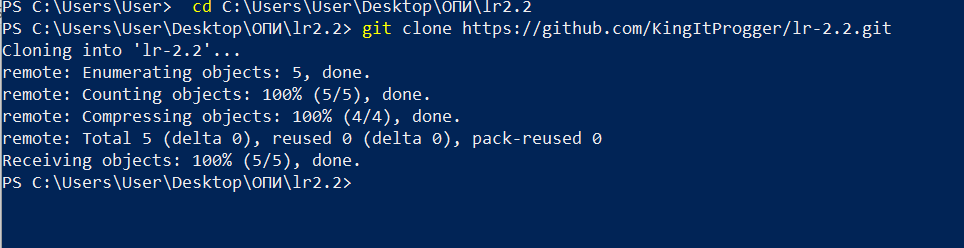


Рисунок 2 – клонирование репозитория.

Дополнить файл gitignore необходимыми правилами.

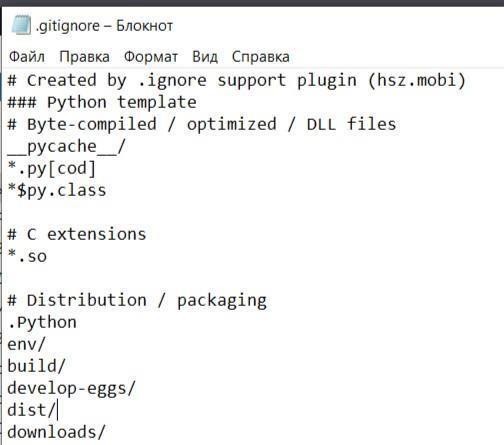


Рисунок 3 - – .gitignore для IDE PyCharm

1. Организовать свой репозиторий в соответствии с моделью ветвления git-flow.



Рисунок 4 – создание ветки develop

1. Проработать примеры лабораторной работы. Создать для каждого примера отдельный модуль языка Python. Зафиксировать изменения в репозитории. Привести в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры.

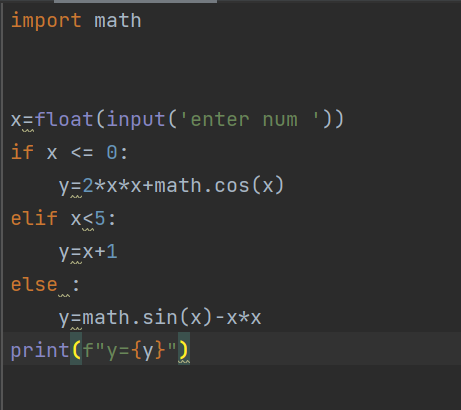


Рисунок 5 – пример 1

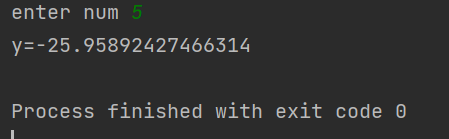


Рисунок 6 – примеры выполнения для примера 1

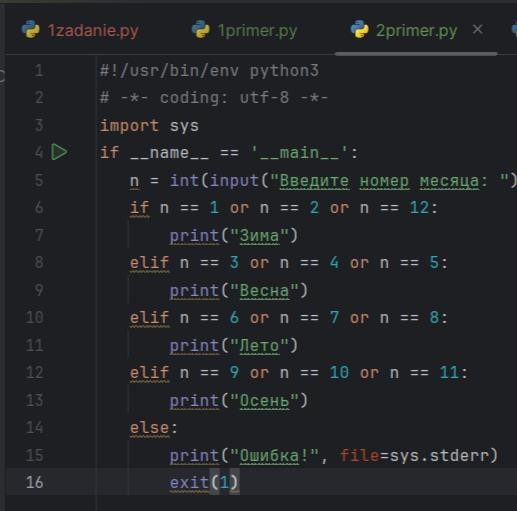


Рисунок 7 – пример 2

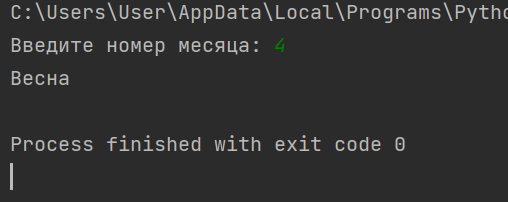


Рисунок 8 – примеры выполнения для 2 примера

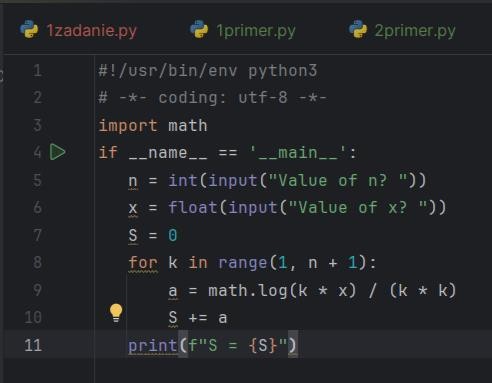


Рисунок 9 – пример 3

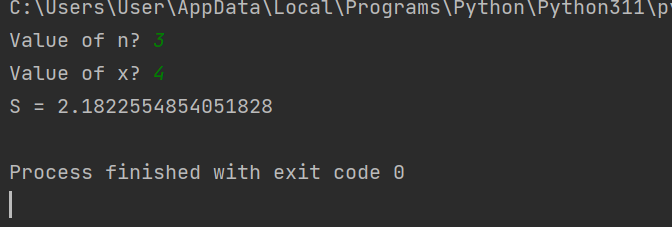


Рисунок 10 – примеры выполнения для примера 3

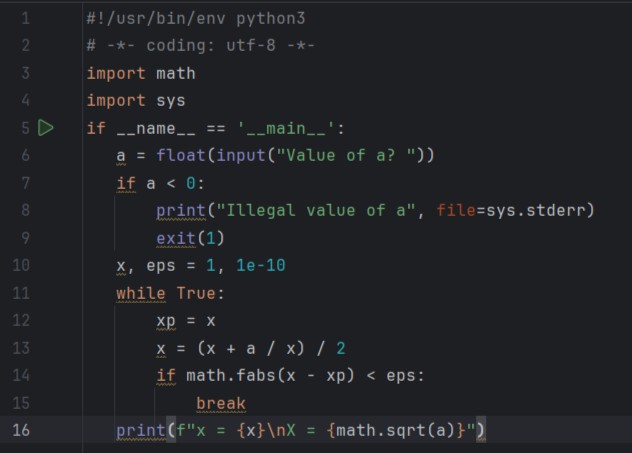


Рисунок 10 – пример 4

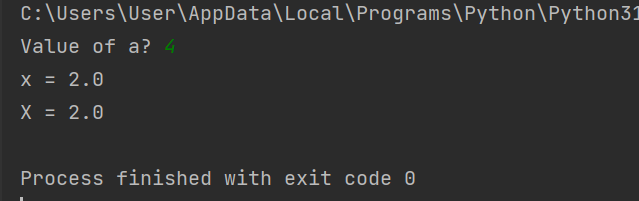


Рисунок 11 – примеры выполнения для примера 4

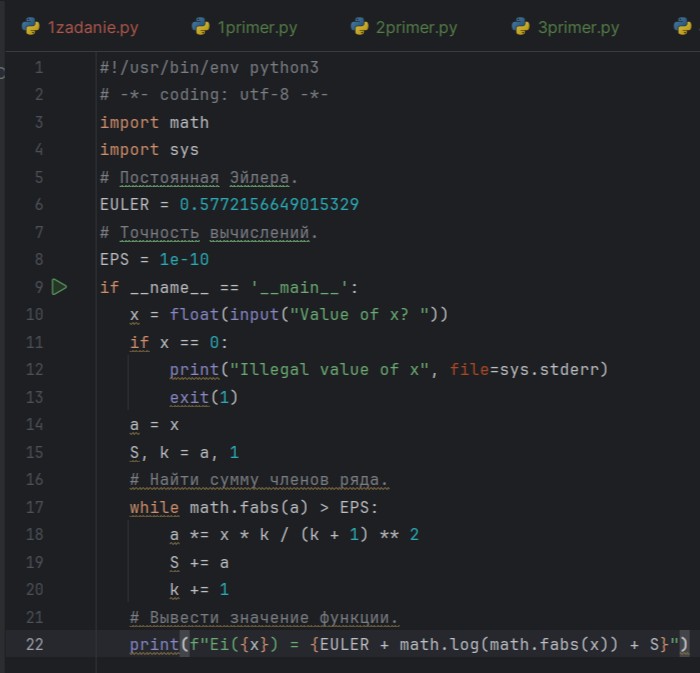


Рисунок 12 – пример 5

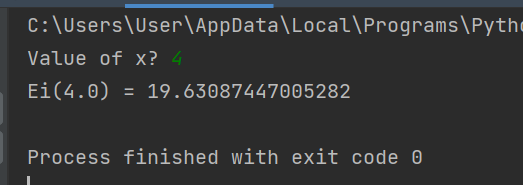


Рисунок 13 – примеры выполнения для примера 5

1. Для примеров 4 и 5 построить UML-диаграмму деятельности. Для построения диаграмм деятельности использовать веб-сервис Google <https://www.diagrams.net/>.

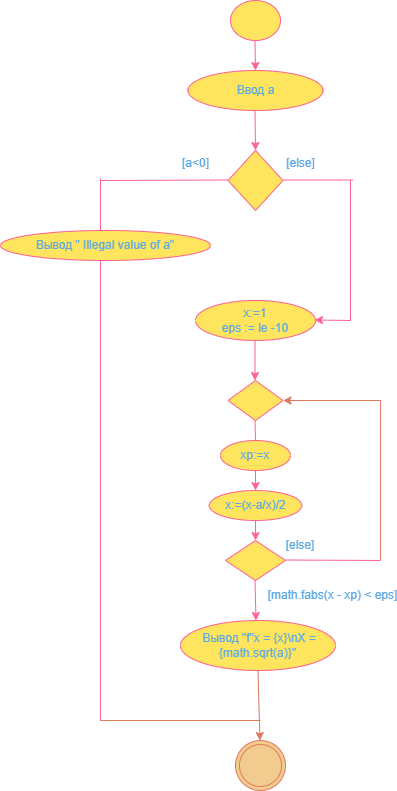


Рисунок 14 - UML-диаграмма деятельности для примера 4

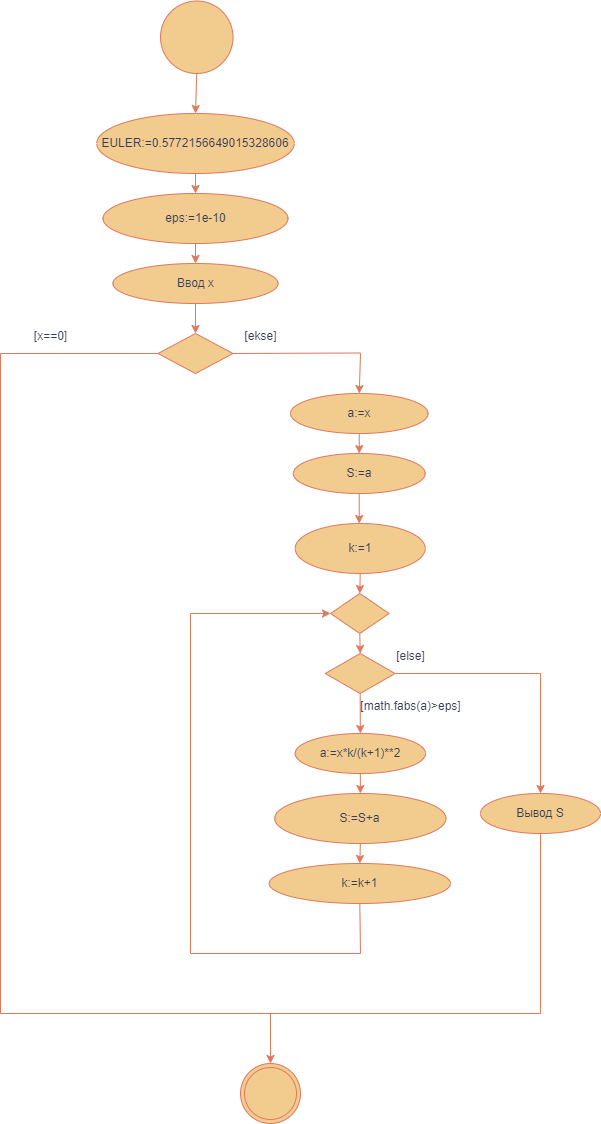


Рисунок 15 - UML-диаграмма деятельности для примера 5

1. С клавиатуры вводится цифра (от 1 до 4). Вывести на экран названия месяцев, соответствующих времени года с номером (считать зиму временем года

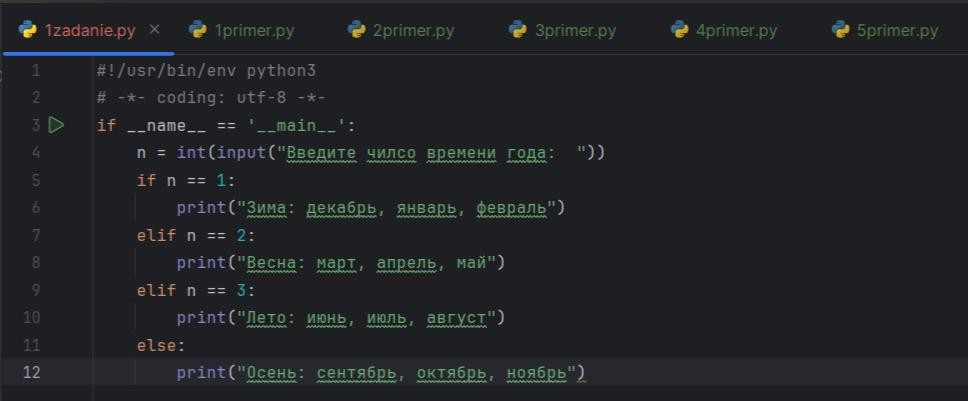
№ 1).

Рисунок 16 – решение задания 1

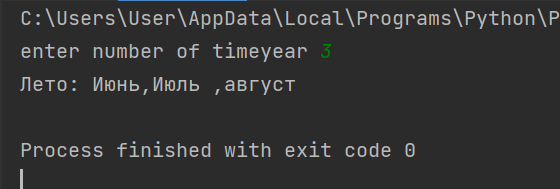


Рисунок 17 – результат выполнения задания 1

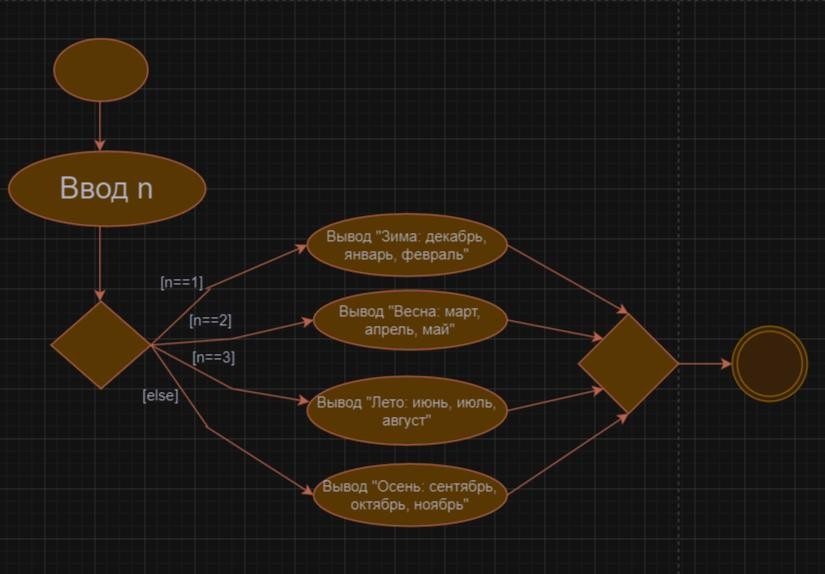


Рисунок 18 - UML-диаграмму деятельности для задания 1

1. Определить принадлежит ли точка A(a,b) кольцу определяемому окружностями x^2+y^2=1 x^2+y^2=0.25.

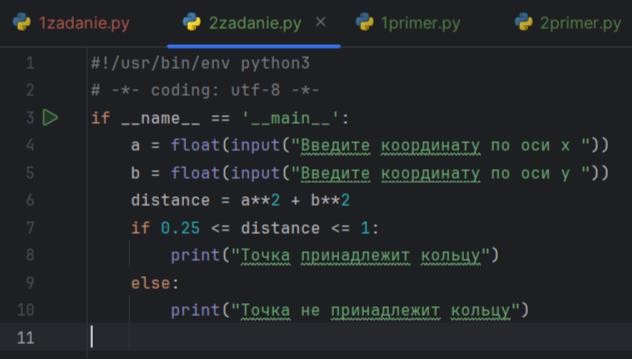


Рисунок 19 – решение задания 2

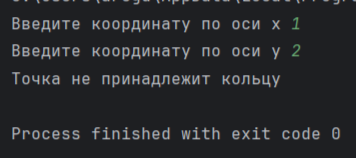


Рисунок 20 – результат выполнения задания 2

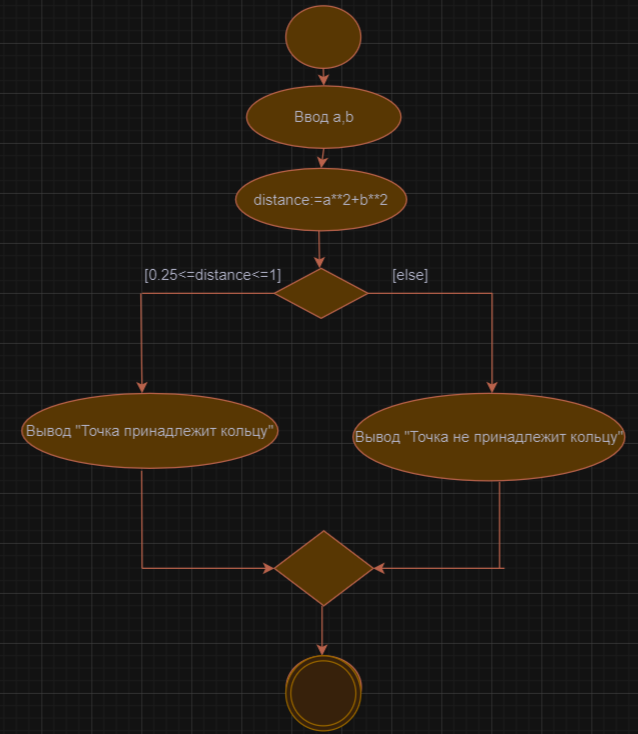


Рисунок 21 - UML-диаграмму деятельности для задания 2

1. Одноклеточная амеба каждые три часа делится на 2 клетки. Определить, сколько будет клеток через 6 часов.

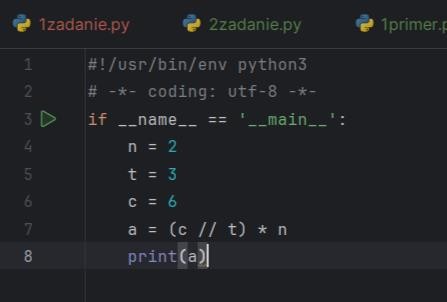


Рисунок 22 – решение задания 3

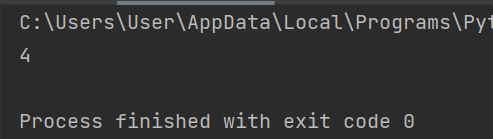


Рисунок 23 – результат выполнения задания 3

1. Зафиксировал все изменения в ветку develop.

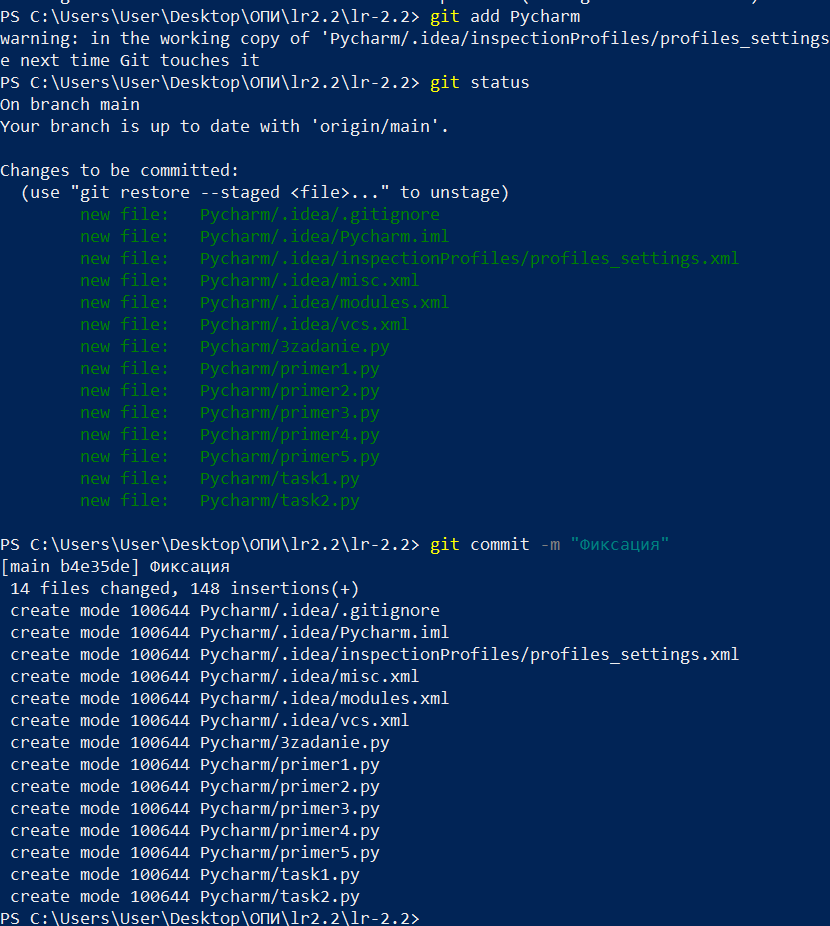


Рисунок 28 – фиксация изменений в ветку develop

Контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем. UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию.

1. Что такое состояние действия и состояние деятельности?

В потоке управления, моделируемом диаграммой деятельности, происходят различные события. Вы можете вычислить выражение, в результате чего изменяется значение некоторого атрибута или возвращается некоторое значение. Также, например, можно выполнить операцию над объектом, послать

ему сигнал или даже создать его или уничтожить. Все эти выполняемые атомарные вычисления называются состояниями действия, поскольку каждое из них есть состояние системы, представляющее собой выполнение некоторого действия. 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.

1. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах 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. Поток управления должен где-то начинаться и заканчиваться.

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.

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. Программа разветвляющейся структуры реализует такой алгоритм. 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.

1. Чем отличается разветвляющийся алгоритм от линейного?

Линейный алгоритм — это такой, в котором все операции выполняются последовательно одна за другой.

Алгоритмы разветвленной структуры применяются, когда в зависимости от некоторого условия необходимо выполнить либо одно, либо другое действие.

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

Оператор ветвления if позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Существует несколько форм конструкиций – if, if – else, if – elif – else

1. Какие операторы сравнения используются в Python?

В языках программирования используются специальные знаки, подобные тем, которые используются в математике: > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), == (равно), != (не равно).

1. Что называется простым условием? Приведите примеры Простым условием называется выражение, составленное из двух

арифметических выражений или двух текстовых величин связанных одним из знаков. Например, логические выражение типа kByte >= 1023 является простым, так как в нём выполняется только одна логическая операция.

1. Что такое составное условие? Приведите примеры.

Составное условие – логическое выражение, содержащее несколько простых условий, объединенных логическими операциями. Например, "на улице идет снег или дождь", "переменная news больше 12 и меньше 20".

1. Какие логические операторы допускаются при составлении сложных условий?

В таких случаях используются специальные операторы, объединяющие два и более простых логических выражения. Широко используются два оператора – так называемые логические И (and) и ИЛИ (or).

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

Да, внутри оператора ветвления можно определить и другие ветвления

1. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры – это алгоритм, в котором предусмотрено неоднократное выполнение одной и той же последовательности действий.

1. Типы циклов в языке Python.

В Python есть два вида циклов: while и for.

1. Назовите назначение и способы применения функции range.

Функция range возвращает неизменяемую последовательность чисел в виде объекта range. Синтаксис функции:

range(stop)

range(start, stop[, step])

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.

Самый простой вариант range - передать только значение stop. Если передаются два аргумента, то первый используется как start, а второй - как stop. И чтобы указать шаг последовательности надо передать три аргумента.

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

for x in range (15, -1, -2): print(x)

1. Могут ли быть циклы вложенными?

Существует возможность организовать цикл внутри тела другого цикла.

Такой цикл будет называться вложенным циклом.

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

Чтобы организовать бесконечный цикл, используют конструкцию while (true). При этом он, как и любой другой цикл, может быть прерван командой break или сам прекратит работу, когда его условие работы не равно True.

1. Для чего нужен оператор break?

Оператор break предназначен для досрочного прерывания работы цикла

while.

1. Где употребляется оператор continue и для чего он используется? Оператор continue запускает цикл заново, при этом код, расположенный

после данного оператора, не выполняется

1. 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. 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.

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

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.

1. Каково назначение функции exit?

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