Назначение лекции

Вы узнаете, в каких случаях и как применяется инструкция if и варианты ее использования.

Новые слова

**Инструкция if -** позволяет вашей программе принимать решения и изменять тот путь, по которому происходит передача управления в вашей программе.

# Инструкция if или программа немного думает.

Использование инструкции if заставляет компьютер выбирать одно решение из двух или более вариантов. Это обычная ситуация в реальной программе.

Варианты, из которых требуется выбирать часто зависят от внешних данных. Например, какие данные пользователь вводит с клавиатуры.

Таким образом программа, как бы сама принимает решение, но оно зависит от входных данных. Именно инструкция if позволяет задать такой «умный» механизм действий.

**Примеры на Phyton**

if num>10:

print("Введено число больше 10")

else:

print("Введено число меньше 10 или равно 0")

Читается этот код следующим образом:

**Если** num больше 10, то выдать на экран сообщение "Число больше 10"

**иначе** выдать на экран сообщение "Число меньше 10 или равно 0"

if num > 10:

print ("Число больше 10")

elif num == 10:

print ("Число равно 10")

else:

print ("Число меньше 10")

Читается этот код следующим образом:

**Если** num больше 10, то выдать на экран сообщение "Число больше 10"

**иначе** **если** num равна 10, то выдать на экран сообщение "Число равно 10"

**иначе** выдать на экран сообщение "Число меньше 10"

Допустимы и следующие варианты:

if num > 10: print ("Число больше 10")

else: print ("Число меньше или равно 10")

или

if num > 10: print ("Число больше 10")

В этих примерах переменная ***num*** неопределена. И если попытаться выполнить этот код, то компьютер выдаст сообщение об ошибке. Например, «Ошибка: переменная num не определена» (NameError: name 'num' is not defined).

Из первого занятия уже известно как определять переменную или выполнять ввод в эту переменную данных с помощью клавиатуры. Например,

num=11 #в переменую num записали 11

if num > 10: print ("Число больше 10")

elif num == 10: print ("Число равно 10")

else: print ("Число меньше 10")

или

#в переменую num введем любое целое число

num=int(input("Введите любое целое число: "))

if num > 10: print ("Число больше 10")

elif num == 10: print ("Число равно 10")

else: print ("Число меньше 10")

Часто такой код записывают несколько иначе, с отступами. Отступы играют важную роль в Phyton. Они разделяют команды, т.к. в отличии от с++ отсутствует ; в конце команды.

Отступ создается клавишей tab или четыре подряд пробела. Например,

#в перенную num введем любое целое число

num=int(input("Введите любое целое число: "))

if num > 10:

print ("Число больше 10")

elif num == 10:

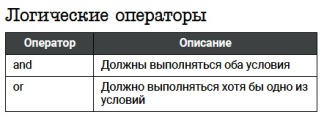
print ("Число равно 10")

else:

print ("Число меньше 10")

Каждая конструкция if содержит условие. Например, num > 10. Это условие может выполнятся или нет. Условие может быть и более сложным и содержать другие операторы сравнения.

Примеры операторов сравнения:

Еще примеры кода:

num=int(input("Введите любое целое число: "))

if num >= 10:

if num <= 20:

print("Число в диапазоне от 10 до 20")

else:

print("Число больше 20")

else:

print("Число меньше 10")

Этот варианте в котором одна инструкция if содержит другое условие (красным) и читается так:

если значение **num** больше или равно 10, выполняется другая команда **if**, которая проверяет, что значение **num** меньше или равно 20. В этом случае выводится сообщение «Число в диапазоне от 10 до 20». Если же значение **num** не меньше или равно 20, выводится сообщение «Число больше 20». Если **num** не больше 10, выводится сообщение «Число меньше 10».

num = int(input("Введите целое число в диапазоне от 10 до 20: "))

if num >= 10 and num <= 20:

print("Отлично")

else:

print("Вне диапазона")

Оператор **and** используется для проверки нескольких условий в инструкции **if**. Для вывода сообщения **"Отлично"** должны выполняться оба условия.

num = int(input("Введите целое число 10 или 20: "))

if num == 10 or num == 20:

print("Отлично")

else:

print("Ошибка")

Оператор **or** используется для проверки нескольких условий в инструкции **if**. Для вывода сообщения **"Отлично"** должны либо одно либо другое условие (10 или 20).

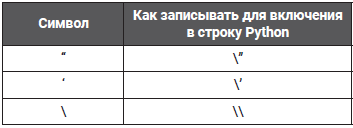
# Строки

Строка — термин для обозначения текста. Чтобы определить блок кода как строку, необходимо заключить его в двойные кавычки (") или в одинарные кавычки ('). Неважно, какой из способов вы выберете, главное — действовать последовательно.

При включении некоторых символов в строки необходимо действовать особенно осторожно. Прежде всего речь идет о таких: " ' \

Дело в том, что эти символы имеют особый смысл в Python, и при использовании их в строках может возникнуть путаница.

Если вы хотите использовать один из этих символов в строке, поставьте перед ним символ \; тогда Python будет знать, что специальный смысл этого символа нужно игнорировать, и будет рассматривать его как обычный текст, который нужно вывести на экран.



# Строки и числа как переменные

Если вы определяете переменную как строку (даже если в ней хранятся только представления чисел в текстовом виде), такую переменную нельзя будет позднее использовать в математических вычислениях. Для этого нужно преобразовать строку в число перед использованием.

num = input("Введите число: ")

total = num + 10

print(total)

В этом примере запрашивается число, но не определяется как числовое значение, и при попытке выполнить программу произойдет ошибка, например:

TypeError: can only concatenate str (not "int") to str

Хотя это сообщение об ошибке выглядит устрашающе, оно просто объясняет, что строка **total = num + 10** не работает, потому что переменная **num** содержит строковые данные.

Проблему можно решить двумя способами. Переменная либо определяется как число при исходном создании переменной:

num = int(input("Введите число: "))

либо преобразуется в число уже после создания:

num = int(num)

То же самое может происходить со строками.

name = input("Введите имя: ")

num = int(input("Введите число: ")

ID = name + num

print(ID)

В этой программе пользователю предлагается ввести имя и номер. Предполагается, что эти данные будут соединены и образуют идентификатор; при использовании со строками знак **+** выполняет операцию **конкатенации**. При выполнении этого кода вы получите почти такое же сообщение об ошибке, например:

invalid syntax

Чтобы обойти эту проблему, либо изначально не определяйте переменную как число, либо преобразуйте ее в строку позднее:

name = input("Enter a name: ")

num = int(input("Enter a number: "))

ID = name + str(num)

print(ID)

# Внутренние разрывы строк

Если вы хотите определить строку, которая состоит из нескольких внутренних (логических) строк, либо используйте символ новой строки (**\n**), либо заключите всю строку в тройные кавычки (этот способ сохраняет форматирование текста).

address="""Бишкек, Чуй

Дом 28

Кв. 65"""

print(address)

или

address="Бишкек, Чуй\nДом 28\nКв. 65"

print(address)

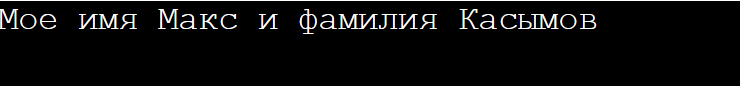
# F строки

**F строки** — это строки которые в которые включены параметры.

name="Макс"  
fam="Касымов"  
print(f"Мое имя {name} и фамилия {fam}")

Описание структуры: f – признак строки. Весь текст заключен в двойные кавычки. Пара фигурных скобок заключает параметр – это переменная, которой заранее присвоили значение (любого допустимого типа)

Результат:



Пока все