Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе №1

По теме “Определение модели языка. Выбор инструментальной языковой среды.”

Выполнил:

студент гр. 853505

Лазарева Е. В.

Проверил:

Ст. преподаватель КИ Шиманский В. В.

Минск 2021

Содержание

1. **Цель работы**
2. **Подмножество языка программирования**
   1. **Числовые и строковые константы**
   2. **Типы переменных**
   3. **Условные операторы**
3. **Инструментальная языковая среда**

**Примечание. Код программы**

1. Цель работы

Необходимо определить подмножество языка программирования (типы констант, переменных, операторов и функций). В подмножество как минимум должны быть включены:

- числовые и текстовые константы;

- 3-4 типа переменных;

- операторы цикла (**do**...**while**, **for**);

- условные операторы (**if**...**else,** **case**).

Определение инструментальной языковой среды, т.е. языка программирования и операционной системы для разработки включает:

- язык программирования c указанием версии, на котором ведётся разработка (напр. Python 3.7);

- операционная система (Windows, Linux и т.д.), в которой выполняется разработка;

- компьютер (PC / Macintosh).

В данном отчете по лабораторной работе дается полное определение подмножества языка программирования, тексты 2-3-х программ, включающих все элементы этого подмножества. Приводится подробное описание инструментальной языковой среды.

1. Подмножество языка программирования

В лабораторных работах будет проведен анализ и построение интерпретатора подмножества языка Python.

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нем программ. Язык является полностью объектно-ориентированным – все является объектами. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объем полезных функций.

Необычной особенностью языка является выделение блоков кода пробельными отступами. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нем программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как Си или C++.

Каждое значение в Python имеет тип. Поскольку все в Python – объекты, типы являются классами, а значения – экземплярами (объектами) этих классов. В подмножестве будут реализованы следующие типы:

* 1. Числовые и строковые константы

**Числа.** Числа в языке Python представлены тремя встроенными типами: целые (int), вещественные (float) и комплексные (complex). Целые числа могут быть любой длины и ограничиваются только доступной памятью. Числа с плавающей точкой имеют ограниченную точность. Визуально разницу между целым числом и числом с плавающей точкой можно заметить в консоли по наличию точки: 1 – целое число, 1.0 или 1. – число с плавающей точкой.

**Строки.** Строка представляет собой последовательность символов. Для создания строки можно использовать одинарные или двойные кавычки, а многострочные строки можно обозначить тройными кавычками ''' или """. Также возможно использование оператора [] со строками. Стоит отметить, что строки в Python относятся к категории неизменяемых последовательностей, то есть все функции и методы могут лишь создавать новую строку.

* 1. Типы переменных

Python поддерживает динамическую типизацию, то есть тип переменной определяется только во время исполнения. Поэтому вместо «присваивания значения переменной» лучше говорить о «связывании значения с некоторым именем». К примитивным типам в Python относятся булевый, целое число произвольной точности, число с плавающей точкой и комплексное число. Из контейнерных типов в Python встроены: строка, список, кортеж (неизменяемый список), словарь и множество. Все значения являются объектами, в том числе функции, методы, модули, классы.

**Операторы цикла.** В интерпретаторе будут реализованы следующие операторы циклов:

* **for** – выполняет тело цикла, итерируясь по объекту (например, списку или строке)
* **while** – выполняет тело цикла до тех пор, пока условие цикла истинно
* **continue** – начинает следующий проход цикла, не исполняя оставшееся тело цикла
* **break** – оператор, который прерывает исполнение цикла
  1. Условные операторы

В интерпретаторе будут реализованы условные операторы if-elif-else:

**Оператор if**

if выражение:

инструкция\_1

…

инструкция\_n

**Конструкция if-else**

if выражение:

инструкция\_1

…

инструкция\_n

else:

инструкция\_x

…

инструкция\_y

1. Инструментальная языковая среда

Для разработки интерпретатора подмножества языка Python будет использован язык программирования C#. Разработка проекта будет вестись в среде Visual Studio.

Операционной системой будет выступать Windows 10. Платформа – обычный PC. Интерфейс пользователя будет реализован с помощью терминала Windows PowerShell.

В качестве языковой среды выбран язык программирования C#. Разработка основана на работе с операционной системой Windows на PC.

C# - объектно-ориентированный язык программирования для разработки приложений для платформы Microsoft .NET Framework. Был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщенные типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников – языков C++, Delphi, Модула, Smalltalk и, в особенности, Java – C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).

Примечание. Код программы

Пример 1. Пирамидальная сортировка

def heapify(nums, heap, root) :  
 largest = root  
 left = (2 \* root) + 1  
 right = (2 \* root) + 2  
 if left < heap and nums[left] > nums[largest] :  
 largest = left  
 if right < heap and nums[right] > nums[largest] :  
 largest = right  
 if largest != root :  
 nums[root] = nums[largest]  
 nums[largest] = nums[root]  
 heapify(nums, heap, largest)  
  
def heap\_sort(nums) :  
 n = len(nums)  
 for i in range(n, -1, -1) :  
 heapify(nums, n, i)  
 for i in range(n - 1, 0, -1) :  
 nums[i] = nums[0]  
 nums[0] = nums[i]  
 heapify(nums, i, 0)  
  
random\_nums = [46, 18, 1, 35, 26]  
heap\_sort(random\_nums)  
print('Пирамидальная сортировка')  
print(random\_nums)

Вывод:

Пирамидальная сортировка

[1, 18, 26, 35, 46]

Пример 2. Сортировка слиянием

def merge(left\_list, right\_list) :  
 sorted = []  
 left\_index = right\_index = 0  
 left\_length, right\_length = len(left\_list), len(right\_list)  
 for \_ in range(left\_length + right\_length) :  
 if left\_index < left\_length and right\_index < right\_length :  
 if left\_list[left\_index] <= right\_list[right\_index] :  
 sorted.append(left\_list[left\_index])  
 left\_index += 1  
 else :  
 sorted.append(right\_list[right\_index])  
 right\_index += 1  
 elif left\_index == left\_length :  
 sorted.append(right\_list[right\_index])  
 right\_index += 1  
 elif right\_index == right\_length :  
 sorted.append(left\_list[left\_index])  
 left\_index += 1  
 return sorted  
  
def merge\_sort(nums) :  
 if len(nums) <= 1 :  
 return nums  
 mid = len(nums) // 2  
 left\_list = merge\_sort(nums[:mid])  
 right\_list = merge\_sort(nums[mid:])  
 return merge(left\_list, right\_list)  
  
random\_nums = [64, 83, 50, 168, 5]  
random\_nums = merge\_sort(random\_nums)  
print('Сортировка слиянием')  
print(random\_nums)

Вывод:

Сортировка слиянием

[5, 50, 64, 83, 168]