Назначение лекции

Новые слова

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

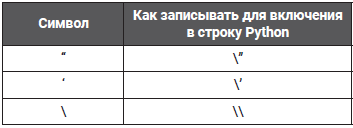
# Строки

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)

# Функции для строк

if msg.**isupper**():

print("Uppercase")

else:

print("This is not in uppercase")

Если строка msg состоит из букв верхнего регистра, выводится сообщение «Uppercase», в противном случае выводится сообщение «This is not in uppercase». Это может использоваться, например, при формировании сложного пароля.

msg.**islower**()

Может использоваться вместо функции isupper**()** для проверки того, что переменная msg состоит из букв нижнего регистра.

msg="Hello"

for letter in msg:

print(letter,end="\*")

Выводит сообщение, при этом после каждого символа выводится символ **\***. В данном примере будет выведена следующая строка: **H\*e\*l\*l\*o\***

Служебные символы в строке

\ позволяет добавить внутрь строки слеш

\' позволяет добавить внутрь строки одинарную кавычку

\" позволяет добавить внутрь строки двойную кавычку

\n осуществляет переход на новую строку

\t добавляет табуляцию (4 отступа)

string = "hello world"

c0 = string[0] # h

print(c0)

c6 = string[6] # w

print(c6)

c11 = string[11] # ошибка IndexError: string index out of range

print(c11)

Можно обратиться к отдельным символам строки по индексу в квадратных скобках. Индексация начинается с нуля, поэтому первый символ строки будет иметь индекс 0. А если мы попытаемся обратиться к индексу, которого нет в строке, то мы получим исключение IndexError. Например, в случае выше длина строки 11 символов, поэтому ее символы будут иметь индексы от 0 до 10.

string = "hello world"

c1 = string[-1] # d

print(c1)

c5 = string[-5] # w

print(c5)

Чтобы получить доступ к символам, начиная с конца строки, можно использовать отрицательные индексы. Так, индекс -1 будет представлять последний символ, а -2 - предпоследний символ и так далее.

string = "hello world"

string[0] = "H"

При работе с символами следует учитывать, что строка - это неизменяемый (immutable) тип, поэтому если мы попробуем изменить какой-то отдельный символ строки, то мы получим ошибку, как в данном случае. Мы можем только полностью переустановить значение строки, присвоив ей другое значение: string = "Hello world"

string = "hello world"

for char in string:

print(char)

С помощью цикла for можно перебрать все символы строки string.

string[:end]

Извлекается последовательность символов начиная с 0-го индекса по индекс end (не включая).

string[start:end]

Извлекается последовательность символов начиная с индекса start по индекс end (не включая)

string[start:end:step]

Извлекается последовательность символов начиная с индекса start по индекс end (не включая) через шаг step

name = "Tom"

surname = "Smith"

fullname = name + " " + surname

print(fullname) # Tom Smith

Для объединения строк применяется операция сложения.

name = "Tom"

age = 33

info = "Name: " + name + " Age: " + str(age)

print(info) # Name: Tom Age: 33

Если нам надо сложить строку и число? В этом случае необходимо привести число к строке с помощью функции str().

print("a" \* 3) # aaa

print("he" \* 4) # hehehehe

Повторение строк.

str1 = "Tom"

str2 = "tom"

print(str1 == str2) # False - строки не равны

print(str1.lower() == str2.lower()) # True

Зависимость от регистра не всегда желательна, так как по сути мы имеем дело с одинаковыми строками. В этом случае перед сравнением мы можем привести обе строки к одному из регистров. Функция lower() приводит строку к нижнему регистру, а функция upper() - к верхнему.

print(ord("A")) # 65

Поскольку строка содержит символы Unicode, то с помощью функции ord() мы можем получить числовое значение для символа в кодировке Unicode.

string = "hello world"

length = len(string)

print(length) # 11

Для получения длины строки можно использовать функцию len().

string = "hello world"

exist = "hello" in string

print(exist) # True

exist = "sword" in string

print(exist) # False

С помощью выражения term in string можно найти подстроку term в строке string. Если подстрока найдена, то выражение вернет значение True, иначе возвращается значение False.

str=' Привет '

print(str.strip())

Удаляет начальные и конечные пробелы в строке

s = 'A string with white space'  
str=s.replace(' ', '')  
print(str) *#Astringwithwhitespace*

Удаляет все пробелы из строки.

'123a'.isnumeric() # False

'123'.isnumeric() # True

Проверка, что в строке только цифры.

str = "Пример строки Python"

print(str.split()) # ['Пример', 'строки', 'Python']

Разделить строку на подстроки в виде списка

# F строки

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

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

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)

name = 'Адилет'  
age = '20'  
greeting = "Привет, {}! Тебе {} лет.".format(name, age)  
print(greeting) #Привет, Адилет! Тебе 20 лет.

Использование форматирования строки greeting. Фигурные скобки содержат name и age.

Пока все