ПРАКТИЧЕСКАЯ РАБОТА

Блок задач на тему “Условные конструкции. Цикл for и while.”

- Напишите программу, которая запрашивает у пользователя число и выводит все числа от 1 до введенного числа, кроме числа, которое делится на 3.
- Напишите программу, которая запрашивает у пользователя число и выводит сумму всех чисел от 1 до введенного числа, кроме чисел, которые делятся на 5.
- Напишите программу, которая запрашивает у пользователя число и проверяет, является ли оно простым числом.

Простым числом называется натуральное число, большее 1, которое имеет только два делителя: 1 и само себя

- Напишите программу, которая выводит все простые числа в заданном диапазоне.
- Напишите программу, которая запрашивает у пользователя число и выводит его факториал.

Факториал числа представляет собой произведение всех натуральных чисел от 1 до этого числа. Формула для вычисления факториала числа n записывается следующим образом:
n! = n * (n-1) * (n-2) * ... * 3 * 2 * 1

Дополнительно:

Напишите программу, которая выводит все числа Фибоначчи до заданного числа.

Последовательность Фибоначчи - это числовая последовательность, в которой каждое число является суммой двух предыдущих чисел. Первые два числа последовательности обычно равны 0 и 1.

Например, первые несколько чисел в последовательности Фибоначчи выглядят следующим образом:
F(0) = 0 F(1) = 1 F(2) = F(1) + F(0) = 1 + 0 = 1 F(3) = F(2) + F(1) = 1 + 1 = 2 F(4) = F(3) + F(2) = 2 + 1 = 3 F(5) = F(4) + F(3) = 3 + 2 = 5

Блок задач на тему “Списки”

- Напишите программу, которая создает список случайных чисел и выводит его на экран.

- Напишите программу, которая принимает список чисел и возвращает новый список, содержащий только четные числа, умноженные на 2.

- Напишите программу, которая принимает два списка чисел и возвращает новый список, содержащий элементы, которые присутствуют только в одном из списков.

- Напишите программу, которая принимает список чисел и возвращает новый список, содержащий только числа, которые являются степенями тройки.

Общая задачка :)

Создайте приложение “Змейка” используя библиотеку pygame.

Шаги :

- Импортируем необходимые модули: pygame, time и random. pygame используется для создания игры, time - для управления временными задержками, а random - для генерации случайных чисел.

- Определяем цвета, используемые в игре, в виде RGB-значений.

- Определяем размеры окна игры и создаем окно с помощью pygame.display.set_mode(). Затем устанавливаем заголовок окна с помощью pygame.display.set_caption().

- Создаем объект clock, который будет использоваться для управления скоростью обновления экрана.

```
clock = pygame.time.Clock()
```

- Определяем размер блока змейки (snake_block) и скорость змейки (snake_speed).

- Определяем функцию our_snake, которая отрисовывает змейку на экране. Она принимает параметры snake_block (размер блока змейки) и snake_list (список координат блоков змейки) и использует pygame.draw.rect() для отрисовки каждого блока змейки.

- Определяем функцию gameLoop, которая содержит основной цикл игры. Внутри функции определяются переменные и списки, необходимые для управления змейкой и едой.

```
def gameLoop():
    game_over = False
    game_close = False

    x1 = dis_width / 2
    y1 = dis_height / 2

    x1_change = 0
    y1_change = 0

    snake_List = []
    Length_of_snake = 1

    foodx = round(random.randrange(0, dis_width - snake_block) / 10.0) * 10.0
    foody = round(random.randrange(0, dis_height - snake_block) / 10.0) * 10.0
```

- Начинаем основной цикл игры, который будет выполняться, пока переменная game_over равна False.

```
while not game_over:
```

- Обрабатываем события, такие как нажатия клавиш. Если нажата клавиша со стрелкой, меняем направление движения змейки.

- Обновляем позицию змейки и отрисовываем ее на экране:
  - проверяем, достигла ли змейка границ игрового окна. Если голова змейки (x1, y1) выходит за границы, то переменная game_close устанавливается в значение True, что означает, что игра должна быть завершена.
  - обновляем положение головы змейки на основе изменений координат x и y (x1_change и y1_change).
  - заполняем всё игровое окно синим цветом, эффективно очищая экран для следующего кадра(wind.fill()).
  - рисуev еду (представленную зелёным прямоугольником) в позиции, указанной foodx и foody, с размером snake_block на snake_block.

- Эти строки создают новый сегмент для тела змейки (голову), добавляя текущее положение головы змейки (x1, y1) в snake_List.

```
snake_Head = []
snake_Head.append(x1)
snake_Head.append(y1)
snake_List.append(snake_Head)
```

- Это условное выражение проверяет, превысила ли длина тела змейки необходимую длину (Length_of_snake). Если да, то первый сегмент тела змейки удаляется, эффективно имитируя движение змейки.

```
if len(snake_List) > Length_of_snake:
    del snake_List[0]
```

- Отрисовываем змейку на экране и обновляем экран.

```
our_snake(snake_block, snake_List)
pygame.display.update()
```

- Проверяем, не съела ли змейка еду. Если съела, генерируем новые координаты для еды и увеличиваем длину змейки.

- Ограничиваем скорость обновления экрана с помощью clock.tick().

```
clock.tick(snake_speed)
```