План-конспект урока информатики

Тема: «Циклические алгоритмы»

**Цель урока:** познакомить обучающихся с понятием циклических алгоритмов на языке программирования Python

**Задачи урока:**

* Рассмотреть основные виды циклов в языке программирования Python (for и while).
* Научиться правильно использовать циклы для решения задач.

**План-конспект**

*Теоретическая часть*

*1. Что такое цикл?*

Циклом называется последовательность операций, которая выполняется многократно до тех пор, пока выполняется некоторое условие. Циклы используются для автоматизации выполнения одних и тех же действий большое количество раз. Это позволяет значительно сократить объём кода и упростить решение многих задач.

*2.Применение циклов в программировании*

Примеры, где используют циклы:

* Печать последовательностей чисел.
* Поиск элементов в массиве.
* Выполнение однотипных операций над большими наборами данных.

Преимущества циклов перед повторением действий вручную:

* Сокращение объёма кода.
* Повышение надёжности программы (меньше ошибок при наборе одинаковых команд и действий).
* Упрощение изменения и поддержки кода.

*3.Виды циклов*

for i in range(n):

# команды

Цикл for

Синтаксис:

Цикл «for» предназначен для перебора элементов последовательности (списка, строки, диапазона и т.п.). На каждом шаге цикла переменная «i» принимает значение очередного элемента последовательности.

Пример использования:

numbers = [1, 2, 3, 4, 5]

for num in numbers:

print(num)

Этот код выведет на экран все числа списка «numbers».

Функция range() в Python используется для генерации числовой последовательности. Она особенно полезна при работе с циклами, такими как for, для перебора элементов последовательности. Давайте рассмотрим её подробнее.

Синтаксис:

range(start, stop, step)

Аргументы:

start — начальное значение последовательности. По умолчанию равно 0.

stop—конечное значение последовательности. Последовательность останавливается до достижения этого значения (не включительно).

step — шаг увеличения или уменьшения последовательности. По умолчанию 1.

Примеры:

|  |  |
| --- | --- |
| Базовый случай:  for i in range(5):  print(i) | Результат:  0  1  2  3  4 |

Здесь range(5) генерирует последовательность от 0 до 4 (включительно).

|  |  |
| --- | --- |
| Указание начального и конечного значения:  for i in range(2, 10):  print(i) | Результат:  2  3  4  5  6  7  8  9 |

Здесь range(2, 10) генерирует последовательность от 2 до 9 (включительно).

|  |  |
| --- | --- |
| Использование шага:  for i in range(0, 10, 2):  print(i) | Результат:  0  2  4  6  8 |

Здесь range(0, 10, 2) генерирует последовательность от 0 до 8 с шагом 2.

|  |  |
| --- | --- |
| Отрицательные шаги:  for i in range(10, 0, -1):  print(i) | Результат:  10  9  8  7  6  5  4  3  2  1 |

Здесь range(10, 0, -1) генерирует последовательность от 10 до 1 (включительно) с шагом -1.

Последовательность всегда заканчивается на одно значение меньше, чем указано в аргументе stop. По умолчанию: если указан только один аргумент, он считается значением stop, а start принимается равным 0.

Функция range() — мощный инструмент для работы с последовательностями чисел в Python. Её гибкость и эффективность делают её незаменимой при работе с циклами и перебором элементов.

Цикл while

while condition:

# команды

Синтаксис:

Цикл «while» продолжает выполняться, пока условие «condition» остаётся истинным. Важно следить за изменением условия, чтобы избежать бесконечного цикла.

Пример использования:

i = 1

while i <= 5:

print(i)

i += 1

Этот код напечатает числа от 1 до 5.

*4. Сравнение циклов «for» и «while»*

Когда лучше использовать «for»:

* Для перебора известных последовательностей (списки, строки, диапазоны).
* Когда заранее известно количество итераций.

Когда лучше использовать «while»:

* Когда неизвестно точное количество итераций.
* Когда нужно проверять сложное условие.

*Практическая часть*

1. Задачи на цикл «for»

# Задача 1: Напечатать все числа от 1 до 10

Решение:

for i in range(1, 11):

print(i)

# Задача 2: Посчитать сумму всех чисел от 1 до N

Решение:

N = int(input("Введите число N: "))

total\_sum = 0

for i in range(1, N+1):

total\_sum += i

print("Сумма чисел от 1 до", N, "равна", total\_sum)

2. Задачи на цикл «while»

# Задача 1: Найти сумму всех нечётных чисел до N

Решение:

N = int(input("Введите число N: "))

i = 1

total\_sum = 0

while i <= N:

if i % 2 != 0:

total\_sum += i

i += 1

print("Сумма нечётных чисел до", N, "равна", total\_sum)

*Закрепление материала*

Вопросы для обсуждения:

1. В каких случаях удобнее использовать цикл «for», а в каких — «while»?

* «for» удобен для перебора объектов, но ограничен заранее известными данными для обработки.
* «while» гибче, но требует больше контроля за условиями завершения цикла.

1. Почему циклы являются важной частью программирования?

* Циклы позволяют автоматизировать повторяющиеся действия, экономить время и уменьшать вероятность ошибок при написании кода. Это делает программы более компактными и удобными для чтения и сопровождения.

Самостоятельная работа:

1. Написать программу, которая выводит все нечётные числа от 1 до 100.

for i in range(1, 101, 2):

print(i)

2. Написать программу, которая считывает число и печатает его факториал.

# Пользователь вводит число

num = int(input("Введите число: "))

# Инициализация переменной для хранения факториала

factorial\_result = 1

# Подсчёт факториала с помощью цикла while

current\_num = num

while current\_num > 1:

factorial\_result \*= current\_num

current\_num -= 1

# Вывод результата

print("Факториал числа", num, "равен", factorial\_result)

*Рефлексия и подведение итогов*

1. Обсуждение успехов и трудностей.

2. Ответы на вопросы учащихся.

3. Подведение итогов урока.

*Домашнее задание*

1. Напишите программу, которая запрашивает у пользователя натуральное число N и затем вычисляет сумму всех целых чисел от 1 до N. Используйте цикл while для решения этой задачи.

N = int(input("Введите число N: "))

# Инициализируем переменную для хранения суммы

total\_sum = 0

# Используем цикл while для суммирования чисел от 1 до N

i = 1

while i <= N:

total\_sum += i

i += 1

# Выводим результат

print("Сумма чисел от 1 до", N, "равна", total\_sum)

2. Последовательность Фибоначчи определяется следующим образом: первые два числа равны 1 и 1, каждое следующее число равно сумме двух предыдущих. Напишите программу, которая выводит все числа Фибоначчи, меньшие или равные данному числу N используя цикл for.

# Запрашиваем у пользователя число N

N = int(input("Введите максимальное число N: "))

# Начальные значения для первых двух чисел Фибоначчи

a, b = 0, 1

# Используем цикл for для вывода чисел Фибоначчи

print("Числа Фибоначчи до", N, ":")

while b <= N:

print(b, end=' ')

a, b = b, a + b