ЛАБОРАТОРНАЯ РАБОТА № 3
ИЗУЧЕНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Время выполнения – 6 часов.

Цель работы: познакомиться со средой разработки Python. Изучить основной синтаксис языка программирования.

Задачи работы

1. Ознакомиться с языком программирования Python;
2. Научиться решать базовые задачи на языке программирования python;

Перечень обеспечивающих средств

Для выполнения работы необходимо иметь компьютер с установленной операционной системой семейства Windows, установленным python и IDE PyCharm Professional.

Общие теоретические сведения

Python представляет популярный высокоуровневый язык программирования, который предназначен для создания приложений различных типов. Это и веб-приложения, и игры, и настольные программы, и работа с базами данных. Довольно большое распространение python получил в области машинного обучения и исследований искусственного интеллекта.

Впервые язык Python был анонсирован в 1991 году голландским разработчиком Гвидо Ван Россумом. С тех пор данный язык проделал большой путь развития. В 2000 году была издана версия 2.0, а в 2008 году - версия 3.0. Несмотря на вроде такие большие промежутки между версиями

постоянно выходят подверсии. Так, текущей актуальной версией на момент написания данного материала является 3.9. Более подробную информацию о всех релизах, версиях и изменения языка, а также собственно интерпретаторы и необходимые утилиты для работы и прочую полезную информацию можно найти на официальном сайте https://www.python.org/.

После установки интерпретатора, как было описано в лабораторной работе №1, можно начать создавать приложения на Python.

Программа на языке Python состоит из набора инструкций. Каждая инструкция помещается на новую строку. Например:

```
print(2 + 3)
print("Hello")
```

Большую роль в Python играют отступы. Неправильно поставленный отступ фактически является ошибкой. Например, в следующем случае мы получим ошибку, хотя код будет практически аналогичен приведенному выше:

```
print(2 + 3)
    print("Hello")
```

Поэтому стоит помещать новые инструкции сначала строки. В этом одно из важных отличий пайтона от других языков программирования, как C# или Java.

Однако стоит учитывать, что некоторые конструкции языка могут состоять из нескольких строк. Например, условная конструкция if:

```
if 1 < 2:
    print("Hello")
```

В данном случае если 1 меньше 2, то выводится строка "Hello". И здесь уже должен быть отступ, так как инструкция print("Hello") используется не сама по себе, а как часть условной конструкции if. Причем отступ, согласно руководству по оформлению кода, желательно делать из такого количество пробелов, которое кратно 4 (то есть 4, 8, 16 и т.д.) Хотя если отступов будет не 4, а 5, то программа также будет работать.

Таких конструкций не так много, поэтому особой путаницы по поводу где надо, а где не надо ставить пробелы, не должно возникнуть.

Регистрозависимость

Python – регистрозависимый язык, поэтому выражения print и Print или PRINT представляют разные выражения. И если вместо метода print для вывода на консоль мы попробуем использовать метод Print:

```
Print("Hello World")
```

то у нас ничего не получится.

Комментарии

Для отметки, что делает тот или иной участок кода, применяются комментарии. При трансляции и выполнении программы интерпретатор игнорирует комментарии, поэтому они не оказывают никакого влияния на работу программы.

Комментарии в Python бывают блочные и строчные. Все они предваряются знаком решетки (#).

Блочные комментарии ставятся в начале строки:

```
# Вывод сообщения на консоль
print("Hello World")
```

Строчные комментарии располагаются на той же строке, что и инструкции языка:

```
print("Hello World")  # Вывод сообщения на консоль
```

Основные функции

Python предоставляет ряд встроенных функций. Некоторые из них используются очень часто, особенно на начальных этапах изучения языка, поэтому рассмотрим их.

Основной функцией для вывода информации на консоль является функция print(). В качестве аргумента в эту функцию передается строка, которую мы хотим вывести:

```
print("Hello Python")
```

Если же нам необходимо вывести несколько значений на консоль, то мы можем передать их в функцию print через запятую:

```
print("Full name:", "Tom", "Smith")
```

Если функция print отвечает за вывод, то функция input отвечает за ввод информации. В качестве необязательного параметра эта функция принимает приглашение к вводу и возвращает введенную строку, которую мы можем сохранить в переменную:

```
name = input("Введите имя: ")
print("Привет", name)
```

Переменные и типы данных

Переменная хранит определенные данные. Название переменной в Python должно начинаться с алфавитного символа или со знака подчеркивания и может содержать алфавитно-цифровые символы и знак подчеркивания. И кроме того, название переменной не должно совпадать с названием ключевых слов языка Python. Ключевых слов не так много, их легко запомнить: and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield.

Например, создадим переменную:

```
name = "Tom"
```

Здесь определена переменная name, которая хранит строку "Tom".

Переменная хранит данные одного из типов данных. В Python существует множество различных типов данных, которые подразделяются на категории: числа, последовательности, словари, наборы:

boolean - логическое значение True или False

int - представляет целое число, например, 1, 4, 8, 50.

float - представляет число с плавающей точкой, например, 1.2 или 34.76

complex - комплексные числа

str - строки, например "hello". В Python 3.x строки представляют набор символов в кодировке Unicode

bytes - последовательность чисел в диапазоне 0-255

byte array - массив байтов, аналогичен bytes с тем отличием, что может изменяться

list - список

tuple - кортеж

set - неупорядоченная коллекция уникальных объектов

frozen set - то же самое, что и set, только не может изменяться (immutable)

dict - словарь, где каждый элемент имеет ключ и значение

Python является языком с динамической типизацией. Он определяет тип данных переменной исходя из значения, которое ей присвоено. Так, при присвоении строки в двойных или одинарных кавычках переменная имеет тип str. При присвоении целого числа Python автоматически определяет тип переменной как int. Чтобы определить переменную как объект float, ей присваивается дробное число, в котором разделителем целой и дробной части является точка.

Число с плавающей точкой можно определять в экспоненциальной записи:

```
x = 3.9e3
print(x)  # 3900.0

x = 3.9e-3
print(x)  # 0.0039
```

Число float может иметь только 18 значимых симолов. Так, в данном случае используются только два символа - 3.9. И если число слишком велико или слишком мало, то мы можем записывать число в подобной нотации, используя экспоненту. Число после экспоненты указывает степень числа 10, на которое надо умножить основное число - 3.9.

При этом в процессе работы программы мы можем изменить тип переменной, присвоив ей значение другого типа:

```
user_id = "12tomsmith438"  # тип str
print(user_id)

user_id = 234  # тип int
print(user_id)
```

С помощью функции type() динамически можно узнать текущий тип переменной:

```
user_id = "12tomsmith438"
print(type(user_id))  # <class 'str'>

user_id = 234
print(type(user_id))  # <class 'int'>
```

Арифметические операции

Ниже представлена таблица с кратким обзором математических операторов, доступных в Python.

Операция	**Возвращаемое значение**
x + y	Сумма x и y.
x — y	Разность x и y.
-x	Изменение знака x.
+x	Тождественность x.
x * y	Произведение x и y.
x / y	Частное от деления x на y.
x // y	Частное от целочисленного деления x на y.
x % y	Остаток от деления x / y.
x ** y	x в степени y.

Арифметические операции с присвоением

Ряд специальных операций позволяют использовать присвоить результат операции первому операнду:

+=

Присвоение результата сложения

-=

Присвоение результата вычитания

*=

Присвоение результата умножения

/=

Присвоение результата от деления

//=

Присвоение результата целочисленного деления

**=

Присвоение степени числа

%=

Присвоение остатка от деления

Примеры операций:

```
number = 10
number += 5
print(number)  # 15

number -= 3
print(number)  # 12

number *= 4
print(number)  # 48
```

Ряд операций представляют условные выражения. Все эти операции принимают два операнда и возвращают логическое значение, которое в Python представляет тип boolean. Существует только два логических значения - True (выражение истинно) и False (выражение ложно).

Операции сравнения

Простейшие условные выражения представляют операции сравнения, которые сравнивают два значения. Python поддерживает следующие операции сравнения:

==

Возвращает True, если оба операнда равны. Иначе возвращает False.

!=

Возвращает True, если оба операнда НЕ равны. Иначе возвращает False.

> (больше чем)

Возвращает True, если первый операнд больше второго.

< (меньше чем)

Возвращает True, если первый операнд меньше второго.

>= (больше или равно)

Возвращает True, если первый операнд больше или равен второму.

<= (меньше или равно)

Возвращает True, если первый операнд меньше или равен второму.

Примеры операций сравнения:

```
a = 5
b = 6
result = 5 == 6  # сохраняем результат операции в переменную
print(result)  # False - 5 не равно 6
print(a != b)  # True
print(a > b)  # False - 5 меньше 6
print(a < b)  # True

bool1 = True
bool2 = False
print(bool1 == bool2)  # False - bool1 не равно bool2
```

Операции сравнения могут сравнивать различные объекты - строки, числа, логические значения, однако оба операнда операции должны представлять один и тот же тип.

Логические операции

Для создания составных условных выражений применяются логические операции. В Python имеются следующие логические операторы:

and (логическое умножение)

Возвращает True, если оба выражения равны True

or (логическое сложение)

Возвращает True, если хотя бы одно из выражений равно True

not (логическое отрицание)

Возвращает True, если выражение равно False

Если один из операндов оператора and возвращает False, то другой операнд уже не оценивается, так как оператор в любом случае возвратит False. Подобное поведение позволяет немного увеличить производительность, так как не приходится тратить ресурсы на оценку второго операнда.

Аналогично если один из операндов оператора or возвращает True, то второй операнд не оценивается, так как оператор в любом случае возвратит True.

Операции со строками

Строка представляет последовательность символов в кодировке Unicode, заключенных в кавычки. Причем в Python мы можем использовать как одинарные, так и двойные кавычки:

```
name = "Tom"
surname = 'Smith'
print(name, surname)  # Tom Smith
```

Одной из самых распространенных операций со строками является их объединение или конкатенация. Для объединения строк применяется знак плюса:

```
name = "Tom"
surname = 'Smith'
fullname = name + " " + surname
print(fullname)  # Tom Smith
```

С объединением двух строк все просто, но что, если нам надо сложить строку и число? В этом случае необходимо привести число к строке с помощью функции str():

```
name = "Tom"
age = 33
info = "Name: " + name + " Age: " + str(age)
print(info)  # Name: Tom Age: 33
```

Условная конструкция if

Условные конструкции используют условные выражения и в зависимости от их значения направляют выполнение программы по одному из путей. Одна из таких конструкций - это конструкция if. Она имеет следующее формальное определение:

```
if логическое_выражение:
    инструкции
[elif логическое выражение:
    инструкции]
[else:
    инструкции]
```

В самом простом виде после ключевого слова if идет логическое выражение. И если это логическое выражение возвращает True, то выполняется последующий блок инструкций, каждая из которых должна начинаться с новой стоки и должна иметь отступы от начала строки:

```
age = 22
if age > 21:
    print("Доступ разрешен")
print("Завершение работы")
```

Поскольку в данном случае значение переменной age больше 21, то будет выполняться блок if

Отступ желательно делать в 4 пробела или то количество пробелов, которое кратно 4.

Обратите внимание в коде на последнюю стоку, которая выводит сообщение "Завершение работы". Она не имеет отступов от начала строки, поэтому она не принадлежит к блоку if и будет выполняться в любом случае, даже если выражение в конструкции if возвратит False.

Но если бы мы поставили бы отступы, то она также принадлежала бы к конструкции if

Если вдруг нам надо определить альтернативное решение на тот случай, если условное выражение возвратит False, то мы можем использовать блок else:

```
age = 18
if age > 21:
    print("Доступ разрешен")
else:
    print("Доступ запрещен")
```

Если выражение age > 21 возвращает True, то выполняется блок if, иначе выполняется блок else.

Если необходимо ввести несколько альтернативных условий, то можно использовать дополнительные блоки elif, после которого идет блок инструкций.

```
age = 18
if age >= 21:
    print("Доступ разрешен")
elif age >= 18:
    print("Доступ частично разрешен")
else:
    print("Доступ запрещен")
```

Вложенные конструкции if

Конструкция if в свою очередь сама может иметь вложенные конструкции if:

```
age = 18
if age >= 18:
    print("Больше 17")
    if age > 21:
        print("Больше 21")
    else:
        print("От 18 до 21")
```

Стоит учитывать, что вложенные выражения if также должны начинаться с отступов, а инструкции во вложенных конструкциях также должны иметь отступы. Отступы, расставленные не должным образом, могут изменить логику программы. Так, предыдущий пример НЕ аналогичен следующему:

```
age = 18
if age >= 18:
    print("Больше 17")
if age > 21:
    print("Больше 21")
else:
    print("От 18 до 21")
```

Циклы

Циклы позволяют повторять некоторое действие в зависимости от соблюдения некоторого условия.

Цикл while

Первый цикл, который мы рассмотрим, это цикл while. Он имеет следующее формальное определение:

```
while условное_выражение:
    инструкции
```

После ключевого слова while указывается условное выражение, и пока это выражение возвращает значение True, будет выполняться блок инструкций, который идет далее.

Все инструкции, которые относятся к циклу while, располагаются на последующих строках и должны иметь отступ от начала строки.

Цикл for

Другой тип циклов представляет конструкция for. Цикл for вызывается для каждого числа в некоторой коллекции чисел. Коллекция чисел создается с помощью функции range(). Формальное определение цикла for:

```
for int_var in функция_range:
    инструкции
```

После ключевого слова for идет переменная int_var, которая хранит целые числа (название переменной может быть любое), затем ключевое слово in, вызов функции range() и двоеточие.

А со следующей строки располагается блок инструкций цикла, которые также должны иметь отступы от начала строки.

При выполнении цикла Python последовательно получает все числа из коллекции, которая создается функцией range, и сохраняет эти числа в переменной int_var. При первом проходе цикл получает первое число из коллекции, при втором - второе число и так далее, пока не переберет все числа. Когда все числа в коллекции будут перебраны, цикл завершает свою работу.

Функция range

Функция range имеет следующие формы:

range(stop): возвращает все целые числа от 0 до stop

range(start, stop): возвращает все целые числа в промежутке от start (включая) до stop (не включая). Выше в программе факториала использована именно эта форма.

range(start, stop, step): возвращает целые числа в промежутке от start (включая) до stop (не включая), которые увеличиваются на значение step

Функции

Функции представляют блок кода, который выполняет определенную задачу и который можно повторно использовать в других частях программы. Формальное определение функции:

```
def имя_функции ([параметры]):
    инструкции
```

Определение функции начинается с выражения def, которое состоит из имени функции, набора скобок с параметрами и двоеточия. Параметры в скобках необязательны. А со следующей строки идет блок инструкций, которые выполняет функция. Все инструкции функции имеют отступы от начала строки.

Например, определение простейшей функции:

```
def say_hello():
    print("Hello")
```

Функция называется say_hello. Она не имеет параметров и содержит одну единственную инструкцию, которая выводит на консоль строку "Hello".

Для вызова функции указывается имя функции, после которого в скобках идет передача значений для всех ее параметров. Например:

```
def say_hello():
    print("Hello")

say_hello()
say_hello()
say_hello()
```

Функция main

В программе может быть определено множество функций. И чтобы всех их упорядочить, хорошей практикой считается добавление специальной функции main, в которой потом уже вызываются другие функции:

```
def main():
    say_hello("Tom")
    usd_rate = 56
    money = 30000
    result = exchange(usd_rate, money)
    print("К выдаче", result, "долларов")


def say_hello(name):
    print("Hello,", name)


def exchange(usd_rate, money):
    result = round(money/usd_rate, 2)
    return result

# Вызов функции main
main()
```

Список

Для работы с наборами данных Python предоставляет такие встроенные типы как списки, кортежи и словари.

Список (list) представляет тип данных, который хранит набор или последовательность элементов. Для создания списка в квадратных скобках ([]) через запятую перечисляются все его элементы. Во многих языках программирования есть аналогичная структура данных, которая называется массив. Например, определим список чисел:

```
numbers = [1, 2, 3, 4, 5]
```

Также для создания списка можно использовать конструктор list():

```
numbers1 = []
numbers2 = list()
```

Оба этих определения списка аналогичны - они создают пустой список.

Конструктор list для создания списока может принимать другой список:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
numbers2 = list(numbers)
```

Для обращения к элементам списка надо использовать индексы, которые представляют номер элемента в списка. Индексы начинаются с нуля. То есть второй элемент будет иметь индекс 1. Для обращения к элементам с конца

можно использовать отрицательные индексы, начиная с -1. То есть у последнего элемента будет индекс -1, у предпоследнего - -2 и так далее.

```
numbers = [1, 2, 3, 4, 5]
print(numbers[0])   # 1
print(numbers[2])   # 3
print(numbers[-3])  # 3

numbers[0] = 125  # изменяем первый элемент списка
print(numbers[0])   # 125
```

Кортежи

Кортеж (tuple) представляет последовательность элементов, которая во многом похожа на список за тем исключением, что кортеж является неизменяемым (immutable) типом. Поэтому мы не можем добавлять или удалять элементы в кортеже, изменять его.

Для создания кортежа используются круглые скобки, в которые помещаются его значения, разделенные запятыми:

```
user = ("Tom", 23)
print(user)
```

Для создания кортежа из списка можно передать список в функцию tuple(), которая возвратит кортеж:

```
users_list = ["Tom", "Bob", "Kate"]
users_tuple = tuple(users_list)
print(users_tuple)      # ("Tom", "Bob", "Kate")
```

Словари

Наряду со списками и кортежами Python имеет еще одну встроенную структуру данных, которая называется словарь (dictionary). В ряде языков программирования есть похожие структуры (словарь в C#, ассоциативный массив в PHP).

Как и список, словарь хранит коллекцию элементов. Каждый элемент в словаре имеет уникальный ключ, с которым ассоциировано некоторое значение.

Определение словаря имеет следующий синтаксис:

```
dictionary = { ключ1:значение1, ключ2:значение2, ....}
```

Определим пару словарей:

```
users = {1: "Tom", 2: "Bob", 3: "Bill"}
elements = {"Au": "Золото", "Fe": "Железо", "H": "Водород", "O": "Кислород"}
```

В словаре users в качестве ключей используются числа, а в качестве значений - строки. В словаре element в качестве ключей используются строки.

Множества

Множество (set) представляют еще один вид набора элементов. Для определения множества используются фигурные скобки, в которых перечисляются элементы:

```
users = {"Tom","Bob","Alice", "Tom"}
print(users)    # {"Tom","Bob","Alice"}
```

Обратите внимание, что несмотря на то, что функция print вывела один раз элемент "Tom", хотя в определении множества этот элемент содержится два раза. Все потому что множество содержит только уникальные значения.

Работа с файлами

Открытие и закрытие файлов

Python поддерживает множество различных типов файлов, но условно их можно разделить на два виде: текстовые и бинарные. Текстовые файлы - это к примеру файлы с расширением cvs, txt, html, в общем любые файлы, которые сохраняют информацию в текстовом виде. Бинарные файлы - это изображения, аудио и видеофайлы и т.д. В зависимости от типа файла работа с ним может немного отличаться.

При работе с файлами необходимо соблюдать некоторую последовательность операций:

Открытие файла с помощью метода open()

Чтение файла с помощью метода read() или запись в файл посредством метода write()

Закрытие файла методом close()

Открытие и закрытие файла

Чтобы начать работу с файлом, его надо открыть с помощью функции open(), которая имеет следующее формальное определение:

```
open(file, mode)
```

Первый параметр функции представляет путь к файлу. Путь файла может быть абсолютным, то есть начинаться с буквы диска, например, C://somedir/somefile.txt. Либо можно быть относительным, например, somedir/somefile.txt - в этом случае поиск файла будет идти относительно расположения запущенного скрипта Python.

Второй передаваемый аргумент - mode устанавливает режим открытия файла в зависимости от того, что мы собираемся с ним делать. Существует 4 общих режима:

r (Read). Файл открывается для чтения. Если файл не найден, то генерируется исключение FileNotFoundError

w (Write). Файл открывается для записи. Если файл отсутствует, то он создается. Если подобный файл уже есть, то он создается заново, и соответственно старые данные в нем стираются.

a (Append). Файл открывается для дозаписи. Если файл отсутствует, то он создается. Если подобный файл уже есть, то данные записываются в его конец.

b (Binary). Используется для работы с бинарными файлами. Применяется вместе с другими режимами - w или r.

После завершения работы с файлом его обязательно нужно закрыть методом close(). Данный метод освободит все связанные с файлом используемые ресурсы.

Например, откроем для записи текстовый файл "hello.txt":

```
myfile = open("hello.txt", "w")
myfile.close()
```

При открытии файла или в процессе работы с ним мы можем столкнуться с различными исключениями, например, к нему нет доступа и т.д. В этом случае программа выпадет в ошибку, а ее выполнение не дойдет до вызова метода close, и соответственно файл не будет закрыт.

Запись в текстовый файл

Чтобы открыть текстовый файл на запись, необходимо применить режим w (перезапись) или a (дозапись). Затем для записи применяется метод write(str), в который передается записываемая строка. Стоит отметить, что записывается именно строка, поэтому, если нужно записать числа, данные других типов, то их предварительно нужно конвертировать в строку.

Запишем некоторую информацию в файл "hello.txt":

```
with open("hello.txt", "w") as file:
    file.write("hello world")
```

Чтение файла

Для чтения файла он открывается с режимом r (Read), и затем мы можем считать его содержимое различными методами:

readline(): считывает одну строку из файла

read(): считывает все содержимое файла в одну строку

readlines(): считывает все строки файла в список

Например, считаем выше записанный файл построчно:

```
with open("hello.txt", "r") as file:
    for line in file:
        print(line, end="")
```

Задания по вариантам

Задание 1

Напишите программу, которая запрашивала бы у пользователя:

Вариант 1

Имя, Фамилия, Возраст, Место жительства

- фамилия, имя ("Ваши фамилия, имя?")

- возраст ("Сколько Вам лет?")

- место жительства ("Где вы живете?")

После этого выводила бы три строки:

"Ваши фамилия, имя"

"Ваш возраст"

"Вы живете в"

Вариант 2

Имя, Дата рождения, Образование

- имя ("Ваше, имя?")

- дата рождения ("Ваша дата рождения?")

- образование ("Где Вы учитесь?")

После этого выводила бы три строки:

"Ваше имя"

"Дата рождения"

"Вы учитесь в "

Вариант 3

Фамилия, Место жительства

- Фамилия("Ваша фамилия?")

- место жительства ("Где Вы живете?")

После этого выводила бы две строки:

"Ваша фамилия"

"Вы живете в"

Вариант 4

Фамилия, Место рождения, любимая музыка

- Фамилия, ("Ваша фамилия?")

- место рождения ("Где Вы родились?")

- музыка("Какая музыка нравится? ")

После этого выводила бы три строки:

"Ваши имя, фамилия"

"Вы родились в"

"Ваша любимая музыка "

Вариант 5

Имя, Фамилия, ФИО мамы, ФИО отца

- ФИО (например, "Ваши фамилия, имя, отчество?")
- возраст ("Сколько Вам лет?")
- место жительства ("Где Вы живете?")

После этого выводила бы три строки:

"Ваши имя, фамилия, отчество"

"Ваш возраст"

"Вы живете в"

Вариант 6

Имя, Любимый предмет в школе, Номер класса

- имя ("Ваше имя?")
- любимый предмет ("Какой Ваш любимый предмет в школе?")
- номер класса ("В каком классе Вы учитесь?")

После этого выводила бы три строки:

"Ваше имя"

"Ваш любимый предмет в школе"

"Вы учитесь в классе номер"

Вариант 8

Имя, Фамилия, Отчество, Хобби

- ФИО (например, "Ваши фамилия, имя, отчество?")
- хобби ("Чем Вы увлекаетесь?")

После этого выводила бы две строки:

"Ваши имя, фамилия, отчество"

"Ваше хобби"

Вариант 9

Имя, Фамилия, любимый спорт

- Фамилия, имя ("Ваши фамилия, имя?")
- образование ("В какой школе Вы учитесь?")
- ФИО Вашего руководителя по информатики ("ФИО Вашего руководителя по информатики?")

После этого выводила бы три строки:

"Ваши имя, фамилия"

"Вы учитесь в школе номер: "

"ФИО Вашего руководителя по информатике "

Вариант 10

Имя, Фамилия, Любимый предмет в школе (в институте), ФИО классного руководителя (куратора)

- Фамилия, имя ("Ваши фамилия, имя?")
- любимый предмет в школе ("Какой Ваш любимый предмет в школе?")
- ФИО классного руководителя ("ФИО Вашего классного руководителя?")

После этого выводила бы три строки:

"Ваши имя, фамилия"

"Ваш любимый предмет в школе "

"ФИО Вашего классного руководителя"

Вариант 11

Имя, Фамилия, Возраст, Дата рождения

- Фамилия, имя ("Ваши фамилия, имя?")
- возраст ("Сколько Вам лет?")
- дата рождения ("Когда Вы родились?")

После этого выводила бы три строки:

"Ваши имя, фамилия"

"Ваш возраст"

"Дата Вашего рождения"

Вариант 12

Имя, Фамилия, Место жительства, Месторождения

- Фамилия, имя ("Ваши фамилия, имя?")
- место рождения ("Где Вы родились?")
- место жительства ("Где Вы живете?")

После этого выводила бы три строки:

"Ваши имя, фамилия"

"Вы родились в"

"Вы живете в"

Вариант 13

Имя, Фамилия, Возраст, Номер телефона

- Фамилия, имя ("Ваши фамилия, имя?")
- возраст ("Сколько тебе лет?")
- номер телефона ("Номер Вашего телефона?")

После этого выводила бы три строки:

"Ваши имя, фамилия"

"Ваш возраст"

"Ваш номер телефона"

Вариант 14

Имя, Фамилия, Страна, Край , Город

- Фамилия, имя ("Ваши фамилия, имя?")
- страна ("В какой стране Вы живете?")
- город ("В каком городе Вы живете?")

После этого выводила бы три строки:

"Ваши имя, фамилия"

"Вы живете в стране"

"Вы живете в крае"

"Вы живете в городе"

Вариант 15

Имя, Фамилия, ФИО Вашего классного руководителя

- Фамилия, имя ("Ваши фамилия, имя?")

- ФИО Вашего классного руководителя ("ФИО Вашего классного руководителя?")

После этого выводила бы три строки:

"Ваши имя, фамилия"

"ФИО Вашего руководителя по информатике"

"ФИО Вашего классного руководителя"

Задание 2

Вариант 1

Даны три целых числа. Выбрать из них те, которые принадлежат интервалу [1,3].

Вариант 2

Дан номер года (положительное целое число). Определить количество дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный — 366 дней. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400 (например, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 — являются).

Вариант 3

Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется в том случае, если сумма покупки больше 500 руб., в 5% - если сумма больше 1000 руб.

Вариант 4

Написать программу, которая бы по введенному номеру единицы измерения (1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер) и массе M выдавала соответствующее значение массы в килограммах.

Вариант 5

Найти косинус минимального из 4 заданных чисел.

Вариант 6

Вывести на экран синус максимального из 3 заданных чисел.

Вариант 7

Даны три стороны одного треугольника и три стороны другого треугольника. Определить, будут ли эти треугольники равновеликими, т. е. имеют ли они равные площади. Если это не так, то вывести «Foul!!!»

Вариант 8

Составьте программу подсчёта площади равнобедренного треугольника. Если площадь треугольника чётная, разделить её на 2, в противном случае вывести сообщение «Не могу делить на 2!»

Вариант 9

Составить программу, которая по данному числу (1-12) выводит название соответствующего ему месяца на английском языке.

Вариант 10

Составить программу, осуществляющую перевод величин из радианной меры в градусную или наоборот. Программа должна запрашивать, какой перевод нужно осуществить, и выполнять указанное действие.

Вариант 11

Дано три числа. Найти количество положительных чисел среди них;

Вариант 12

Если действительные числа x и y – одного знака, найти их среднее геометрическое, в противном случае найти их среднее арифметическое.

Вариант 13

Определить, существует ли прямоугольный треугольник со сторонами x,y,z. Если – да, вычислить его площадь.

Вариант 14

Определить, существует ли треугольник с длинами сторон a, b, c. Если – да, вычислить его площадь по формуле Герона.

Формула Герона имеет вид:

$S=p(p-a)(p-b)(p-c)$, где $p=12(a+b+c)$

Вариант 15

Дано три числа. Найти количество отрицательных чисел среди них;

Задание 3

Вариант 1

1. Дано вещественное число – цена 1 кг конфет. Вывести стоимость 1, 2, … 10 кг конфет. Решить задачу используя циклическую конструкцию for.

2. Дана непустая последовательность целых чисел, оканчивающаяся нулем. Найти:

а) сумму всех чисел последовательности;

б) количество всех чисел последовательности

Решить задачу используя циклическую конструкцию while.

Вариант 2

1. Даны два числа A и B (A<B). Найти сумму всех целых чисел от A до B включительно. Решить задачу используя циклическую конструкцию for.

2. Дана последовательность отрицательных целых чисел, оканчивающаяся положительным числом. Найти среднее арифметическое всех чисел последовательности (без учета положительным числа).

Решить задачу используя циклическую конструкцию while.

Вариант 3

1. Даны два числа A и B (A<B). Найти сумму квадратов всех целых чисел от A до B включительно. Решить задачу используя циклическую конструкцию for.

2. Дана последовательность из n целых чисел. Первое число в последовательности чётное. Найти сумму всех идущих подряд в начале последовательности чётных чисел. Условный оператор не использовать

Решить задачу используя циклическую конструкцию while.

Вариант 4

1. Найти среднее арифметическое всех целых чисел от a до 200 (значения a и b вводятся с клавиатуры; a≤ 200). Решить задачу используя циклическую конструкцию for.

2. Дана последовательность из n вещественных чисел, начинающаяся с положительного числа. Определить, какое количество положительных чисел записано в начале последовательности. Условный оператор не использовать.

Решить задачу используя циклическую конструкцию while.

Вариант 5

1. Найти сумму всех целых чисел от a до b (значения a и b вводятся с клавиатуры; b ≥a). Решить задачу используя циклическую конструкцию for.

2.Дано целое число N (> 0), являющееся некоторой степенью числа 2: N = 2^K. Найти целое число K — показатель этой степени.

Решить задачу используя циклическую конструкцию while.

Вариант 6

1. Найти сумму квадратов всех целых чисел от a до 50 (значение a вводится с клавиатуры; $0 \le a \le 50$). Решить задачу используя циклическую конструкцию for.

2.Дано целое число N (> 1). Найти наименьшее целое число K, при котором выполняется неравенство $5^K > N$.

Решить задачу используя циклическую конструкцию while.

Вариант 7

1. Дана непустая последовательность целых чисел, оканчивающаяся нулем.

Найти:

а) сумму всех чисел последовательности;

б) количество всех чисел последовательности.

Решить задачу используя циклическую конструкцию for.

2.Дано целое число N (> 1). Найти наибольшее целое число K, при котором выполняется неравенство $2^K > N$.

Решить задачу используя циклическую конструкцию while.

Вариант 8

1. Дана последовательность из n вещественных чисел. Первое число в последовательности нечетное. Найти сумму всех идущих подряд в начале последовательности нечетных чисел. Условный оператор не использовать.

Решить задачу используя циклическую конструкцию for.

2.Дано целое число N (> 0). Используя операции деления нацело и взятия остатка от деления, найти количество и сумму его цифр.

Решить задачу используя циклическую конструкцию while.

Вариант 9

1. Среди чисел 1, 4, 9, 16, 25, ... найти первое число, большее n. Решить задачу используя циклическую конструкцию for.
2.Среди чисел 1, 5 10, 16, 23, ... найти первое число, большее n. Условный оператор не использовать.
Решить задачу используя циклическую конструкцию while.

Вариант 10
1. Известны оценки по физике каждого из 20 учеников класса. Определить среднюю оценку. Решить задачу используя циклическую конструкцию for.
2.Дано число A (> 1). Вывести наибольшее из целых чисел K, для которых сумма 1 + 1/2 + … + 1/K будет меньше A, и саму эту сумму.
Решить задачу используя циклическую конструкцию while.

Вариант 11
1. Известно сопротивление каждого из элементов электрической цепи. Все элементы соединены последовательно. Определить общее сопротивление цепи. Решить задачу используя циклическую конструкцию for.
2.Дано целое число N (> 0). Найти наибольшее целое число K, квадрат которого не превосходит N: K^2 ≤ N. Функцию извлечения квадратного корня не использовать.
Решить задачу используя циклическую конструкцию while.

Вариант 12
1. Известны оценки по физике каждого ученика двух классов. Определить среднюю оценку в каждом классе. Количество учащихся в каждом классе одинаковое. Решить задачу используя циклическую конструкцию for.
2. Выведите на экран для числа 2 его степени от 0 до 20

Решить задачу используя циклическую конструкцию while.

Вариант 13

1. В области 12 районов. Известны количество жителей (в тысячах человек) и площадь (в км2) каждого района. Определить среднюю плотность населения по области в целом. Решить задачу используя циклическую конструкцию for.

2.Мой богатый дядюшка подарил мне один доллар в мой первый день рождения. В каждый день рождения он удваивал свой подарок и прибавлял к нему столько долларов, сколько лет мне исполнилось. Написать программу, указывающую, к какому дню рождения подарок превысит 100$.

Решить задачу используя циклическую конструкцию while.

Вариант 14

1. Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить, сколько клеток будет через 3, 6, 9, ..., 24 часа, если первоначально была одна амеба. Решить задачу используя циклическую конструкцию for.

2. Вывести таблицу значений функции y = -0.5x + x. Значения аргумента (x) задаются минимумом, максимумом и шагом. Например, если минимум задан как 1, максимум равен 3, а шаг 0.5. То надо вывести на экран изменение x от 1 до 3 с шагом 0.5 (1, 1.5, 2, 2.5, 3) и значения функции (y) при каждом значении x.

Решить задачу используя циклическую конструкцию while.

Вариант 15

1. Начав тренировки, лыжник в первый день пробежал 10 км. Каждый следующий день он увеличивал пробег на 10% от пробега предыдущего дня. Определить:

а) пробег лыжника за второй, третий, ..., десятый день тренировок;

б) какой суммарный путь он пробежал за первые 7 дней тренировок.

Решить задачу используя циклическую конструкцию for.

2. Найти сумму и произведение цифр, введенного целого числа. Например, если введено число 325, то сумма его цифр равна 10 (3+2+5), а произведение 30 (3*2*5).
Решить задачу используя циклическую конструкцию while.

Задание 4

Вариант 1

1. Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры. Найти максимальный элемент. Вывести массив на экран в обратном порядке.
2. В массиве действительных чисел все нулевые элементы заменить на среднее арифметическое всех элементов массива.

Вариант 2

1. Дан одномерный массив, состоящий из N целочисленных элементов. Ввести массив с клавиатуры. Найти минимальный элемент. Вывести индекс минимального элемента на экран.
2. Дан массив целых чисел. Переписать все положительные элементы во второй массив, а остальные - в третий.

Вариант 3

1. В одномерном числовом массиве D длиной n вычислить сумму элементов с нечетными индексами. Вывести на экран массив D, полученную сумму.
2. Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

Вариант 4

1. Дан массив целых чисел. Найти максимальный элемент массива и его порядковый номер.
2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Вариант 5

1. Дан одномерный массив из 10 целых чисел. Вывести пары отрицательных чисел, стоящих рядом.
2. Дан целочисленный массив размера 10. Создать новый массив, удалив все одинаковые элементы, оставив их 1 раз.

Вариант 6

1. Дан одномерный массив из 10 целых чисел. Найти максимальный элемент и сравнить с ним остальные элементы. Вывести количество меньших максимального и больших максимального элемента.
2. Одномерный массив из 10-и целых чисел заполнить с клавиатуры, определить сумму тех чисел, которые >5.

Вариант 7

1. Дан массив целых чисел. Найти сумму элементов с четными номерами и произведение элементов с нечетными номерами. Вывести сумму и произведение.
2. Переставить в одномерном массиве минимальный элемент и максимальный.

Вариант 8

1. Найдите сумму и произведение элементов списка. Результаты вывести на экран.
2. В массиве действительных чисел все нулевые элементы заменить на среднее арифметическое всех элементов массива.

Вариант 9

1. Дан одномерный массив, состоящий из N вещественных элементов. Ввести массив с клавиатуры. Найти и вывести минимальный по модулю элемент. Вывести массив на экран в обратном порядке.

2. Даны массивы A и B одинакового размера 10. Вывести исходные массивы. Поменять местами их содержимое и вывести в начале элементы преобразованного массива A, а затем — элементы преобразованного массива B.

Вариант 10

1. Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран это значение, иначе сообщение об их отсутствии.

2. Дан одномерный массив из 15 элементов. Элементам массива меньше 10 присвоить нулевые значения, а элементам больше 20 присвоить 1. Вывести на экран монитора первоначальный и преобразованный массивы в строчку.

Вариант 11

1 Найти наибольший элемент списка, который делиться на 2 без остатка и вывести его на экран.

2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из четных чисел исходного массива, меньше 10, или сообщить, что таких чисел нет. Полученный массив вывести в порядке возрастания элементов.

Вариант 12

1. Найти наименьший нечетный элемент списка и вывести его на экран.

2. Даны массивы A и B одинакового размера 10. Поменять местами их содержимое и вывести вначале элементы преобразованного массива A, а затем — элементы преобразованного массива B.

Вариант 13

1. Дан одномерный массив целых чисел. Проверить, есть ли в нем одинаковые элементы. Вывести эти элементы и их индексы.
2. Дан одномерный массив из 8 элементов. Заменить все элементы массива меньшие 15 их удвоенными значениями. Вывести на экран монитора преобразованный массив.

Вариант 14

1. Найти максимальный элемент численного массива и поменять его местами с минимальным.
2. Программа заполняет одномерный массив из 10 целых чисел числами, считанными с клавиатуры. Определить среднее арифметическое всех чисел массива. Заменить элементы массива большие среднего арифметического на 1.

Вариант 15

1. Определите, есть ли в списке повторяющиеся элементы, если да, то вывести на экран эти значения.
2. Дан одномерный массив целого типа. Получить другой массив, состоящий только из нечетных чисел исходного массива или сообщить, что таких чисел нет. Полученный массив вывести в порядке убывания элементов.

Вопросы и задания для защиты работы

1. Каковы основные особенности структуры программы на Python?
2. Опишите правила именования переменных, а также объясните отличия объявления переменных в Python от других языков программирования.
3. Что такое инструкции в Python?
4. Опишите особенности использования функций print() и input().

5. Каков синтаксис организации ветвления алгоритма программы?

6. Как организуются циклы в Python? Перечислите и опишите основные способы.

7. Как создать пользовательскую функцию и вызвать ее в теле программы?

8. Что такое модули? Перечислите основные модули стандартной библиотеки Python.

9. Как организовать работу с файлами?

10. Что такое исключения? Каковы способы их обработки?