**Множественное ветвление в Python для обучающихся 8 классов на примере алгоритма игры «Камень, ножницы, бумага»**

Бритун Евгения Евгеньевна, учитель информатики

МБОУ ООШ №10 им.Атамана Головатого

г. Геленджик, с. Марьина Роща

В программировании, особенно в языке Python, одна из ключевых концепций – это управление потоком выполнения программы. Одним из важнейших инструментов для этого является конструкция множественного ветвления, которая позволяет программам принимать решения в зависимости от различных условий.

Что такое множественное ветвление?

Множественное ветвление – это способ управления потоком выполнения программы, который позволяет программе принимать одно из нескольких возможных действий в зависимости от условий. В Python для этого используется конструкция if, elif и else.

Логика выполняющейся программы может быть сложнее, чем выбор одной из двух ветвей. Например, в зависимости от значения той или иной переменной, может выполняться одна из трех (или более) ветвей программы. Как организовать такое множественное ветвление? Наверное, можно использовать несколько инструкций if: сначала проверяется условное выражение в первой инструкции if (если оно возвращает истину, то будет выполняться вложенный в нее блок кода), затем во второй инструкции if и т.д. Однако при таком подходе проверка последующих инструкций будет продолжаться даже тогда, когда первое условие было истинным, и блок кода при данной ветке был выполнен. Проверка последующих условий может оказаться бессмысленной.

Обычно такую проблему можно решить с помощью вложенных конструкций if-else. Однако при этом часто появляется проблема правильной трактовки кода: непонятно, к какому if относится else (хотя в Python такая путаница не возможна из-за обязательных отступов).

С другой стороны, в ряде языков программирования, в том числе и Python, предусмотрено специальное расширение инструкции if, позволяющее направить поток выполнения программы по одной из множества ветвей. Данная расширенная инструкция, помимо необязательной части else, содержит ряд ветвей elif (сокращение от "else if" - "еще если") и выглядит примерно так, как показано на блок-схеме. Частей elif может быть сколь угодно много (в пределах разумного, конечно), как показано на рисунке 1.

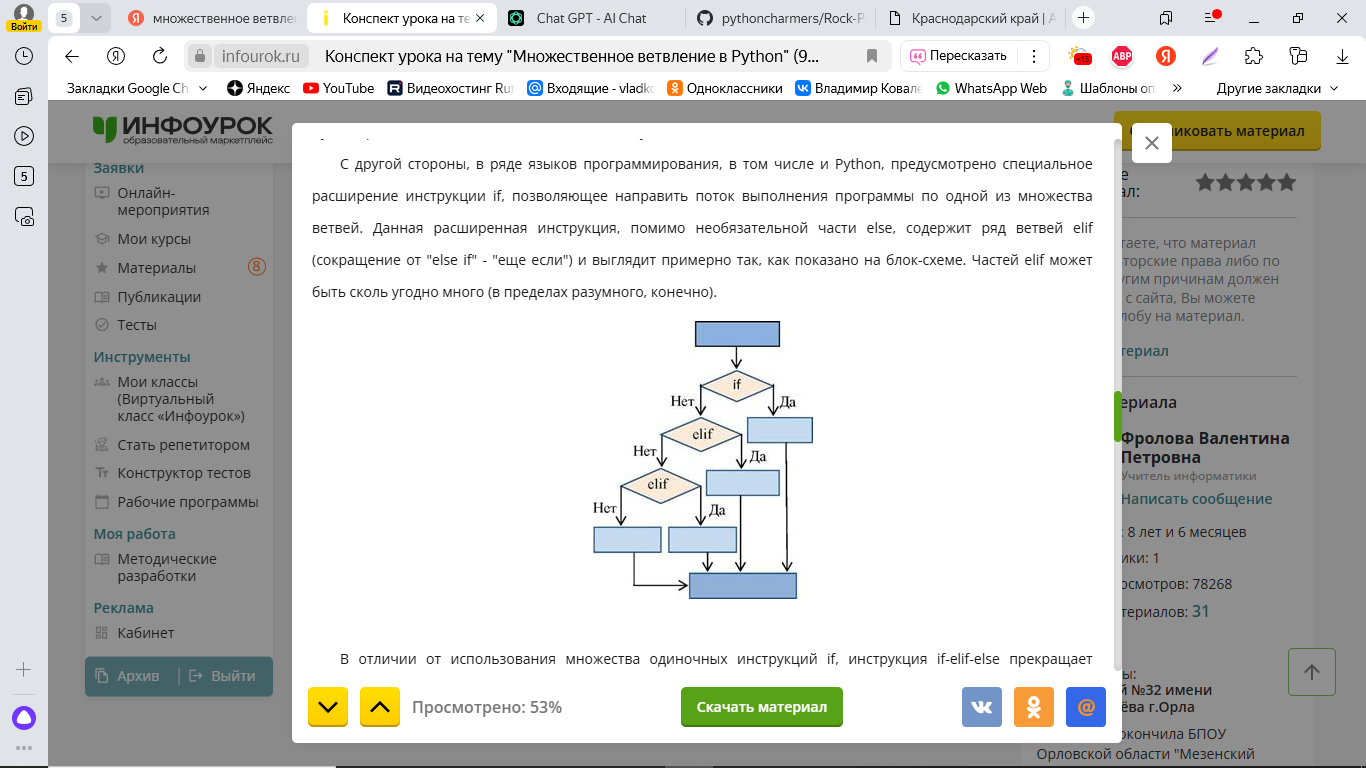


Рисунок 1 – Блок-схема алгоритма

В отличии от использования множества одиночных инструкций if, инструкция if-elif-else прекращает просмотр последующих ветвей, как только логическое выражение в текущей ветке вернет true. Например, если выражение при if (первая ветка) будет истинным, то после выполнения вложенного блока выражений, программа вернется в основную ветку.

Для практической отработки алгоритма с множественным ветвлением предлагается следующая программа игры «Камень, ножницы, бумага», который наглядно показывает работу конструкций ветвления.

#импорт библиотеки

import random

#список

com = ["Камень", "Бумага", "Ножницы"]

#алгоритм игры

def battle(player1,player2):

if player1.lower() == player2.lower(): #.lower читает маленькие буквы

print ("Computer: "+ player2) #показывает выбор компьютером

print ("Ничья!")

elif player1.lower() == "Камень".lower() and player2 == "Бумага":

print ("Computer: "+ player2)

print ("Ты проиграл!")

elif player1.lower() == "Камень".lower() and player2 == "Ножницы":

print ("Computer: "+ player2)

print ("Ты выиграл!")

elif player1.lower() == "Бумага".lower() and player2 == "Камень":

print ("Computer: "+ player2)

print ("Ты выиграл!")

elif player1.lower() == "Бумага".lower() and player2 == "Ножницы":

print ("Computer: "+ player2)

print ("Ты проиграл!")

elif player1.lower() == "Ножницы".lower() and player2== "Камень":

print ("Computer: "+ player2)

print ("Ты проиграл!")

elif player1.lower() == "Ножницы".lower() and player2== "Бумага":

print ("Computer: "+ player2)

print ("Ты выиграл!")

else:

print ("===Опечатка!===") #покажет, если в слове опечатка

#запуск программы

run = True

while run:

player1 = input("Введите свой выбор\nВы: ") #введите один из 3х вариантов

player2 = random.choice(com) #компьютер выберет рандомный вариант себе

battle(player1,player2) #покажет функцию

Результат работы программы показан на рисунке 2.

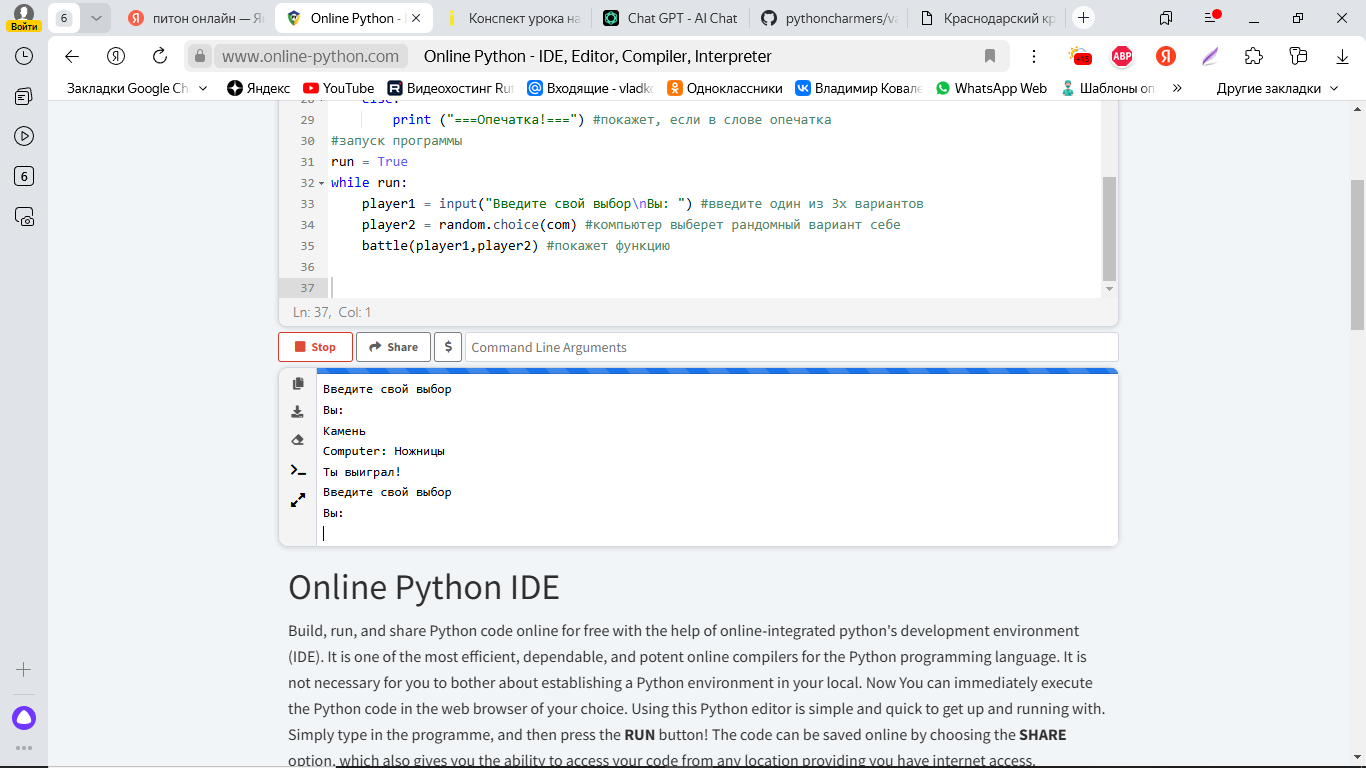


Рисунок 2 – Работа программы «Камень, ножницы, бумага»

Множественное ветвление – это один из основополагающих элементов программирования в Python, который позволяет создавать гибкие и многофункциональные программы. Освоение этой концепции открывает двери к более сложным алгоритмическим задачам и улучшает аналитические способности студентов. Практика через решение разнообразных задач поможет закрепить полученные знания и развить навыки программирования.