

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: Пихтов Виталий Алексеевич

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

Проверил преподаватель: Гусятинер Леонид Борисович

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

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

Королев, 2020

Практическое занятие №3.

Материал подготовили студенты группы П2-18:

1| Пихтов Виталий Алексеевич

Создание файла с кодом.

Код исходной программы:  
print('Hello, world') **(test.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') **(test.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 Boolean.

**Задача 1: ( begin1.py )**

#Выполнил Пихтов В. А. П2-18.

#Задача "Площадь прямоугольного треугольника".

#Напишите программу, которая считывает длины двух катетов в прямоугольном треугольнике и выводит его площадь. Каждое число записано в отдельной строке.

a = int(input())

b = int(input())

print(a \* b / 2)

**Задача 2: ( begin2.py )**

#Выполнил Пихтов В. А. П2-18.

#Ввод и вывод данных. Задача «Дележ яблок».

#n школьников делят k яблок поровну, неделящийся остаток остается в корзинке. Сколько яблок достанется каждому школьнику? Сколько яблок останется в корзинке? Программа получает на вход числа n и k и должна вывести искомое количество яблок (два числа).

n = int(input())

k = int(input())

print(k // n)

print(k % n)

Задачи на Case.

**Задача 1: ( case1.py )**

#Выполнил Пихтов В. А. П2-18.

#Пятилетний Петя учит цвета радуги. Напишите программу, которая принимает на вход натуральное число n и печатает первые n цветов радуги с большой буквы. При этом, если n > 7, программа должна ответить "Радуга состоит только из семи цветов".

RainbowSix = ("Красный", "Оранжевый", "Желтый", "Зеленый", "Голубой", "Синий", "Фиолетовый")

a = int(input())

if a <= 7:

for b in range(a):

print(RainbowSix[b])

elif a > 7:

print("Радуга состоит только из семи цветов")

**Задача 2: ( case2.py )**

#Выполнил Пихтов В. А. П2-18.

#Напишите программу, которая печатает названия месяцев до месяца n (n натуральное, вводится с клавиатуры) согласно шаблону, представленному в примере. Если n > 12, программа должна вывести "Error".

a = ["Первый месяц - январь", "Второй месяц - февраль", "Третий месяц - март", "Четвертый месяц - апрель", "Пятый месяц - май", "Шестой месяц - июнь", "Седьмой месяц - июль", "Восьмой месяц - август", "Девятый месяц - сентябрь", "Десятый месяц - октябрь", "Одиннадцатый месяц - ноябрь", "Двенадцатый месяц - декабрь",]

b = int(input())

if b <= 12:

for i in range(0, b):

print(a[i])

if b > 12:

print("Error")

**Задача 3: ( case3.py )**

#Выполнил Пихтов В. А. П2-18.

#Некий гражданин NN пришел в магазин с n рублями. Просмотрев свой список покупок, он считает стоимость всех товаров и соотносит ее с той суммой, что у него есть. Если ему хватает денег, то выведите "Покупает", иначе "Не покупает".

#Формат ввода: на первой строке вводится натуральное число – количество денег, на второй строке – количество товаров n, на следующих n строках – цена каждого из товаров.

#Формат вывода: строки "Покупает" или "Не покупает".money = int(input())

count = int(input())

counter = 0

for n in range(count):

a = int(input())

counter = counter + a

if counter <= money:

b = "Покупает"

else:

b = "Не покупает"

print(b)

Задачи на If.

**Задача 1: ( if1.py )**

#Выполнил Пихтов В. А. П2-18.

#Напишите функцию fib(n), которая по данному целому неотрицательному n возвращает n-e число Фибоначчи. В этой задаче нельзя использовать циклы — используйте рекурсию.

def fib(n):

if (n == 1) or (n == 2):

return 1

else:

return fib(n - 1) + fib(n - 2)

n = int(input())

print(fib(n))

**Задача 2: ( if2.py )**#Выполнил Пихтов В. А. П2-18.

#Выведите все четные элементы списка. При этом используйте цикл for, перебирающий элементы списка, а не их индексы!

spisok = input().split()

for n in spisok:

if int(n) % 2 == 0:

print(n, end = ' ')

**Задача 3: ( if3.py )**

#Выполнил Пихтов В. А. П2-18.

#Дан список чисел. Выведите все элементы списка, которые больше предыдущего элемента.

spisok = [int(n) for n in input().split()]

for n in range(1, len(spisok)):

if spisok[n] > spisok[n - 1]:

print(spisok[n])

**Задача 1: ( boolean1.py )**

#Выполнил Пихтов В. А. П2-18.

#Пользователь вводит числа по одному в строке, последовательность оканчивается числом 0. Выведите сумму чисел, введенных до нуля.

summa = 0

a = int(input())

while a != 0:

summa += a

a = int(input())

print(summa)

**Задача 2: ( boolean2.py )**

#Выполнил Пихтов В. А. П2-18.

#Минимальный делитель.

#Программа получает на вход натуральное число n > 1. Выведите минимальный делитель этого числа, отличный от единицы.

a = int(input())

n = 2

while a % n != 0:

n += 1

print(n)

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 Пихтов В. А. П2-18.

#Цикл с условием.

#С клавиатуры вводятся числа a и b. Необходимо вывести те числа на полуинтервале [a; b) или [b; a), которые являются четными и дают остаток 1 при делении на 7.

#Примечание: запись [a; b) означает промежуток из чисел от a включительно до b невключительно. В языке Python это соответствует записи range(a, b). Аналогично, промежуток [b; a) соответствует записи range(b, a).

a = int(input())

b = int(input())

if a < b:

for c in range(a, b):

if c % 7 == 1 and c % 2 == 0:

print(c)

else:

for c in range(b ,a):

if c % 7 == 1 and c % 2 == 0:

print(c)

**Задача 2: ( for2.py )**

#Выполнил Пихтов В. А. П2-18.

#Посчитайте сумму всех чисел на полуинтервале [a; b) или [b; a), в зависимости от того, какое число больше.

a = int(input())

b = int(input())

sum = 0

if a < b:

for c in range(a, b):

sum += c

else:

for c in range(b, a):

sum += c

print(sum)

**Задача 3: ( for3.py )**

#Выполнил Пихтов В. А. П2-18.

#Вы уже знакомы с термином "факториал", однако предыдущее наше решение не идеально – оно считает факториал только одного числа. Модифицируйте вашу программу из задачи "18!" таким образом, чтобы она принимала на вход натуральное число n и печатала значение выражения n!

a = int(input())

factorial = 1

for n in range(2, a+1):

factorial \*= n

print(factorial)

**Задача 4: ( for4.py )**

#Выполнил Пихтов В. А. П2-18.

#Дан список из чисел и индекс элемента в списке k. Удалите из списка элемент с индексом k, сдвинув влево все элементы, стоящие правее элемента с индексом k.

Программа получает на вход список, затем число k. Программа сдвигает все элементы, а после этого удаляет последний элемент списка при помощи метода pop() без параметров.

#Программа должна осуществлять сдвиг непосредственно в списке, а не делать это при выводе элементов. Также нельзя использовать дополнительный список. Также не следует использовать метод pop(k) с параметром.

spisok = [int(s) for s in input().split()]

a = int(input())

for n in range(a + 1, len(spisok)):

spisok[n - 1] = spisok[n]

spisok.pop()

print(' '.join([str(n) for n in spisok]))

**Задача 5: ( for5.py )**

#Выполнил Пихтов В. А. П2-18.

#Выведите все элементы списка с четными индексами (то есть A[0], A[2], A[4], ...).

spisok = input().split()

for index in range(0, len(spisok), 2):

print(spisok[index])

Задачи на While.

**Задача 1: ( while1.py )**

#Выполнил Пихтов В. А. П2-18.

#По данному целому числу N распечатайте все квадраты натуральных чисел, не превосходящие N, в порядке возрастания.

b = 1

a = int(input())

while b \*\* 2 <= a:

print(b \*\* 2)

b += 1

**Задача 2: ( while2.py )**

#Выполнил Пихтов В. А. П2-18.

#Дано целое число, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1.

b = 2

a = int(input())

while a % b != 0:

b += 1

print(b)

**Задача 3: ( while3.py )**

#Выполнил Пихтов В. А. 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!

a = int(input())

b = 1

c = 2

while c <= a:

c \*= 2

b += 1

print(b - 1, c // 2)

**Задача 4: ( while4.py )**

#Выполнил Пихтов В. А. П2-18.

#В первый день спортсмен пробежал x километров, а затем он каждый день увеличивал пробег на 10% от предыдущего значения. По данному числу y определите номер дня, на который пробег спортсмена составит не менее y километров.

#Программа получает на вход действительные числа x и y и должна вывести одно натуральное число.

x = int(input())

y = int(input())

n = 1

while y > x:

x \*= 1.1

n += 1

print(n)

**Задача 5: ( while5.py )**

#Выполнил Пихтов В. А. П2-18.

#Программа получает на вход последовательность целых неотрицательных чисел, каждое число записано в отдельной строке. Последовательность завершается числом 0, при считывании которого программа должна закончить свою работу и вывести количество членов последовательности (не считая завершающего числа 0). Числа, следующие за числом 0, считывать не нужно.

dlinna = 0

while int(input()) != 0:

dlinna += 1

print(dlinna)

Блок-схемы для задач к практическому занятию №5

Блок-схема для задачи while1:



(Рис. 1)

Блок-схема для задачи while5:



(Рис. 2)

Практическое занятие №6.

Decimal.

# Выполнил: Пихтов Виталий Алексеевич

# Группа: П2-18

# Decimal- это вычисления с заданной точностью

2.4 + 3.8 # Ответом будет 6.199999999999999, в нашем же случае 6.2

from decimal import Decimal

a = Decimal('2.4') + Decimal('3.8')

print(a)

2.4 + 3.8 == 6.2 # должно быть True, но с числами типа Decimal все верно:

c = Decimal('2.4') + Decimal('3.8')

print(c)

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

from decimal import Decimal

num = Decimal("0.20")

num = 3 \* num

print(num) # 0.60

# 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("0.888")

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

print(number) # 0.89

number = Decimal("0.1112345")

print(number.quantize(Decimal("1.00"))) # 0.11

number = Decimal("0.999")

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

# Как отделить целое от дроби в вещественных числах ?

# Decimal вместо float:

a = Decimal('580.140')

b = int(a)

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

print(b, c) # => 580 14

**(decimal1.py)**

#Использовался сайт

<https://metanit.com/python/tutorial/6.4.php>

Fraction.

# Пихтов Виталий Алексеевич

# 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) # 7074029114692207/2251799813685248

print(a.limit\_denominator()) # 6283/2000

# 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>

**(fraction1.py)**

Практическое занятие №7.

Техника работы со строками

# выполнил: Пихтов Виталий Алексеевич

# Задание 1. <https://stepik.org/lesson/201702/step/5?unit=175778>

# С клавиатуры вводятся строки, последовательность заканчивается точкой.

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

# (spiski1.py)

a = input()

List = []

while a != '.':

List.append(a)

a = input()

for n in List:

b = ''

for l in n:

b += l.upper() + ' '

print(b)

# выполнил: Пихтов Виталий Алексеевич

#Задание 2. <https://stepik.org/lesson/201702/step/8?unit=175778>

# Известно, что для логина часто не разрешается использовать строки содержащие пробелы.

# Но пользователю нашего сервиса особенно понравилась какая-то строка.

# Замените пробелы в строке на символы нижнего подчеркивания, чтобы строка

# могла сгодиться для логина. Если строка состоит из одного слова, менять ничего не нужно.

# (spiski2.py)

a = input()

b = a.replace(" ", "\_")

print(b)

# выполнил: Пихтов Виталий Алексеевич

#Задание 3. <https://stepik.org/lesson/201702/step/9?unit=175778>

# Уберите точки из введенного IP-адреса. Выведите сначала четыре числа через пробел, а затем сумму получившихся чисел.

# (spiski3.py)

l = []

a = input().split(".")

for n in a:

l.append(int(n))

a = " ".join(a)

print(a, "\n", sum(l))

# выполнил: Пихтов Виталий Алексеевич

#Задание 4. <https://stepik.org/lesson/201702/step/14?unit=175778>

#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

# (spiski4.py)

a=input()

while a!='.':

if '@@' in a:

print('ошибка')

elif '!!' in a:

print('предупреждение')

elif '//' in a:

print('информация')

elif '\*\*' in a:

print('подробное сообщение')

a=input()

7.2

#1 Форматирование строк (оператор %)

Строки в Python содержат встроенную операцию, доступ к которой можно получить через оператор %. Это позволяет упростить позиционное форматирование. Вот простой пример:

Python

print('Hello, %s' % name)

# Вывод: "Hello, Bob"

print('Hello, %s' % name)

# Вывод: "Hello, Bob"

Существуют другие определители формата, которые позволяют контролировать формат выдачи. Например, возможно конвертировать числа в шестнадцатеричную нотацию или добавлять пробелы для создания хорошо отформатированных таблиц и отчетов. (См Python Docs: ““printf-style String Formatting”.)

Можно использовать определитель формата %x для конвертации значения int в строку и представить его в качестве шестнадцатеричного числа:

Python

errno = 50159747054

print('%x' % errno)

# Вывод: 'badc0ffee'

errno = 50159747054

print('%x' % errno)

# Вывод: 'badc0ffee'

Синтаксис форматирования строки немного меняется. Так как оператор % принимает только один аргумент, нужно обернуть правую часть в кортеж, вот так:

Python

errno = 50159747054

name = 'Bob'

print('Hey %s, there is a 0x%x error!' % (name, errno))

# 'Hey Bob, there is a 0xbadc0ffee error!'

errno = 50159747054

name = 'Bob'

print('Hey %s, there is a 0x%x error!' % (name, errno))

# 'Hey Bob, there is a 0xbadc0ffee error!'

Также возможно сослаться на заменители переменных по имени в строке формата, если вы передадите сопоставление оператору %:

Python

print(

'Hey %(name)s, there is a 0x%(errno)x error!' % {

"name": name, "errno": errno

}

)

# Вывод: 'Hey Bob, there is a 0xbadc0ffee error!'

print(

'Hey %(name)s, there is a 0x%(errno)x error!' % {

"name": name, "errno": errno

}

)

# Вывод: 'Hey Bob, there is a 0xbadc0ffee error!'

Это помогает сделать настройки формата более простым процессом, а также упрощает его модификацию в будущем. Недостаток — это то, что эта техника требует немного больше ручного ввода.

#2 Форматирование строк (str.format)

Python 3 предоставил новый способ форматирования, который был внесен в раннюю версию Python 2.7. Это форматирование строк избавляется от специального синтаксиса оператора % и делает синтаксис для форматирования строк более регулярным. Теперь форматирование обрабатывается вызовом .format() в объекте строки.

Python

print('Hello, {}'.format(name))

# Вывод: 'Hello, Bob'

print('Hello, {}'.format(name))

# Вывод: 'Hello, Bob'

Или можно сослаться на подстановки переменных по имени, и использовать их в том порядке, в котором хочется. Это достаточно мощный способ, так как он позволяет повторно упорядочить порядок отображения без изменения переданных функции format() аргументов:

Python

print(

'Hey {name}, there is a 0x{errno:x} error!'.format(

name=name, errno=errno

)

)

# Вывод: 'Hey Bob, there is a 0xbadc0ffee error!'

print(

'Hey {name}, there is a 0x{errno:x} error!'.format(

name=name, errno=errno

)

)

# Вывод: 'Hey Bob, there is a 0xbadc0ffee error!'

Это также демонстрирует, что синтаксис формата переменной int — это шестнадцатеричная измененная строка.

Простой синтаксис

Синтаксис аналогичен тому, который вы используете в str.format(), но не такой перегруженный. Посмотрите на эту читабельность:

Python

name = "Eric"

age = 74

print(f"Hello, {name}. You are {age}.")

# Вывод: 'Hello, Eric. You are 74.'

name = "Eric"

age = 74

print(f"Hello, {name}. You are {age}.")

# Вывод: 'Hello, Eric. You are 74.'

Произвольные выражения

Так как f-строки оцениваются по мере выражения, можно внести любую или все доступные выражения Python в них. Это позволит делать интересные вещи, например следующее:

Python

print(f"{2 \* 37}")

# Вывод: '74'

1

2

print(f"{2 \* 37}")

# Вывод: '74'

Скорость

Буква f в f-strings может также означать и “fast”. f-строки заметно быстрее чем % и str.format() форматирования, f-строки являются выражениями, которые оцениваются по мере выполнения, а не постоянные значения.