Упражнения по программированию главы 5

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

Упражнение по программированию 5.1. Конвертер километров

```
# Глобальная константа для преобразования
KILOMETERS_TO_MILES = 0.6214

# Определение главной функции
def main():
    # Локальные переменные
    mykilometers = 0.0   # Переменная для расстояний в километрах

    # Получить расстояние в километрах
    mykilometers = float(input("Введите расстояние в километрах: "))

    # Напечатать мили
    showMiles(mykilometers)

# Функция showMiles принимает километры в качестве аргумента
# и печатает эквивалент в милях.
def showMiles(kilometers):
    # Объявить локальную переменную
    miles = 0.0

    miles = kilometers * KILOMETERS_TO_MILES
    print(f'Преобразование {kilometers:,.2f} километров')
    print (f'в мили дает {miles:,.2f} миль.')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.2. Рефакторизация программы расчета налога с продаж

```
# Глобальные константы для федерального и регионального налогов с продаж
STATE_TAX_RATE = 0.05
COUNTY_TAX_RATE = 0.025

# Главная функция
def main():
    # Локальные переменные
    purchase = 0.0
    stateTax = 0.0
    countyTax = 0.0

    # Получить сумму покупки
    purchase = float(input('Введите сумму покупки: '))
```

```
    # Вычислить федеральный налог с продаж
    stateTax = purchase * STATE_TAX_RATE

    # Вычислить региональный налог с продаж
    countyTax = purchase * COUNTY_TAX_RATE

    # Напечатать информацию о продаже
    showSale(purchase, stateTax, countyTax)

# Функция showSale принимает в качестве аргументов
# purchase, stateTax, countyTax и печатает соответствующую
# информацию о сумме продажи.
def showSale (purchase, stateTax, countyTax):
    # Локальные переменные
    totalTax = 0.0
    totalSale = 0.0
    totalTax = stateTax + countyTax
    totalSale = purchase + totalTax

    print (f'Сумма покупки: {purchase:,.2f}')
    print (f'Федеральный налог: {stateTax:,.2f}')
    print (f'Региональный налог: {countyTax:,.2f}')
    print (f'Суммарный налог: {totalTax:,.2f}')
    print (f'Сумма продажи: {totalSale:,.2f}')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.3. Какова стоимость страховки?

```
# Глобальная константа для стоимости замещения в процентах
REPLACE_PERCENT = 0.8

# Главная функция
def main():
    # Локальные переменные
    replace= 0.0
    minInsure = 0.0

    # Получить стоимость замещения.
    replace = float(input('Введите стоимость замещения: '))

    # Вычислить сумму страховки
    minInsure = replace * REPLACE_PERCENT

    # Напечатать информацию о страховке.
    showInsure(replace, minInsure)

# Функция showInsure принимает в качестве аргументов значение
```

```
# замещения replace и минимальной рекомендуемой страховки minInsure
# и показывает информацию о сделке.
def showInsure (replace, minInsure):
    print (f'Стоимость замещения:\t${replace:,.2f}')
    print (f'Страхуемый процент:\t\t{int(REPLACE_PERCENT * 100)}%')
    print (f'Минимальная сумма, подлежащая страхованию: ${minInsure:,.2f}')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.4. Расходы на автомобиль

```
# Главный модуль
def main():
    # Локальные переменные
    loan = 0.0
    insurance = 0.0
    gas = 0.0
    oil = 0.0
    tires = 0.0
    maintenance = 0.0

    # Получить сумму выплат по ссуде.
    loan = float(input('Введите ежемесячную сумму выплат по ссуде: '))

    # Получить сумму выплат по страховке.
    insurance = float(input('Введите ежемесячную сумму выплат по страховке: '))

    # Получить ежемесячную сумму расходов на топливо.
    gas = float(input('Введите ежемесячную сумму расходов на топливо: '))

    # Получить ежемесячную сумму расходов на масло.
    oil = float(input('Введите ежемесячную сумму расходов на масло: '))

    # Получить ежемесячную сумму расходов на шины.
    tires = float(input('Введите ежемесячную сумму расходов на шины: '))

    # Получить ежемесячную сумму расходов на техобслуживание.
    maintenance = float(input('Введите ежемесячную сумму расходов на' \
                              ' техобслуживание: '))

    # Напечатать информацию о транспортном средстве.
    showExpenses(loan, insurance, gas, oil, \
                 tires, maintenance)

# Функция showExpenses принимает в качестве аргументов информацию
# о ссуде loan, страховке insurance, топливе gas, масле oil,
# шинах tires и техобслуживании maintenance и показывает
# соответствующую информацию о суммарных расходах.
def showExpenses(loan, insure, gas, oil, \
```

```
                tires, maintenance):
    # Локальные переменные
    totalMonth = 0.0
    totalYear = 0.0
    totalMonth = loan + insure + gas + oil + \
                tires + maintenance
    totalYear = totalMonth * 12

    # Print monthly and annual information.
    print (f'Суммарные ежемесячные расходы: ${totalMonth:,.2f}')
    print (f'Суммарные ежегодные расходы:   ${totalYear:,.2f}')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.5. Налог на недвижимое имущество

```
# Глобальные константы для имущественного налога
ASSESS_PERCENT = 0.6
PROPERTY_TAX_PERCENT = 0.0072

# Главная функция
def main():
    # Локальные переменные
    actualValue = 0.0
    assessValue = 0.0
    propertyTax = 0.0

    # Получить фактическую стоимость.
    actualValue = float(input('Введите фактическую стоимость: '))

    # Вычислить оценочную стоимость.
    assessValue = actualValue * ASSESS_PERCENT

    # Вычислить имущественный налог.
    propertyTax = assessValue * PROPERTY_TAX_PERCENT

    # Напечатать информацию об имущественном налоге.
    showPropertyTax(assessValue, propertyTax)

# Функция showPropertyTax принимает в качестве аргументов
# оценочную стоимость и величину имущественного налога и
# показывает информацию об имущественном налоге.
def showPropertyTax (assessValue, propertyTax):
    print (f'Оценочная стоимость: ${assessValue:,.2f}')
    print (f'Имущественный налог: ${propertyTax:,.2f}')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.6. Калории от жиров и углеводов

```
# Глобальные константы для калорий
CALORIES_FROM_FAT = 9    # калории от потребления жиров
CALORIES_FROM_CARBS = 4  # калории от потребления углеводов

# Главный модуль
def main():
    # Локальные переменные
    gramsFat = 0.0
    gramsCarbs = 0.0
    caloriesFat = 0.0
    caloriesCarbs = 0.0

    # Получить граммы жиров.
    gramsFat = float(input('Введите граммы потребленных жиров: '))

    # Получить граммы углеводов.
    gramsCarbs = float(input('Введите граммы потребленных углеводов: '))

    # Вычислить калории от потребления жиров.
    caloriesFat = gramsFat * CALORIES_FROM_FAT

    # Вычислить калории от потребления углеводов.
    caloriesCarbs = gramsCarbs * CALORIES_FROM_CARBS

    # Напечатать калории.
    showCarbs(gramsFat, gramsCarbs, caloriesFat, caloriesCarbs)

# Функция showCarbs принимает в качестве аргументов количество
# граммов жиров и углеводов, а также калорий от жиров и углеводов
# и показывает получающиеся в результате калории.
def showCarbs(gramsFat, gramsCarbs, caloriesFat, caloriesCarbs):
    print(f'Граммы жиров:              {gramsFat:.2f}')
    print(f'Калории за счет жиров:     {caloriesFat:.2f}')
    print(f'Граммы углеводов:          {gramsCarbs:.2f}')
    print(f'Калории за счет углеводов: {caloriesCarbs:.2f}')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.7. Сидячие места на стадионе

```
# Глобальные константы для стоимости сидячих мест на стадионе
CLASS_A_SEATS = 20
CLASS_B_SEATS = 15
CLASS_C_SEATS = 10

# Главная функция
def main():
```

```
    # Локальные переменные
    countAseats = 0
    countBseats = 0
    countCseats = 0
    incomeAseats = 0.0
    incomeBseats = 0.0
    incomeCseats = 0.0

    # Получить количество A
    countAseats = int(input('Введите количество мест класса A: '))

    # Получить количество B
    countBseats = int(input('Введите количество мест класса B: '))

    # Получить количество C
    countCseats = int(input('Введите количество мест класса C: '))

    # Вычислить доход за счет мест класса A
    incomeAseats = countAseats * CLASS_A_SEATS

    # Вычислить доход за счет мест класса B
    incomeBseats = countBseats * CLASS_B_SEATS

    # Вычислить доход за счет мест класса C
    incomeCseats = countCseats * CLASS_C_SEATS

    # Напечатать доход
    showIncome(incomeAseats, incomeBseats, incomeCseats)

# Функция showIncome принимает в качестве аргументов доход,
# полученный за счет мест класса A, B и C показывает суммарный доход.
def showIncome(incomeAseats, incomeBseats, incomeCseats):
    # Локальная переменная
    totalIncome = 0.0

    # Вычислить суммарный доход
    totalIncome = incomeAseats + incomeBseats + incomeCseats

    # Показать результаты
    print (f'Доход за счет мест класса A: ${incomeAseats:.2f}')
    print (f'Доход за счет мест класса B: ${incomeBseats:.2f}')
    print (f'Доход за счет мест класса C: ${incomeCseats:.2f}')
    print (f'Суммарный доход: ${totalIncome:.2f}')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.8. Оценщик малярных работ

```
# Глобальные константы оценщика малярных работ
```

```
FEET_PER_GALLON = 5     # Объем банки краски в литрах
LABOR_HOURS = 8         # Количество рабочих часов
LABOR_CHARGE = 2000     # Стоимость работы рублей в час

# Главный модуль
def main():
    # Локальные переменные
    pricePaint = 0.0
    feetWall = 0.0
    gallonPaint = 0
    hourLabor = 0
    costPaint = 0.0
    costLabor = 0.0

    # Получить площадь поверхности
    feetWall = float(input('Введите площадь поверхности в кв. метрах: '))

    # Получить paint price
    pricePaint = float(input('Введите стоимость 5-литровой банки краски, руб.: '))

    # Вычислить объем краски
    gallonPaint = int(0.1 * feetWall * FEET_PER_GALLON)

    # Вычислить количество рабочих часов
    hourLabor = int(0.1 * feetWall * LABOR_HOURS)

    # Вычислить стоимость работы
    costLabor = hourLabor * LABOR_CHARGE

    # Вычислить стоимость краски
    costPaint = gallonPaint/5 * pricePaint

    # Напечатать оценочную стоимость
    showCostEstimate(gallonPaint, hourLabor, costPaint, costLabor)

# Функция showCostEstimate принимает в качестве аргументов
# gallonPaint, hourLabor, costPaint, costLabor и
# показывает соответствующие данные.
def showCostEstimate(gallonPaint, hourLabor, costPaint, costLabor):
    # Локальная переменная
    totalCost = 0.0

    # Вычислить суммарную стоимость
    totalCost = costPaint + costLabor

    # Показать результаты
    print ('Краска, литров: ', gallonPaint)
    print ('Работа, часы: ', hourLabor)
    print (f'Стоимость краски, руб.:    {costPaint:,.2f}')
```

```
    print (f'Стоимость работы, руб.:    {costLabor:,.2f}')
    print (f'Суммарная стоимость, руб.: {totalCost:,.2f}')

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.9. Месячный налог с продаж

```
# Объявления переменных
sales = 0.0
stateTax = 0.0
countyTax = 0.0
totalTax = 0.0

# Константы для федеральной и региональной ставки налога
STATE_TAX_RATE = 0.05
COUNTY_TAX_RATE = 0.025

# Получить суммарные продажи за месяц.
sales = float(input('Введите суммарные продажи за месяц: '))

# Вычислить федеральный налог с продаж.
stateTax = sales * STATE_TAX_RATE

# Вычислить региональный налог с продаж.
countyTax = sales * COUNTY_TAX_RATE

# Вычислить суммарный налог.
totalTax = stateTax + countyTax

# Напечатать информацию о налогах.
print('Федеральный налог: $', format(stateTax, ',.2f'), sep='')
print('Региональный налог: $', format(countyTax, ',.2f'), sep='')
print('Суммарный налог: $', format(totalTax, ',.2f'), sep='')
```

Упражнение по программированию 5.10. Футы в дюймы

```
# Главная функция
def main():

    # Локальные переменные
    feet = 0.0
    inches = 0.0

    # Получить от пользователя количество футов.
    feet = float(input('Введите количество футов: '))

    # Показать соответствующее количество дюймов.
    inches = feet_to_inches(feet)
    print (feet, 'футов =', inches, 'дюймов')
```

```
# Функция feet_to_inches принимает в качестве аргументов
# количество футов и возвращает количество дюймов для
# этого количество футов.
def feet_to_inches(feet):
    return 12 * feet

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.11. Математический тест

```
import random

# Главная функция
def main():
    # Локальные переменные
    num1 = 0
    num2 = 0
    correctAnswer = 0
    userAnswer = 0

    # Получить числа
    num1 = random.randint(0, 999)
    num2 = random.randint(0, 999)

    # Показать математическую задачу
    displayProblem(num1, num2)

    # Получить ответ пользователя
    userAnswer = getAnswer()

    # Вычислить правильный ответ
    correctAnswer = num1 + num2

    # Показать результат
    showResult(correctAnswer, userAnswer)

# Функция displayProblem принимает в качестве аргументов
# числа и их показывает
def displayProblem(num1,num2):
    print (format(num1, '5'))
    print ('+', end='')
    print (format(num2, '4'))

# Функция getAnswer получает и возвращает ответ пользователя
def getAnswer():
    inputAnswer = int(input('Введите сумму чисел: '))
    return inputAnswer
```

```
# Функция showResult сообщает, является ли ответ
# пользователя правильным или нет
def showResult (correctAnswer, answer):
    if correctAnswer == answer:
        print ('Правильный ответ – хорошая работа!')
    else:
        print ('Неправильно... Правильный ответ равняется:', \
               correctAnswer)

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.12. Максимумальное из двух значений

```
# Главный модуль
def main():
    # Локальные переменные
    num1 = 0
    num2 = 0

    # Получить числа
    num1 = int(input('Введите число № 1: '))
    num2 = int(input('Введите число № 2: '))

    # Показать результат
    print ('Максимальное число равняется:', maximum(num1, num2))

# Функция maximum возвращает максимальное из
# двух чисел, которые она получает в качестве аргументов
def maximum(num1, num2):
    if num1 > num2:
        return num1
    else:
        return num2

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.13. Высота падения

```
# Главная функция
def main():
    # Локальная переменная
    distance = 0.0

    # Определить таблицу результатов
    print ('Время\t Расстояние падения')
    print ('--------------------------')
```

```
    # В цикле перебрать время (в секундах)
    for time in range(1, 11):
        distance = falling_distance(time)
        print(time, '\t\t', format(distance, '10.2f'))

# Функция falling_distance получает время падения
# объекта и возвращает расстояние, которое он пролетел
# за это время
def falling_distance(time):
    fallDistance = (9.8 * time * time) / 2
    return fallDistance

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.14. Кинетическая энергия

```
# Главная функция
def main():
    # Локальные переменные
    mass = 0.0
    velocity = 0.0
    KE = 0.0

    # Получить массу
    mass = float(input('Введите массу объекта в килограммах: '))

    # Получить скорость
    velocity = float(input('Введите скорость объекта в метрах в секунду: '))

    # Получить кинетическую энергию
    KE = kinetic_energy(mass, velocity)

    # Показать кинетическую энергию
    print ('Кинетическая энергия равняется:', format(KE, '.2f'), 'джоулей')

# Функция kinetic_energy получает массу и скорость объекта и
# возвращает его кинетическую энергию
def kinetic_energy(mass, velocity):
    # Локальная переменная
    KE = 0.0

    KE = (mass * velocity * velocity) / 2
    return KE

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.15. Средний балл и его уровень

```
# Главная функция
def main():
    # Локальные переменные
    average = 0.0
    score1 = 0.0
    score2 = 0.0
    score3 = 0.0
    score4 = 0.0
    score5 = 0.0

    # Получить scores
    score1 = float(input('Введите оценку 1: '))
    score2 = float(input('Введите оценку 2: '))
    score3 = float(input('Введите оценку 3: '))
    score4 = float(input('Введите оценку 4: '))
    score5 = float(input('Введите оценку 5: '))

    # Вычислить средний балл
    average = calc_average(score1, score2, score3, score4, score5)

    # Показать уровень оценки и средний балл в табличной форме
    print('\nоценка\t\t\t число\t буква')
    print('------------------------------')
    print('оценка 1:\t\t', score1, '\t', determine_grade(score1))
    print('оценка 2:\t\t', score2, '\t', determine_grade(score2))
    print('оценка 3:\t\t', score3, '\t', determine_grade(score3))
    print('оценка 4:\t\t', score4, '\t', determine_grade(score4))
    print('оценка 5:\t\t', score5, '\t', determine_grade(score5))
    print('------------------------------')
    print ('Средний балл:\t', average, '\t', \
           determine_grade(average))

# Функция calc_average возвращает средний балл из 5 уровней
def calc_average(s1, s2, s3, s4, s5):
    return  (s1 + s2 + s3 + s4 + s5) / 5.0

# Функция determine_grade получает числовой
# уровень и возвращает соответствующий буквенный уровень
def determine_grade(score):
    if score >= 90:
        return 'A'
    elif score >= 80:
        return 'B'
    elif score >= 70:
        return 'C'
    elif score >= 60:
        return 'D'
    else:
```

```
        return 'F'

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.16. Счетчик четных/нечетных чисел

```
import random

# Главная функция
def main():
    # Локальные переменные
    currentNumber = 0
    oddCounter = 0
    evenCounter = 0
    totalNumbers = 100

    for counter in range(totalNumbers):

        # Получить случайное число
        currentNumber = random.randint(1, 1000)

        # Проверить число на четность/нечетность
        if isEven(currentNumber):
            evenCounter+=1
        else:
            oddCounter+=1

    print ('Из', totalNumbers, 'случайных чисел,', oddCounter,\
           'были нечетными и', evenCounter, 'были четными.')

# Функция isEven возвращает True, если число четное, и
# False, если нечетное.
def isEven(number):
    if number % 2 == 0:
        return True
    else:
        return False

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.17. Простые числа

```
# Главная функция
def main():

    # Локальная переменная
    number = 0
```

```
    # Получить число
    number = int(input('Введите целое число: '))

    # Показать информацию о том, является ли число простым
    if is_prime(number):
        print ('Введенное вами число является простым.')
    else:
        print ('Введенное вами число не является простым.')

# Функция is_prime получает в качестве аргумента число и
# возвращает True, если число простое, и False в противном случае.
def is_prime(number):
    # Локальные переменные
    half = int(number / 2)
    status = True

    for count in range(2, half + 1):
        if number % count == 0:
            status = False

    return status

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.18. Список простых чисел

```
# Главная функция
def main():
    # Локальная переменная
    totalNumbers = 100
    print('число', '\t', 'простое или нет')
    print('------------------------')

    # Для каждого числа напечатать, является ли оно простым или нет
    for number in range(1, totalNumbers + 1):

        # Показать, является ли число простым
        if is_prime(number):
            print (format(number,'3'), '\t', 'простое')
        else:
            print (format(number,'3'), '\t', 'не простое')

# Функция is_prime получает в качестве аргумента число и
# возвращает True, если число простое, и False в противном случае.
def is_prime(number):
    # Локальные переменные
    half = int(number / 2)
    status = True
```

```
    for count in range(2, half + 1):
        if number % count == 0:
            status = False

    return status

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.19. Будущая стоимость

```
# Главная функция
def main():

    # Локальные переменные для вводимых пользователем значений
    presentValue = 0.0
    interestRate = 0.0
    months = 0
    futureValue = 0.0

    # Получить от пользователя конкретные значения
    presentValue = float(input('Введите текущую сумму ' \
                               'на счету в долларах: '))

    interestRate = float(input('Введите ежемесячную процентную ' \
                               'ставку: '))

    months = int(input('Введите количество месяцев: '))

    # Получить ожидаемую будущую сумму на счету
    futureValue = getFutureValue(presentValue, interestRate, months)

    print ('Информация по Вашему счету следующая:')
    print('Текущая сумма: $', format(presentValue, ',.2f'), sep='')
    print(f'Процентная ставка: {interestRate:,.2f}%')
    print('После ', months, \
          ' месяцев сумма на счету составит $', \
          format(futureValue, '.2f'), sep='')

# Функция getFutureValue получает текущую сумму, процентную
# ставку и количество месяцев, в течение которых деньги будут
# оставаться на счету, и возвращает будущую сумму на счету.
def getFutureValue(P, interest, t):
    # Определить локальную переменную
    F = 0.0
    i = interest / 100 # написать процент как дробь
    F = P * ((1 + i) **t)
    return F

# Вызвать главную функцию.
```

```
main()
```

Упражнение по программированию 5.20. Игра в угадывание случайного числа

```
import random

# Главная функция
def main():

    # Инициализация локальных переменных
    number = 0
    play = 1

    # Продолжить представлять числа для угадывания пользователю
    # до тех пор, пока пользователь желает продолжать игру.
    while(play > 0):
        number = random.randint(1, 100)
        play = playGuessingGame(number)

    print('Спасибо за игру!')

# Функция playGuessingGame получает в качестве аргумента
# число, которое пользователю нужно угадать и предлагает
# пользователю его угадать. Если пользователь угадывает неправильно,
# то он получает об этом сообщение, и ему предлагается попытаться еще.
# В противном случае возвращается догадка пользователя.
def playGuessingGame(number):
    # Получить догадку пользователя.
    userGuess = int(input('Введите число между 1 и 100 ' \
                          'либо 0, чтобы завершить игру: '))

    # До тех пор, пока пользователь не хочет прекратить игру
    while userGuess > 0:
        if userGuess > number:
            print('Слишком высоко, попробуйте еще раз')
            userGuess = int(input('Введите число между 1 и 100 ' \
                                  'либо 0, чтобы завершить: '))
        elif userGuess < number:
            print('Слишком низко, попробуйте еще раз')
            userGuess = int(input('Введите число между 1 и 100 ' \
                                  'либо 0, чтобы завершить: '))
        else:
            print('Поздравляем! Вы угадали правильное число!')
            return userGuess  # Start the game again

    return userGuess # userGuess равно 0, и пользователь хочет завершить.

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.21. Игра "Камень, ножницы, бумага"

```
import random

# Глобальные константы
COMPUTER_WINS = 1
PLAYER_WINS = 2
TIE = 0
INVALID = 3
ROCK = 1
PAPER = 2
SCISSORS = 3

# Главная функция
def main():

    result = TIE

    while result==TIE:
        # Получить число для компьютера
        computer = random.randint(1, 3)

        # Получить число от игрока
        player = int(input('Введите 1 для камня, ' \
                           '2 для бумаги, 3 для ножниц: '))

        print ('Компьютер выбрал', choiceString(computer))
        print ('Вы выбрали', choiceString(player))

        result = rockPaperScissors(computer, player)

        if result == TIE:
            print('Вы сделали тот же выбор, что и компьютер. Попробуем еще раз')

    if (result == COMPUTER_WINS):
        print ('Компьютер победил')
    elif result == PLAYER_WINS:
        print ('Вы победили')
    else:
        print ('Вы сделали недопустимый выбор. Победителя нет')

# Функция rockPaperScissors получает числа, представляющие
# варианты выбранные компьютером и игроком.
# Она возвращает:
# 0, если сделан одинаковый выбор,
# 1, если победил компьютер,
# 2, если победил игрок или
# 3, если игрок сделал недопустимый выбор.
def rockPaperScissors(computer, player):
```

```
    if(computer == player):
        return TIE

    if computer == ROCK:
        if player == PAPER:
            return PLAYER_WINS
        elif player == SCISSORS:
            return COMPUTER_WINS
        else:
            return INVALID
    elif computer == PAPER:
        if player == ROCK:
            return COMPUTER_WINS
        elif player == SCISSORS:
            return PLAYER_WINS
        else:
            return INVALID
    else: # компьютер выбрал ножницы
        if player == ROCK:
            return PLAYER_WINS
        elif player == PAPER:
            return COMPUTER_WINS
        else:
            return INVALID

# Функция choiceString показывает вариант в строковом формате
def choiceString(choice):
    if choice == ROCK:
        return 'камень'
    elif choice == PAPER:
        return 'бумага'
    elif choice == SCISSORS:
        return 'ножницы'
    else:
        return 'что-то пошло не так'

# Вызвать главную функцию.
main()
```

Упражнение по программированию 5.22. Черепашья графика: функция рисования треугольника

```
import turtle

def triangle(x1, y1, x2, y2, x3, y3, color):
    # Задать цвет заполнения.
    turtle.fillcolor(color)

    # Поднять перо и переместить черепаху.
    turtle.penup()
```

```
    turtle.goto(x1, y1)

    # Начертить треугольник.
    turtle.pendown()
    turtle.begin_fill()
    turtle.goto(x2, y2)
    turtle.goto(x3, y3)
    turtle.goto(x1, y1)
    turtle.end_fill()

def main():
    # Инициализировать черепаху.
    turtle.hideturtle()
    turtle.speed(0)

    # Начертить три треугольника.
    triangle(0, 0, 100, 0, 0, -100, 'red')
    triangle(0, 0, -100, 0, 0, -100, 'green')
    triangle(0, -100, -100, -200, 100, -200, 'blue')

    # Не закрывать окно.
    turtle.done()

# Вызвать главную функцию
main()
```

Упражнение по программированию 5.23. Черепашья графика: модульный снеговик

```
import turtle

ANIMATION_SPEED = 0
BASE_X = 0
BASE_Y = -200
BASE_RADIUS = 100
MID_X = 0
MID_Y = 0
MID_RADIUS = 60

RIGHT_ARM_X1 = 60
RIGHT_ARM_Y1 = 60
RIGHT_ARM_X2 = 108
RIGHT_ARM_Y2 = 75
RIGHT_ARM_X3 = 118
RIGHT_ARM_Y3 = 75
RIGHT_ARM_X4 = 118
RIGHT_ARM_Y4 = 85

LEFT_ARM_X1 = -60
LEFT_ARM_Y1 = 60
```

```
LEFT_ARM_X2 = -105
LEFT_ARM_Y2 = 70
LEFT_ARM_X3 = -120
LEFT_ARM_Y3 = 110
LEFT_ARM_X4 = -130
LEFT_ARM_Y4 = 115
LEFT_ARM_X5 = -120
LEFT_ARM_Y5 = 125

HEAD_X = 0
HEAD_Y = 120
HEAD_RADIUS = 40

LEFT_EYE_X = -20
LEFT_EYE_Y = 170
RIGHT_EYE_X = 20
RIGHT_EYE_Y = 170
EYE_RADIUS = 5

MOUTH_START_X = -25
MOUTH_START_Y = 140
MOUTH_END_X = 25
MOUTH_END_Y = 140

HAT_X1 = -50
HAT_Y1 = 180
HAT_X2 = 50
HAT_Y2 = 180
HAT_X3 = 50
HAT_Y3 = 205
HAT_X4 = -50
HAT_Y4 = 205
HAT_X5 = -30
HAT_Y5 = 205
HAT_X6 = 30
HAT_Y6 = 205
HAT_X7 = 30
HAT_Y7 = 245
HAT_X8 = -30
HAT_Y8 = 245

def drawBase():
       # Нарисовать основание
    turtle.penup()
    turtle.goto(BASE_X, BASE_Y)
    turtle.pendown()
    turtle.circle(BASE_RADIUS)

def drawMidSection():
```

```
    turtle.penup()
    turtle.goto(MID_X, MID_Y)
    turtle.pendown()
    turtle.circle(MID_RADIUS)

def drawArms():
    # Нарисовать правую руку
    turtle.penup()
    turtle.goto(RIGHT_ARM_X1, RIGHT_ARM_Y1)
    turtle.pendown()
    turtle.goto(RIGHT_ARM_X2, RIGHT_ARM_Y2)
    turtle.goto(RIGHT_ARM_X3, RIGHT_ARM_Y3)
    turtle.penup()
    turtle.goto(RIGHT_ARM_X2, RIGHT_ARM_Y2)
    turtle.pendown()
    turtle.goto(RIGHT_ARM_X4, RIGHT_ARM_Y4)

    # Нарисовать левую руку
    turtle.speed(1)
    turtle.penup()
    turtle.goto(LEFT_ARM_X1, LEFT_ARM_Y1)
    turtle.pendown()
    turtle.goto(LEFT_ARM_X2, LEFT_ARM_Y2)
    turtle.goto(LEFT_ARM_X3, LEFT_ARM_Y3)
    turtle.goto(LEFT_ARM_X4, LEFT_ARM_Y4)
    turtle.penup()
    turtle.goto(LEFT_ARM_X3, LEFT_ARM_Y3)
    turtle.pendown()
    turtle.goto(LEFT_ARM_X5, LEFT_ARM_Y5)

def drawHead():
    # Нарисовать голову
    turtle.penup()
    turtle.goto(HEAD_X, HEAD_Y)
    turtle.pendown()
    turtle.circle(HEAD_RADIUS)

    # Нарисовать левый глаз
    turtle.penup()
    turtle.goto(LEFT_EYE_X, LEFT_EYE_Y)
    turtle.pendown()
    turtle.circle(EYE_RADIUS)

    # Нарисовать правый глаз
    turtle.penup()
    turtle.goto(RIGHT_EYE_X, RIGHT_EYE_Y)
    turtle.pendown()
    turtle.circle(EYE_RADIUS)
```

```
    # Нарисовать рот
    turtle.penup()
    turtle.goto(MOUTH_START_X, MOUTH_START_Y)
    turtle.pendown()
    turtle.goto(MOUTH_END_X, MOUTH_END_Y)

def drawHat():
    # Нарисовать нижнюю часть шляпы
    turtle.penup()
    turtle.goto(HAT_X1, HAT_Y1)
    turtle.fillcolor('black')
    turtle.pendown()
    turtle.begin_fill()
    turtle.goto(HAT_X2, HAT_Y2)
    turtle.goto(HAT_X3, HAT_Y3)
    turtle.goto(HAT_X4, HAT_Y4)
    turtle.end_fill()

    # Нарисовать верхнюю часть шляпы
    turtle.penup()
    turtle.goto(HAT_X5, HAT_Y5)
    turtle.pendown()
    turtle.begin_fill()
    turtle.goto(HAT_X6, HAT_Y6)
    turtle.goto(HAT_X7, HAT_Y7)
    turtle.goto(HAT_X8, HAT_Y8)
    turtle.end_fill()

def main():
    turtle.speed(ANIMATION_SPEED)
    turtle.hideturtle()
    drawBase()
    drawMidSection()
    drawArms()
    drawHead()
    drawHat()

    # Не закрывать окно.
    turtle.done()

# Вызвать главную функцию
main()
```

Упражнение по программированию 5.24. Черепашья графика: прямоугольный узор

```
import turtle

# Именованные константы
ANIMATION_SPEED = 0
```

```
BASE_X = 0
BASE_Y = 0

def rectangle(x, y, width, height, color):
    turtle.setheading(0)
    turtle.fillcolor(color)     # Задать цвет заполнения
    turtle.penup()              # Поднять перо
    turtle.goto(x, y)           # Переместить в указанную позицию
    turtle.pendown()            # Опустить перо

    # Начертить прямоугольник.
    turtle.begin_fill()
    turtle.forward(width)
    turtle.left(90)
    turtle.forward(height)
    turtle.left(90)
    turtle.forward(width)
    turtle.left(90)
    turtle.forward(height)
    turtle.end_fill()

def rectangular_pattern(width, height):
    # Узор состоит из 3 вложенных прямоугольников, в которых
    # диагональные отрезки соединяют углы и четыре стороны.

    # Начертить самый внешний прямоугольник.
    rectangle(BASE_X, BASE_Y, width, height, 'white')

    # Начертить средний прямоугольник.
    middle_x = BASE_X + width / 8
    middle_y = BASE_Y + height / 8
    middle_width = width - width / 4
    middle_height = height - height / 4
    rectangle(middle_x, middle_y, middle_width, middle_height, 'white')

    # Начертить самый внутренний прямоугольник.
    inner_x = BASE_X + width / 4
    inner_y = BASE_Y + height / 4
    inner_width = width - width / 2
    inner_height = height - height / 2
    rectangle(inner_x, inner_y, inner_width, inner_height, 'black')

    # Начертить диагональные соединяющие отрезки.
    turtle.penup()
    turtle.goto(BASE_X, BASE_Y)
    turtle.pendown()
    turtle.goto(inner_x, inner_y)

    turtle.penup()
```

```
    turtle.goto(BASE_X + width, BASE_Y + height)
    turtle.pendown()
    turtle.goto(inner_x + inner_width, inner_y + inner_height)

    turtle.penup()
    turtle.goto(BASE_X + width, BASE_Y)
    turtle.pendown()
    turtle.goto(inner_x + inner_width, inner_y)

    turtle.penup()
    turtle.goto(BASE_X, BASE_Y + height)
    turtle.pendown()
    turtle.goto(inner_x, inner_y + inner_height)

    # Начертить горизонтальные соединяющие отрезки.
    turtle.penup()
    turtle.goto(BASE_X, BASE_Y + height / 2)
    turtle.pendown()
    turtle.goto(inner_x, BASE_Y + height / 2)

    turtle.penup()
    turtle.goto(BASE_X + width, BASE_Y + height / 2)
    turtle.pendown()
    turtle.goto(inner_x + inner_width, BASE_Y + height / 2)

    # Начертить вертикальные соединяющие отрезки.
    turtle.penup()
    turtle.goto(BASE_X + width / 2, BASE_Y)
    turtle.pendown()
    turtle.goto(BASE_X + width / 2, inner_y)

    turtle.penup()
    turtle.goto(BASE_X + width / 2, BASE_Y + height)
    turtle.pendown()
    turtle.goto(BASE_X + width / 2, inner_y + inner_height)

def main():
    turtle.speed(ANIMATION_SPEED)
    turtle.hideturtle()
    w = int(input('Введите ширину: '))
    h = int(input('Введите высоту: '))
    rectangular_pattern(w, h)

    # Не закрывать окно.
    turtle.done()

# Вызвать главную функцию
main()
```

Упражнение по программированию 5.25. Черепашья графика: шахматная доска

```
import turtle

# Именованные константы
ANIMATION_SPEED = 0
SCREEN_WIDTH = 500
SCREEN_HEIGHT = SCREEN_WIDTH
NUM_SQUARES_IN_A_ROW = 5
NUM_SQUARES_IN_A_COL = 5
SQUARE_WIDTH = int(SCREEN_WIDTH / NUM_SQUARES_IN_A_ROW)
SCREEN_LEFT_EDGE_X = int(-(SCREEN_WIDTH / 2))
SCREEN_TOP_EDGE_Y = int(SCREEN_HEIGHT / 2)
FIRST_X = SCREEN_LEFT_EDGE_X
LAST_X = FIRST_X + (NUM_SQUARES_IN_A_ROW * SQUARE_WIDTH)
FIRST_Y = SCREEN_TOP_EDGE_Y - SQUARE_WIDTH
LAST_Y = FIRST_Y - (NUM_SQUARES_IN_A_COL * SQUARE_WIDTH)


# Функция square чертит квадрат. Параметры x и y являются
# координатами левого нижнего угла. Параметр width
# является шириной каждой стороны. Параметр color является
# цветом заполнения, как строковое значение.

def square(x, y, width, color):
    turtle.penup()             # Поднять перо
    turtle.goto(x, y)          # Переместить в указанную позицию
    turtle.fillcolor(color)    # Задать цвет заполнения
    turtle.pendown()           # Опустить перо
    turtle.begin_fill()        # Начать заполнение
    for count in range(4):     # Начертить квадрат
        turtle.forward(width)
        turtle.left(90)
    turtle.end_fill()          # Завершить заполнение

def main():
    turtle.setup(SCREEN_WIDTH, SCREEN_HEIGHT)
    turtle.speed(ANIMATION_SPEED)
    turtle.hideturtle()
    color = 'black'

    for y in range(FIRST_Y, LAST_Y, -SQUARE_WIDTH):
        for x in range(FIRST_X, LAST_X, SQUARE_WIDTH):
            square(x, y, SQUARE_WIDTH, color)
            if color == 'black':
                color = 'white'
            else:
                color = 'black'
```

```
    # Не закрывать окно.
    turtle.done()

# Вызвать главную функцию
main()
```

Упражнение по программированию 5.26. Черепашья графика: городской силуэт

```
import turtle
import random

# Именованные константы
ANIMATION_SPEED = 0
SCREEN_WIDTH = 500                  # Ширина экрана
SCREEN_HEIGHT = 500                 # Высота экрана
NUM_STARS = 20                      # Количество рисуемых звезд
MIN_X = -(int(SCREEN_WIDTH / 2))    # Минимальная координата x на экране
MAX_X = int((SCREEN_WIDTH / 2) - 1) # Максимальная координата x на экране
MIN_Y = -(int(SCREEN_HEIGHT / 2))   # Минимальная координата y на экране
MAX_Y = int(SCREEN_HEIGHT / 2)      # Максимальная координата y на экране

X1 = MIN_X          # Координата x начала горизонта
Y1 = -50            # Координата y начала горизонта

WINDOW_SIZE = 10    # Размер квадратного окна
WX1 = -160          # Координата x окна 1
WY1 = 10            # Координата y окна 1
WX2 = -100          # Координата x окна 2
WY2 = 170           # Координата y окна 2
WX3 = -100          # Координата x окна 3
WY3 = 150           # Координата y окна 3
WX4 = -60           # Координата x окна 4
WY4 = 100           # Координата y окна 4
WX5 = -80           # Координата x окна 5
WY5 = -20           # Координата y окна 5
WX6 = 30            # Координата x окна 6
WY6 = 90            # Координата y окна 6

# Функция square чертить квадрат.
# Параметры x и y являются координатами левого нижнего угла.
# Параметр width является шириной каждой стороны.
# Параметр color является цветом заполнения, как строковое значение.

def square(x, y, width, color):
    turtle.penup()            # Поднять перо
    turtle.goto(x, y)         # Переместить в указанную позицию
    turtle.fillcolor(color)   # Задать цвет заполнения
    turtle.pendown()          # Опустить перо
    turtle.begin_fill()       # Начать заполнение
```

```
    for count in range(4):    # Начертить квадрат
        turtle.forward(width)
        turtle.left(90)
    turtle.end_fill()         # Завершить заполнение

def screen_setup():
       # Настроить экран
    turtle.setup(SCREEN_WIDTH, SCREEN_HEIGHT)
    turtle.speed(ANIMATION_SPEED)
    turtle.hideturtle()
    turtle.bgcolor('black')

def draw_stars():
    # Нарисовать звезды
    turtle.pencolor('white')
    for count in range(NUM_STARS):
        x = random.randint(MIN_X, MAX_X)
        y = random.randint(MIN_Y, MAX_Y)
        turtle.penup()
        turtle.goto(x, y)
        turtle.pendown()
        turtle.dot()

def draw_buildings():
    # Нарисовать здания
    turtle.pencolor('gray')
    turtle.fillcolor('gray')
    turtle.begin_fill()
    turtle.penup()
    turtle.goto(X1, Y1)
    turtle.pendown()
    turtle.setheading(0)
    turtle.forward(80)
    turtle.left(90)
    turtle.forward(100)
    turtle.right(90)
    turtle.forward(60)
    turtle.left(90)
    turtle.forward(150)
    turtle.right(90)
    turtle.forward(80)
    turtle.right(90)
    turtle.forward(180)
    turtle.left(90)
    turtle.forward(50)
    turtle.left(90)
    turtle.forward(100)
    turtle.right(90)
    turtle.forward(60)
```

```
    turtle.right(90)
    turtle.forward(60)
    turtle.left(90)
    turtle.forward(40)
    turtle.right(90)
    turtle.forward(70)
    # Перейти в правый угол экрана.
    turtle.goto(MAX_X, turtle.ycor())
    # Закрыть фигуру.
    turtle.goto(MAX_X, MIN_Y)
    turtle.goto(MIN_X, MIN_Y)
    turtle.goto(X1, Y1)
    turtle.end_fill()

def draw_windows():
       # Нарисовать окна
    square(WX1, WY1, WINDOW_SIZE, 'white')
    square(WX2, WY2, WINDOW_SIZE, 'white')
    square(WX3, WY3, WINDOW_SIZE, 'white')
    square(WX4, WY4, WINDOW_SIZE, 'white')
    square(WX5, WY5, WINDOW_SIZE, 'white')
    square(WX6, WY6, WINDOW_SIZE, 'white')

def main():
    screen_setup()
    draw_stars()
    draw_buildings()
    draw_windows()

    # Не закрывать окно.
    turtle.done()

# Вызвать главную функцию
main()
```