**

Колледж космического машиностроения и технологии

ОТЧЕТ

По учебной практике УП.01.01 Разработка программных модулей программного обеспечения для компьютерных систем

специальность 09.02.03 Программирование в компьютерных системах

Выполнил студент: Кузнецов Матвей Сергеевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Проверил преподаватель: Гусятинер Леонид Борисович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(оценка)

Королев, 2020

Оглавление

[Практическое занятие №2. 3](#_Toc57279879)

[Практическое занятие №3. 4](#_Toc57279880)

[Создание файла с кодом. 4](#_Toc57279881)

[Запуск. 4](#_Toc57279882)

[Import. 4](#_Toc57279883)

[Reload. 4](#_Toc57279884)

[Отработка ошибок. 4](#_Toc57279885)

[Практическое занятие №4. 6](#_Toc57279886)

[Задачи на Begin. 6](#_Toc57279887)

[Задачи на Boolean. 6](#_Toc57279888)

[Задачи на Case. 6](#_Toc57279889)

[Задачи на If. 8](#_Toc57279890)

[Блок-схемы для задач к практическому занятию №4 9](#_Toc57279891)

[Блок-схема для задачи boolean1: 9](#_Toc57279892)

[Блок-схема для задачи boolean2: 10](#_Toc57279893)

[Практическое занятие №5. 11](#_Toc57279894)

[Задачи на For. 11](#_Toc57279895)

[Задачи на While. 12](#_Toc57279896)

[Блок-схемы для задач к практическому занятию №5 14](#_Toc57279897)

[Блок-схема для задачи for1: 14](#_Toc57279898)

[Блок-схема для задачи while5: 15](#_Toc57279899)

[Практическое занятие №6. 16](#_Toc57279900)

[Complex. 16](#_Toc57279901)

[Decimal. 16](#_Toc57279902)

[Fraction. 17](#_Toc57279903)

# Практическое занятие №2.

Материал подготовили студенты группы П2-18:

1| Короленко Иван Романович

2| Кузнецов Матвей Сергеевич

3| Слесарев Алексей Михайлович

Установка Python будет производиться в Visual Studio 2019.

1. Запускаем Visual Studio 2019.

2. Нажимаем на кнопку "Создание проекта";

3. В появившемся/открывшемся окне, прокручиваем до конца списка; (Будет надпись “Не нашли то, что искали?”)

4. Нажимаем на кнопку "Установка других средств и компонентов";

5. В открывшемся установщике выбираем раздел "Рабочие нагрузки";

6. После выбора раздела "Рабочие нагрузки", ищем в "Веб-разработки и облако" (Разработка на Python) и выбираем его;

7. После выбора (Разработка на Python), жмем на кнопку "Изменить"; (Ждем окончание загрузки)

8. После завершение установке, автоматически открывается Visual Studio, в котором можно запустить среду разработки Python.



(Приложение 1)

# Практическое занятие №3.

Материал подготовили студенты группы П2-18:

1| Короленко Иван Романович

2| Кузнецов Матвей Сергеевич

3| Слесарев Алексей Михайлович

## Создание файла с кодом.

Код исходной программы:  
print('Hello, world!') **(sample.py)**

## Запуск.

Для того, чтобы открыть существующий файл с кодом, нужно:  
1. В папке с нужным файлом нажать сочетание клавиш Shift + ПКМ.

2. В появившемся диалоговом окне нажать “Открыть окно PowerShell здесь”. (см. приложение 1)

3. Ввести команду “python название-файла”.

4. Получить результат выполнения программы (см. приложение 2).

## Import.

Чтобы подключить определенную библиотеку, нужно:  
1. В папке с нужным файлом нажать сочетание клавиш Shift + ПКМ.

2. В появившемся диалоговом окне нажать “Открыть окно PowerShell здесь”. (см. приложение 1)

3. Ввести в консоль команду “python”.

4. Ввести в консоль команду “import название-библиотеки”. (см. приложение 3)

## Reload.

Чтобы перезагрузить определенное нечто, требуется:  
1. В папке с нужным файлом нажать сочетание клавиш Shift + ПКМ.

2. В появившемся диалоговом окне нажать “Открыть окно PowerShell здесь”. (см. приложение 1)

3. Ввести в консоль команду “python”.  
4. Импортировать нужный файл. (см. приложение 4)  
5. Перезаписать исходный файл.   
Код измененной программы:

print('Hello!!!!!') **(sample.py)**

6. Написать команду “reload sample”.

## Отработка ошибок.

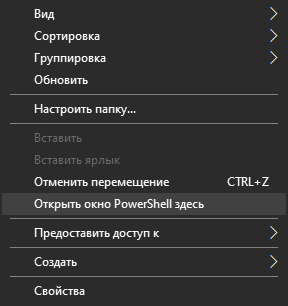
Примеры часто встречающихся ошибок:



Данная ошибка говорит о не прописанной команде “python”.



Данная ошибка говорит о том, что интерпретатор не может найти переменную с указанным именем.



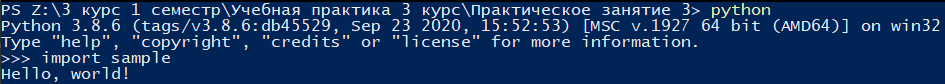
(Приложение 1)



(Приложение 2)



(Приложение 3)



(Приложение 4)

# Практическое занятие №4.

## Задачи на Begin.

**Задача 1: ( begin1.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дана сторона квадрата a. Найти его периметр P = 4\*a.

x = int(input("Введите сторону квадрата: "))

a = x \* 4

print("Периметр квадрата равен", a)

**Задача 2: ( begin2.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дана сторона квадрата a. Найти его площадь S = a^2.

x = int(input("Введите сторону квадрата: "))

a = x \* x

print("Площадь квадрата равна", a)

## Задачи на Boolean.

**Задача 1: ( boolean1.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано целое число A.

#Проверить истинность высказывания: «Число A является положительным».

x = int(input("Введите число A: "))

res = x > 0

print(res)

**Задача 2: ( boolean2.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано целое число A.

#Проверить истинность высказывания: «Число A является нечетным».

x = int(input("Введите число A: "))

res = x % 2 == 0

print(res)

## Задачи на Case.

**Задача 1: ( case1.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано целое число в диапазоне 1–7. Вывести строку — название дня недели,

#соответствующее данному числу (1 — «понедельник», 2 — «вторник» и т. д.).

x = int(input("Введите число в диапазоне 1-7: "))

if 0 < x < 8:

if x == 1:

print("Это понедельник!")

elif x == 2:

print("Это вторник!")

elif x == 3:

print("Это среда!")

elif x == 4:

print("Это четверг!")

elif x == 5:

print("Это пятница!")

elif x == 6:

print("Это суббота!")

elif x == 7:

print("Это воскресенье!")

else:

print("Число не в диапазоне 1-7!")

**Задача 2: ( case2.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано целое число K. Вывести строку-описание оценки, соответствующей числу K

#(1 — «плохо», 2 — «неудовлетворительно», 3 — «удовлетворительно», 4 — «хорошо», 5 — «отлично»).

#Если K не лежит в диапазоне 1–5, то вывести строку «ошибка».

x = int(input("Введите K в диапазоне 1-5: "))

if 0 < x < 6:

if x == 1:

print("Плохо")

elif x == 2:

print("Неудовлетворительно")

elif x == 3:

print("Удовретворительно")

elif x == 4:

print("Хорошо")

elif x == 5:

print("Отлично")

else:

print("Ошибка")

**Задача 3: ( case3.py )**

#Выполнил Кузнецов М. С. П2-18.

#Элементы окружности пронумерованы следующим образом:

#1 — радиус R, 2 — диаметр D = 2·R, 3 — длина L = 2·π·R, 4 — площадь круга S = π·R2.

#Дан номер одного из этих элементов и его значение.

#Вывести значения остальных элементов данной окружности (в том же порядке).

#В качестве значения π использовать 3.14.

R = float(input("Введите радиус окружности: "))

Pi = float(3.14)

a = int(input("Что хотите вывести?\n1 - Радиус(R)\n2 - Диаметр(D)\n3 - Длина окружности(L)\n4 - Площадь круга(S)\n> "))

if a == 1:

print("Радиус =", R)

elif a == 2:

D = 2 \* R

print("Диаметр =", D)

elif a == 3:

L = 2 \* Pi \* R

print("Длина окружности =", L)

elif a == 4:

S = Pi \* (R \* 2)

print("Площадь круга =", S)

## Задачи на If.

**Задача 1: ( if1.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано целое число. Если оно является положительным, то прибавить к нему 1;

#в противном случае не изменять его. Вывести полученное число.

x = int(input("Введите число: "))

if x > 0:

x = x + 1

print("Число положительное.", x)

else:

print("Число отрицательное или ноль.", x)

**Задача 2: ( if2.py )**#Выполнил Кузнецов М.С. П2-18.

#Дано целое число. Если оно является положительным, то прибавить к нему 1;

#в противном случае вычесть из него 2. Вывести полученное число.

x = int(input("Введите число: "))

if x > 0:

x = x + 1

print("Число положительное.", x)

else:

x = x - 2

print("Число отрицательное или ноль.", x)

**Задача 3: ( if3.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано целое число. Если оно является положительным, то прибавить к нему 1;

#если отрицательным, то вычесть из него 2;

#если нулевым, то заменить его на 10. Вывести полученное число

x = int(input("Введите число: "))

if x > 0:

x = x + 1

print("Число положительное.", x)

elif x < 0:

x = x - 2

print("Число отрицательное.", x)

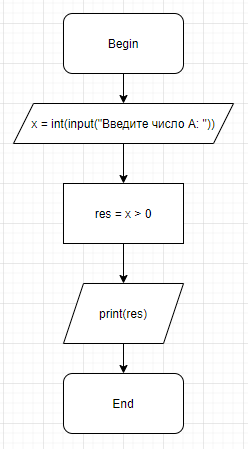
else:

x = 10

print("Введенное число - ноль.", x)

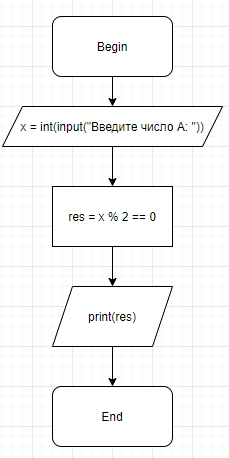
# Блок-схемы для задач к практическому занятию №4

## Блок-схема для задачи boolean1:



(Рис. 1)

Блок-схема для задачи boolean2:



(Рис. 2)

# Практическое занятие №5.

## Задачи на For.

**Задача 1: ( for1.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано вещественное число — цена 1 кг конфет.

#Вывести стоимость 1, 2, …, 10 кг конфет.

x = int(input("Введите цену 1кг конфет: "))

n = 0

for kg in range(1, 11):

n = x + n

print('>', n, 'стоит', kg, 'кг конфет')

kg += 1

**Задача 2: ( for2.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано вещественное число A и целое число N (> 0).

#Найти A в степени N:

A = int(input("Введите A: "))

N = int(input("Введите степень, в которую должно быть возведено число A: "))

n = int(1)

for i in range(1, N + 1):

x = A \*\* n

print(A, "в степени", n, "будет", x)

n += 1

**Задача 3: ( for3.py )**

#Выполнил Кузнецов М. С. П2-18.

#Даны два целых числа A и B (A < B).

#Найти сумму всех целых чисел от A до B включительно.

A = int(input("Введите A: "))

B = int(input("Введите B: "))

x = int(0)

if A < B:

for i in range(A, B + 1):

x = x + i

print("Сумма всех целых чисел от", A, "до", B, "будет", x)

else:

print("A !< B. Exit program...")

**Задача 4: ( for4.py )**

#Выполнил Кузнецов М. С. П2-18.

#Даны два целых числа A и B (A < B).

#Найти произведение всех целых чисел от A до B включительно.

A = int(input("Введите A: "))

B = int(input("Введите B: "))

x = int(1)

if A < B:

for i in range(A, B + 1):

x = x \* i

print("Произведение всех целых чисел от", A, "до", B, "будет", x)

else:

print("A !< B. Exit program...")

**Задача 5: ( for5.py )**

#Выполнил Кузнецов М. С. П2-18.

#Даны два целых числа A и B (A < B).

#Найти сумму квадратов всех целых чисел от A до B включительно.

A = int(input("Введите A: "))

B = int(input("Введите B: "))

x = int(1)

if A < B:

for i in range(A, B + 1):

x = x + (i \*\* 2)

x -= 1

print("Сумма квадратов всех целых чисел от", A, "до", B, "будет", x)

else:

print("A !< B. Exit program...")

## Задачи на While.

**Задача 1: ( while1.py )**

#Выполнил Кузнецов М. С. П2-18.

#По данному целому числу N распечатайте все квадраты натуральных чисел,

#не превосходящие N, в порядке возрастания.

n = int(input("Введите N:"))

i = 1

while i \*\* 2 <= n:

print(i \*\* 2)

i += 1

**Задача 2: ( while2.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано целое число, не меньшее 2.

#Выведите его наименьший натуральный делитель, отличный от 1.

n = int(input("Введите целое число: "))

i = 2

while n % i != 0:

i += 1

print(i)

**Задача 3: ( while3.py )**

#Выполнил Кузнецов М. С. П2-18.

#По данному натуральному числу N найдите наибольшую целую степень двойки, не превосходящую N.

#Выведите показатель степени и саму степень.

#Операцией возведения в степень пользоваться нельзя!

n = int(input("Введите натуральное число N:"))

x = 2

y = 1

while x <= n:

x \*= 2

y += 1

print(y - 1, x // 2)

**Задача 4: ( while4.py )**

#Выполнил Кузнецов М. С. П2-18.

#Дано целое число N (> 0). Если оно является степенью числа 3,

#то вывести true, если не является — вывести false.

n = int(input("Введите целое число N( >0 ):"))

s = 3

r = 1

while (r \* s <= n):

r \*= s

if r == n:

print("True")

else:

print("False")

**Задача 5: ( while5.py )**

#Выполнил Кузнецов М. С. П2-18.

#Даны положительные числа A и B (A > B).

#На отрезке длины A размещено максимально возможное количество отрезков длины B (без наложений).

#Не используя операции умножения и деления, найти количество отрезков B, размещенных на отрезке A.

a = int(input("Введите А: "))

b = int(input("Введите B: "))

i = 1

while (a > b):

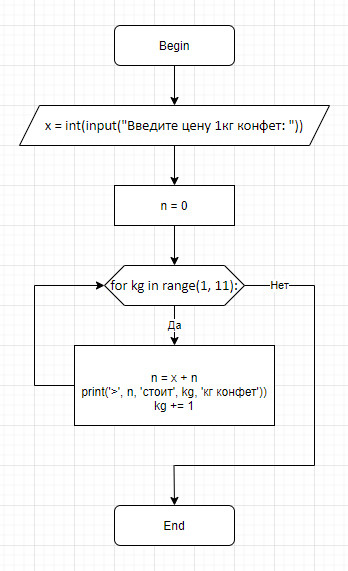
a -= b

i += 1

print("Количество возможных размещенных отрезков B -", i)

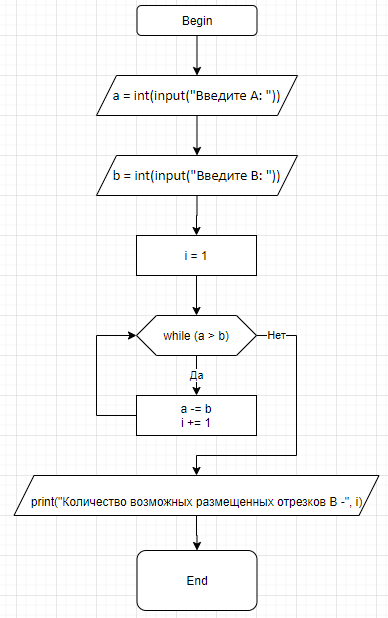
# Блок-схемы для задач к практическому занятию №5

## Блок-схема для задачи for1:



(Рис. 1)

## Блок-схема для задачи while5:



(Рис. 2)

# Практическое занятие №6.

## Complex.

#Выполнил Кузнецов М. С. П2-18.

#Операции с комплексными числами

a = complex(3, 4) #Первое число - действительное, второе - мнимое.

print('Обычное комплексное число a:', a)

b = complex(6, 4)

print('Обычное комплексное число b:', b)

c = a + b

print('Результат сложения этих комплексных чисел:', c)

print('Мнимая часть комплексного числа а:', a.imag)

print('Действительная часть комплексного числа b:', b.real)

print('Проверим на равенство комплексные числа a и b:', a == b)

print('Модуль комплексного числа a:', abs(a))

print('Возведение в четвертую степень комплексного числа b', pow(b, 4))  
**(complex.py)**

## Decimal.

# Выполнил: Короленко Иван Романович

# Группа: П2-18

# Decimal- вычисления с заданной точностью

3.3 + 4.1 # Ответом будет 7,33333, в нашем же случае 7.4

from decimal import Decimal

a = Decimal('3.3') + Decimal('4.1')

print(a)

3.3 + 4.1 == 7.4 # должно быть True, но...

# Но с числами типа Decimal все верно:

c = Decimal('3.3') + Decimal('4.1')

print(c)

# С помощью дополнительных знаков мы можем определить, сколько будет символов в дробной части числа:

number = Decimal("0.10")

number = 3 \* number

print(number) # 0.30

# Однако нельзя смешивать в операциях дробные числа float и Decimal:

# number = Decimal("0.1")

# number = number + 0.1 # здесь возникнет ошибка

# Округление чисел

# Объекты Decimal имеют метод quantize(),который позволяет округлять числа.

# В этот метод в качестве первого аргумента передается также объект Decimal,

# который указывает формат округления числа:

number = Decimal("0.444")

number = number.quantize(Decimal("1.00"))

print(number) # 0.44

number = Decimal("0.555678")

print(number.quantize(Decimal("1.00"))) # 0.56

number = Decimal("0.999")

print(number.quantize(Decimal("1.00"))) # 1.00

# Как отделить рубли от копеек в вещественных числах?

# Decimal вместо float:

x = Decimal('40.80')

a = int(x)

b = int(100 \* (x - a))

print(a, b) # => 40 80

**(decimal.py)**

## Fraction.

# Cлесарев А.М. Задание разбор модуля Fraction

# Модуль fractions

from fractions import Fraction

from decimal import Decimal

import math

#Fraction.limit\_denominator(max\_denominator=1000000)- ближайшее

#рациональное число со знаменателем не больше данного.

a = Fraction(3.1415) # Целое и вещественное число, так же можно преобразовать в обыкновенную дробь

print (a)

print (a.limit\_denominator())

# по умолчанию numerator=0, denominator=1

print (Fraction())

# равносильно Fraction(1, 2)

print (Fraction(numerator=1, denominator=2))

#Если указанные числитель и знаменатель имеют общие делители,

#то перед созданием рационального числа они будут сокращены

print (Fraction(8, 16), Fraction(15, 30))

#Если указанные числитель и знаменатель имеют общие делители,

#то перед созданием рационального числа они будут сокращены

print (Fraction(3, Fraction(1, 2)))

#Округляет до ближайшего четного числа.

print (Fraction('1/2').\_\_round\_\_())

#создает обыкновенную дробь, которая является точным представлением

#десятичной дроби указанной в dec, где dec – это экземпляр класса decimal.Decimal

print (Fraction.from\_decimal(Decimal('0.7')))

#принимает flt – число типа float и возвращает обыкновенную дробь отношение числителя

#к знаменателю которой максимально приближается к значению flt.

print (Fraction.from\_float(0.5))

#fractions.gcd(a, b) - наибольший общий делитель чисел a и b.

print (math.gcd(1000, 3))

print (math.gcd(4, 6))  
#Использовался сайт https://pyprog.pro/python/st\_lib/fractions.html

**(fraction.py)**

# Практическое занятие №7

**Задание 1. (1.py)**

**#Выполнил: Короленко И.Р.**

**#Группы: П2-18**

**#С клавиатуры вводятся строки, последовательность заканчивается точкой.**

**#Выведите буквы введенных слов в верхнем регистре, разделяя их пробелами.**

**#Примечание: узнайте, как работает строковый метод upper().**

**word = input()**

**mass = []**

**while word != '.':**

**mass.append(word)**

**word = input()**

**for item in mass:**

**output = ''**

**for letter in item:**

**output +=letter.upper()**

**print(\*output)**

**Задание 2. (2.py)**

**#Выполнил: Короленко И.Р.**

**#Группы: П2-18**

**words = input()**

**Wrd = words.split()**

**Wrd1= '\_'.join(words)**

**print(Wrd1)**

**#Второй вариант, с решения степика)**

**print(input().replace(' ', '\_'))**

**Задание 3. (3.py)**

**#Выполнил: Короленко И.Р.**

**#Группы: П2-18**

**s = input()**

**s = s.replace("."," ")**

**s1 = s.split()**

**s1 = [int(x) for x in s1]**

**print(s)**

**print(sum(s1))**

**Задание 4. (4.py)**

**#Слесарев А.М.**

**# Программист логирует программу, чтобы хорошо знать, как она себя ведет**

**#(эта весьма распространенная и важная практика).**

**#Он использует разные типы сообщений для вывода ошибок(error),**

**#предупреждений (warning), информации (info) или подробного описания (verbose). Сообщения отличаются по внешнему виду. Назовем модификаторами такие символы, которые отличают сообщения друг от друга, позволяя программисту понять, к какому из типов относится сообщения. Модификаторы состоят из двух одинаковых символов и записываются по разу в начале и в конце строки.**

**#@@ обозначает ошибку**

**#!! обозначает предупреждение**

**#// обозначает информационное сообщение**

**#\*\* обозначает подробное сообщение**

**#Напишите программу, которая принимает строки до точки и выводит,**

**#какого типа это сообщение. Если сообщение не содержит модификаторов,**

**#проигнорируйте его.**

**err = list(input())**

**while err[0] != "." and a :**

**a = len(err)**

**if err[0] and err[1] and err[a-1] and err[a-2] == "@":**

**print('ошибка')**

**elif err[0] and err[1] and err[a-1] and err[a-2] == "!":**

**print('предупреждение')**

**elif err[0] and err[1] and err[a-1] and err[a-2] == "/":**

**print('информация')**

**elif err[0] and err[1] and err[a-1] and err[a-2] == "\*":**

**print('подробное сообщение')**

**err = list(input())**