Упражнения по программированию главы 3

```
# coding: utf-8
```

Упражнение 3.1. День недели

```
# Получить число для дня недели.
day = int(input('Введите число (1-7) для дня недели: '))

# Определить название дня недели и показать его.
if day == 1:
    print('Понедельник')
elif day == 2:
    print('Вторник')
elif day == 3:
    print('Среда')
elif day == 4:
    print ('Четверг')
elif day == 5:
    print ('Пятница')
elif day == 6:
    print ('Суббота')
elif day == 7:
    print ('Воскресенье')
else:
    print ('Ошибка: пожалуйста, введите число в диапазоне между 1 и 7.')
```

Упражнение 3.2. Площади прямоугольников

```
# Локальные переменные
lengthA = 0.0
widthA = 0.0
areaA = 0.0
lengthB = 0.0
widthB = 0.0
areaB = 0.0

# Получить длину A
lengthA = float(input('Введите длину A: '))

# Получить ширину A
widthA = float(input('Введите ширину A: '))

# Получить длину B
lengthB = float(input('Введите длину B: '))

# Получить ширину B
widthB = float(input('Введите ширину B: '))
```

```
# Вычислить площадь A
areaA = lengthA * widthA

# Вычислить площадь B
areaB = lengthB * widthB

# Напечатать сравнение площадей
print ('Площадь A:', format(areaA, '.2f'))
print ('Площадь B:', format(areaB, '.2f'))
if  areaA > areaB:
    print ('Площадь A больше площади B.')
elif  areaA < areaB:
    print ('Площадь B больше площади A.')
else:
    print ('Площадь A равна площади B.')
```

Упражнение 3.3. Классификатор возраста

```
# Получить возвраст человека.
age = int(input('Введите возраст: '))

# Определить, является ли этот человек младенцем, ребенком,
# подростком или взрослым, и показать результат.
if age <= 1:
    print('Младенец')
elif age > 1 and age < 13:
    print('Ребенок')
elif age > 13 and age < 20:
    print('Подросток')
else:
    print ('Взрослый')
```

Упражнение 3.4. Римские цифры

```
# Получить число
number = int(input('Введите целое число между 1 и 10: '))

# Напечатать римскую цифру
if number == 1:
    print ('I')
elif number == 2:
    print ('II')
elif number == 3:
    print ('III')
elif number == 4:
    print ('IV')
elif number == 5:
    print ('V')
elif number == 6:
    print ('VI')
```

```
elif number == 7:
    print ('VII')
elif number == 8:
    print ('VIII')
elif number == 9:
    print ('IX')
elif number == 10:
    print ('X')
else:
    print ('Ошибка: недопустимое число')
```

Упражнение 3.5. Масса и вес

```
# Глобальные константы
MASS_MULTIPLIER = 9.8
TOO_HEAVY = 500.0
TOO_LIGHT = 100.0

# Локальные переменные
weight = 0.0
mass = 0.0

# Получить массу
mass = float(input("Введите массу тела в килограммах: "))

# Вычислить вес
weight = mass * MASS_MULTIPLIER

# Показать расчет веса
print ('Вес объекта: ', format(weight, '.2f'))
if weight > TOO_HEAVY:
    print ('Объект слишком тяжелый. Он весит более',
           TOO_HEAVY, 'ньютонов.')
elif  weight < TOO_LIGHT:
    print ('Объект слишком легкий. Он весит менее',
           TOO_LIGHT, 'ньютонов.')
```

Упражнение 3.6. Волшебные даты

```
# Получить день
day = int(input('Введите день месяца: '))

# Получить месяц
month = int(input('Введите месяц в числовой форме: '))

# Получить год
year = int(input('Введите год в числовой форме: '))

# Проверить, введен ли допустимый день
if day > 31 or day < 1:
```

```
    print('Ошибка: введен недопустимый день')

# Проверить, введен ли допустимый месяц
elif month > 12 or month < 1:
    print('Ошибка: введен недопустимый месяц')

# Проверить, введен ли допустимый год
elif year > 99 or year < 0:
    print('Ошибка: введен недопустимый год')

# Введенные данные допустимы
else:
    # Показать расчет магической даты
    print('Введена дата ', day, '/', month, '/', year)
    if (day * month) == year:
        print ('Это магическая дата.')
    else:
        print ('Это не магическая дата.')
```

Упражнение 3.7. Цветовой микшер

```
# Глобальные переменные
RED = "красный"
BLUE = "синий"
YELLOW = "желтый"

# Получить от пользователя первый цвет.
color1 = input('Введите первый основной цвет буквами в нижнем регистре: ')

# Получить от пользователя второй цвет.
color2 = input('Введите второй основной цвет буквами в нижнем регистре: ')

# Проверить допустимость первого цвета.
if color1 != RED and color1 != BLUE and color1 != YELLOW:
    print('Ошибка: первый введенный цвет недопустимый.')

# Проверить допустимость второго цвета.
elif color2 != RED and color2 != BLUE and color2 != YELLOW:
    print('Ошибка: второй введенный цвет недопустимый.')

# Проверить, не являются ли цвета одинаковыми.
elif color1 == color2:
    print('Ошибка: два введенных цвета одинаковые.')

# Показать вторичный цвет, получающийся путем смешивания двух цветов.
else:
    # Определить вторичный цвет, если первый цвет красный.
    if color1 == RED:
        if color2 == BLUE:
            print('фиолетовый')
```

```
        else: # Цвет 2 должен быть желтым
            print('оранжевый')

    # Определить вторичный цвет, если первый цвет синий.
    elif color1 == BLUE:
        if color2 == RED:
            print('фиолетовый')
        else: # Цвет 2 должен быть желтым.
            print('зеленый')

    else: # Цвет 1 должен быть желтым.
        if color2 == RED:
            print('оранжевый')
        else: # Цвет 2 должен быть синим.
            print('зеленый')
```

Упражнение 3.8. Калькулятор сосисок для пикника

```
# Глобальные переменные
HOT_DOGS_PER_PACKAGE = 10  # количество сосисок в упаковке
BUNS_PER_PACKAGE = 8       # количество булочек в упаковке

# Локальные переменные
numAttending = 0    # Количество участников пикника
numPerPerson = 0    # Количество сосисок и булочек в расчете на человека
total = 0           # Общее количество требующихся сосисок и булочек
minDogs = 0         # Минимальное количество упаковок сосисок
minBuns = 0         # ТМинимальное количество упаковок булочек
dogsLeft = 0        # Количество сосисок, оставшихся в упаковке
bunsLeft = 0        # Количество булочек, оставшихся в упаковке

# Получить от пользователя количество людей, посещающих пикник.
numAttending = int(input('Введите количество людей, посещающих пикник: '))

# Получить от пользователя количество хот-догов в расчете на участника.
numPerPerson = int(input('Введите количество хот-догов в расчете на человека: '))

# Вычислить общее количество требующихся сосисок и булочек.
total = numAttending * numPerPerson

# Вычислить минимальное количество требующихся упаковок с сосисками.
minDogs = total // HOT_DOGS_PER_PACKAGE

# Определить, является ли количество посещающих людей
# достаточно большим, что требуется более одной упаковки
# сосисок.
if minDogs > 0:
    # Вычислить количество сосисок, оставшихся
    # в упаковке, если есть.
    dogsLeft = total % HOT_DOGS_PER_PACKAGE
```

```
    # Если будут остатки, то добавить дополнительную
    # упаковку сосисок.
    if dogsLeft != 0:
        minDogs += 1

# Количество посещающих людей достаточно малое, и поэтому
# требуется только одна упаковка сосисок.
else:
    # Присвоить минимальному количеству упаковок сосисок значение 1.
    minDogs = 1

# Определить количество оставшихся сосисок, если есть.
dogsLeft = HOT_DOGS_PER_PACKAGE * minDogs - total

# Вычислить минимальное количество упаковок булочек,
# необходимых для хот-догов.
minBuns = total // BUNS_PER_PACKAGE

# Определить, является ли количество посещающих людей
# достаточно большим, что требуется более одной упаковки
# булочек для хот-догов.
if minBuns > 0:
    # Вычислить количество булочек для хот-догов, оставшихся
    # в упаковке, если есть.
    bunsLeft = total % BUNS_PER_PACKAGE

    # Если будут остатки, то добавить дополнительную
    # упаковку булочек для хот-догов.
    if bunsLeft != 0:
        minBuns += 1

# Количество посещающих людей достаточно малое, и поэтому
# требуется только одна упаковка булочек для хот-догов.
else:
    # Присвоить минимальному количеству упаковок булочек
    # для хот-догов значение 1.
    minBuns = 1

# Вычислить количество оставшихся булочек для хот-догов, если есть.
bunsLeft = BUNS_PER_PACKAGE * minBuns - total

# Показать минимальное количество требующихся упаковок сосисок.
print('Минимальное количество требующихся упаковок сосисок:', minDogs)

# Показать минимальное количество требующихся упаковок булочек.
print('Минимальное количество требующих упаковок булочек:', minBuns)

# Показать количество оставшихся сосисок.
```

```
print('Количество оставшихся сосисок:', dogsLeft)

# Показать количество оставшихся булочек для хот-догов.
print('Количество оставшихся булочек:', bunsLeft)
```

Упражнение 3.9. Цвета колеса рулетки

```
# Локальные переменные
pocketNum = 0
outputStr = ''

# Получить от пользователя количество карманов.
pocketNum = int(input('Введите количество карманов от 0 до 36: '))

# Определить, является ли номер кармана допустимым.
if pocketNum < 0 or pocketNum > 36:
    outputStr = 'Ошибка: введено недопустимое значение'

# Определить цвет номера кормана.
else:
    # Для карманов с 1 по 10, карманы с нечетными номерами -
    # красные, а карманы с четными номерами - черные.
    if pocketNum >= 1 and pocketNum <= 10:
        if pocketNum % 2:
            outputStr = 'Черный'  # Четный
        else:
            outputStr = 'Красный' # Нечетный

    # Для карманов с  11 по 18, карманы с нечетными номерами -
    # черные, а карманы с четными номерами - красные.
    elif pocketNum >= 11 and pocketNum <= 18:
        if pocketNum % 2:
            outputStr = 'Красный' # Четный
        else:
            outputStr = 'Черный'  # Нечетный

    # Для карманов с 19 по 28, карманы с нечетными номерами -
    # красные, а карманы с четными номерами - черные.
    elif pocketNum >= 19 and pocketNum <= 28:
        if pocketNum % 2:
            outputStr = 'Черный'  # Четный
        else:
            outputStr = 'Красный' # Нечетный

    # Для карманов с 29 по 36, карманы с нечетными номерами -
    # черные, а карманы с четными номерами - красные.
    elif pocketNum >= 29 and pocketNum <= 36:
        if pocketNum % 2:
            outputStr = 'Красный' # Четный
        else:
```

```
            outputStr = 'Черный'  # Нечетный

    # Карман 0 - зеленый.
    else:
        outputStr = 'Зеленый'     # Зеро (ноль)

# Показать результат.
print(outputStr)
```

Упражнение 3.10. Игра в подсчитывание монет

```
# Глобальные переменные
COIN5_VALUE = 5
COIN10_VALUE = 10
COIN50_VALUE = 50
KOPECKS_IN_ROUBLE = 100

# Локальные переменные
coin5 = 0
coin10 = 0
coin50 = 0
totalValue = 0.0
totalRoubles = 0.0

# Получить от пользователей количество
# монет достоинством 5, 10 и 50.
coin5 = int(input('Введите количество монет достоинством 5 коп.: '))
coin10 = int(input('Введите количество монет достоинством 10 коп.: '))
coin50 = int(input('Введите количество монет достоинством 50 коп.: '))

# Сумма монет достоинством 5, 10 и 50 копеек
# для получения общей суммы в копейках.
totalValue = (coin5 * COIN5_VALUE) + \
             (coin10 * COIN10_VALUE)+ \
             (coin50 * COIN50_VALUE)

# Вычислить общее количество рублей
totalRoubles = totalValue / KOPECKS_IN_ROUBLE

# Определить, выиграл ли пользователь игру:
if totalRoubles > 1.0:
    # Сумма была больше одного рубля.
    print('Извините, введенная вами сумма больше одного рубля.')
elif totalRoubles < 1.0:
    # Сумма была меньше одного рубля.
    print('Извините, введенная вами сумма меньше одного рубля.')
else:
    # Сумма была равна ровно одному рублю.
    print('Поздравляем!')
    print('Сумма, которую вы ввели, равняется ровно одному рублю!')
```

```
    print('Вы выиграли!')
```

Упражнение 3.11. Очки книжного клуба

```
# Локальные переменные
number = 0
points = 0

# Получить количество книг, приобретенных пользователем.
number = int(input('Введите количество приобретенных книг: '))

# Определить количество заработанных очков.
if number == 2:
    points = 5
elif number <= 4:
    points = 15
elif number <= 6:
    points = 30
elif number >= 7:
    points = 60
else:
    points = 0

# Показать количество заработанных очков.
print('Вы приобрели', number, 'книг.')
print('В результате вы заработали', points, 'очков.')
```

Упражнение 3.12. Реализация программного обеспечения

```
# Именованная константа
RETAIL_PRICE = 99

# Локальные переменные
quantity = 0
fullPrice = 0.0
discountRate = 0.0
discountAmount = 0.0
totalAmount = 0.0

# Получить количество
quantity = int(input('Введите количество приобретенных пакетов: '))

# Вычислить скидочную ставку
if quantity > 99:
    discountRate = 0.40
elif quantity > 49:
    discountRate = 0.30
elif quantity > 19:
    discountRate = 0.20
elif quantity > 9:
```

```
    discountRate = 0.10
else:
    discountRate = 0

# Вычислить полную цену
fullPrice = quantity * RETAIL_PRICE

# Вычислить сумму скидки
discountAmount = fullPrice * discountRate

# Вычислить общую сумму
totalAmount = fullPrice - discountAmount

# Напечатать результаты
print ('Сумма скидки: $', format(discountAmount, '.2f'))
print ('Общая сумма: $', format(totalAmount, '.2f'))
```

Упражнение 3.13. Стоимость доставки

```
# Локальные переменные
weight = 0.0
shippingCost = 0.0

# Получить от пользователя вес пакета.
weight = float(input('Введите вес пакета в граммах: '))

# Вычислить стоимость доставки.
if weight > 1000:
    shippingCost = 475
elif weight > 600:
    shippingCost = 400
elif weight > 200:
    shippingCost = 300
else:
    shippingCost = 150

# Показать стоимость доставки.
print ('Стоимость доставки в рублях: $', format(shippingCost, '.2f'))
```

Упражнение 3.14. Индекс массы тела

```
# Локальные переменные
weight = 0.0
height = 0.0
IMT = 0.0    # индекс массы тела - ИМТ

# Получить от пользователя вес.
weight = float(input('Введите свою массу тела в килограммах: '))

# Получить от пользователя рост.
```

```
height = float(input('Введите свой рост в метрах: '))

# Вычислить массу тела.
IMT = weight / (height * height)

# Показать ИМТ.
print('Ваш индекс массы тела равняется', format(IMT, '.2f'))

# Определить и показать весовую категорию.
if IMT > 25:
    print('Вы весите выше нормы.')
elif IMT < 18.5:
    print('Вы весите ниже нормы.')
else:
    print('Ваш вес оптимален.')
```

Упражнение 3.15. Калькулятор времени

```
# Локальные переменные
days = 0
hours = 0
minutes = 0
seconds = 0
dayRemainder = 0
hourRemainder = 0
minuteRemainder = 0

# Получить от пользователя количество секунд.
seconds = int(input('Введите количество секунд: '))

# Получить от пользователя количество дней.
if seconds >= 86400:
    days = seconds // 86400
    dayRemainder = seconds % 86400

# Вычислить часы.
if seconds >= 3600:
    hours = seconds // 3600
    hourRemainder = seconds % 3600

# Вычислить минуты.
if seconds >= 60:
    minutes = seconds // 60
    minuteRemainder = seconds % 60

# Показать дни, часы, минуты, секунды.
if minutes == 0:
    print('Количество секунд менее одной минуты.')
else:
    print(seconds, 'секунд равняется:')
```

```
    print (minutes, 'полным минутам и', minuteRemainder, 'секундам.')
    if hours!=0:
        print (hours, 'полным часам и', hourRemainder, 'секундам.')
    if days!=0:
        print (days, 'полным дням и', dayRemainder, 'секундам.')
```

Упражнение 3.16. Дни в феврале

```
# Эта программа определяет количество дней в
# феврале для того или иного года.

print('Введите год: ', end='')
year = int(input())
if year % 100 == 0:
    if year % 400 == 0:
        leap_year = True
    else:
        leap_year = False
else:
    if year % 4 == 0:
        leap_year = True
    else:
        leap_year = False
if leap_year:
    print('Это високосный год. В феврале 29 дней.')
else:
    print('Это не високосный год. В феврале 28 дней.')
```

Упражнение 3.17. Диагностическое дерево проверки качества Wi-Fi

```
# Эта программа проводит вас через процесс отладки
# плохого Wi-Fi-соединения.

print('Перезагрузите компьютер и попробуйте подключиться.')
print('Вы исправили проблему?')
response = input()
if response == 'да':
    print('Рад был помочь.')
else:
    print('Перезагрузите маршрутизатор и попробуйте подключиться.')
    print('Вы исправили проблему?')
    response = input()
    if response == 'да':
        print('Рад был помочь.')
    else:
        print('Убедитесь, что кабели между маршрутизатором и модемом прочно подсоединены.')
        print('Вы исправили проблему?')
        response = input()
        if response == 'да':
            print('Рад был помочь.')
```

```
        else:
            print('Переместите маршрутизатор на новое место.')
            print('Вы исправили проблему?')
            response = input()
            if response == 'да':
                print('Рад был помочь.')
            else:
                print('Возьмите новый маршрутизатор.')
```

Упражнение 3.18. Селектор ресторанов

```
# Эта программа помогает вам выбрать ресторан.

# Инициализировать переменные
vegetarian = False
vegan = False
glutenFree = False

# Есть вегетариантцы?
print('Есть ли в группе людей вегетарианец? ', end='')
response = input()
if response == 'да':
    vegetarian = True

# Есть веганцы?
print('Есть ли в группе людей веганец? ', end='')
response = input()
if response == 'да':
    vegan = True

# Есть приверженцы безглютеновой диеты?
print('Есть ли в группе людей приверженец безглютеновой диеты? ', end='')
response = input()
if response == 'да':
    glutenFree = True

# Показать варианты ресторанов.
print('Вот варианты ресторанов:')
if not vegetarian and not vegan and not glutenFree:
    print("Изысканные гамбургеры от Джо")
if not vegan and not glutenFree:
    print("Блюда от итальянской мамы")
if not vegan:
    print('Центральная пиццерия')
print('Кафе за углом')
print("Кухня шеф-повара")
```

Упражнение 3.19. Модификация игры "Порази цель"

```
import turtle
```

```
# Именованные константы
SCREEN_WIDTH = 600     # Ширина экрана
SCREEN_HEIGHT = 600    # Высота экрана
TARGET_LLEFT_X = 100   # Левая нижняя координата X цели
TARGET_LLEFT_Y = 250   # Левая нижняя координата Y цели
TARGET_WIDTH = 25      # Ширина цели
FORCE_FACTOR = 30      # Фактор произвольной силы
PROJECTILE_SPEED = 1   # Скорость анимации снаряда
NORTH = 90             # Угол северного направления
SOUTH = 270            # Угол южного направления
EAST = 0               # Угол восточного направления
WEST = 180             # Угол западного направления

# Настроить окно.
turtle.setup(SCREEN_WIDTH, SCREEN_HEIGHT)

# Нарисовать цель.
turtle.hideturtle()
turtle.speed(0)
turtle.penup()
turtle.goto(TARGET_LLEFT_X, TARGET_LLEFT_Y)
turtle.pendown()
turtle.setheading(EAST)
turtle.forward(TARGET_WIDTH)
turtle.setheading(NORTH)
turtle.forward(TARGET_WIDTH)
turtle.setheading(WEST)
turtle.forward(TARGET_WIDTH)
turtle.setheading(SOUTH)
turtle.forward(TARGET_WIDTH)
turtle.penup()

# Центрировать черепаху.
turtle.goto(0, 0)
turtle.setheading(EAST)
turtle.showturtle()
turtle.speed(PROJECTILE_SPEED)

# Получить от пользователя угол и силу.
angle = float(input("Введите угол полета снаряда: "))
force = float(input("Введите пусковую силу (1-10): "))

# Рассчитать расстояние.
distance = force * FORCE_FACTOR

# Задать направление.
turtle.setheading(angle)
```

```
# Запустить снаряд.
turtle.pendown()
turtle.forward(distance)

# Снаряд поразил цель?
if (turtle.xcor() >= TARGET_LLEFT_X and
    turtle.xcor() <= (TARGET_LLEFT_X + TARGET_WIDTH) and
    turtle.ycor() >= TARGET_LLEFT_Y and
    turtle.ycor() <= (TARGET_LLEFT_Y + TARGET_WIDTH)):
    print('Цель поражена!')
else:
    print('Вы промахнулись.')

    # Показать подсказки.
    if turtle.xcor() > (TARGET_LLEFT_X + TARGET_WIDTH):
        print('Попробуйте угол побольше.')
    elif turtle.xcor() < TARGET_LLEFT_X:
        print('Попробуйте угол поменьше.')
    elif turtle.ycor() > (TARGET_LLEFT_Y + TARGET_WIDTH):
        print('Попробуйте силу поменьше.')
    elif turtle.ycor() < TARGET_LLEFT_Y:
        print('Попробуйте силу побольше.')
```