

Стать IT-шником может каждый



Python Summary 2



Преподаватель



ИМЯ ФАМИЛИЯ

Текущая должность

Количество лет опыта

Какой у Вас опыт - ключевые кейсы

Самые яркие проекты

Дополнительная информация по вашему усмотрению

[Корпоративный e-mail](#)

[Социальные сети \(по желанию\)](#)

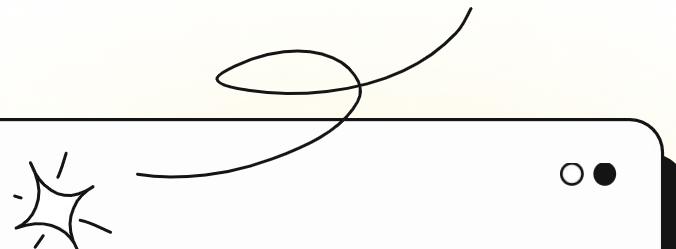


ВАЖНО!

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить микрофон или демонстрацию экрана по просьбе преподавателя.

ПЛАН ЗАНЯТИЯ

- Повторение изученного за неделю
- Вопросы по повторению
- Разбор домашних заданий
- Вопросы по разбору заданий
- Задание для закрепления
- Оставшиеся вопросы



Стать IT-шником может каждый



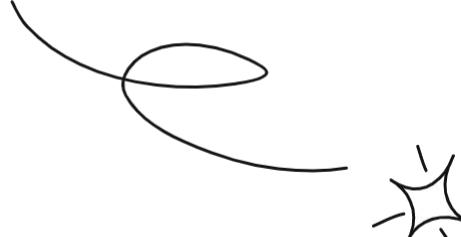
ПОВТОРЕНИЕ ИЗУЧЕННОГО





Цикл в программировании —

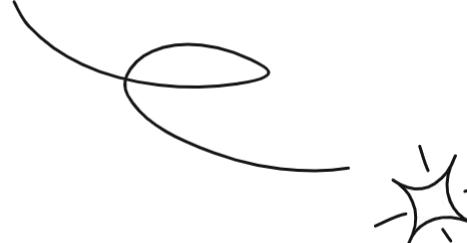
конструкция, которая позволяет выполнять определенный блок кода несколько раз. Полезны, когда требуется выполнить однотипные или повторяющиеся задачи.





Цикл с предусловием —

это цикл с предусловием, то есть он выполняет блок кода, пока условие истинно.



ПРИМЕР

```
while условие:  
    # выполняемые действия
```

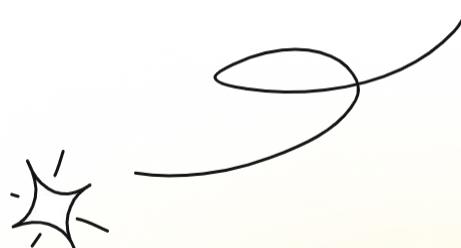
ПРИМЕР

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Стать IT-шником может каждый



Экспресс-опрос



Для чего нужна инструкция break?



Инструкция break —

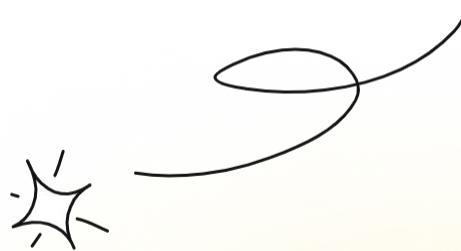
используется для немедленного прерывания выполнения цикла. Когда инструкция `break` выполняется, выполнение цикла немедленно прекращается, и управление передается за пределы цикла.



ПРИМЕР

```
count = 0
while count < 10:
    if count == 5:
        break
    print(count)
    count += 1
```

Экспресс-опрос



Какая инструкция используется для перехода к следующей итерации цикла, игнорируя оставшуюся часть текущей итерации?



Инструкция `continue` —

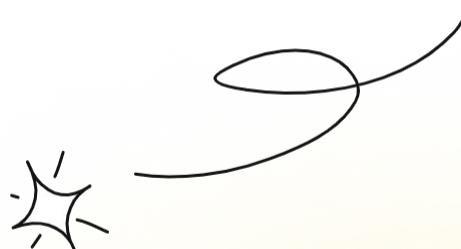
используется для перехода к следующей итерации цикла, игнорируя оставшуюся часть текущей итерации. Когда инструкция `continue` выполняется, выполнение цикла переходит к следующей итерации без выполнения оставшегося кода в текущей итерации.



ПРИМЕР

```
count = 0
while count < 5:
    count += 1
    if count == 3:
        continue
    print(count)
```

Экспресс-опрос



Что позволяет выполнить конструкция `else` после цикла `while`?



Использование else после while



- Конструкция `else` после цикла `while` позволяет выполнить блок кода, когда условие цикла становится ложным.
- Если цикл завершился нормально (без прерывания с помощью `break`), то код в блоке `else` будет выполнен.



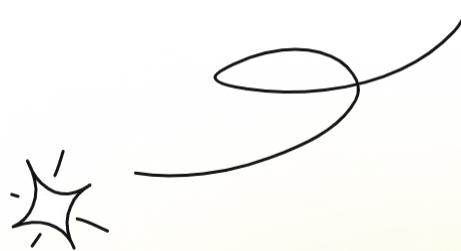
ПРИМЕР

```
count = 0
while count < 5:
    print(count)
    count += 1
else:
    print("Цикл завершен")
```

Стать IT-шником может каждый



Экспресс-опрос



В какой момент прервется бесконечный цикл?



Бесконечный цикл —

это цикл, который будет выполняться бесконечно, пока он не будет прерван с помощью инструкции break или другого условия.



ПРИМЕР

```
while True:  
    # выполняемые действия
```



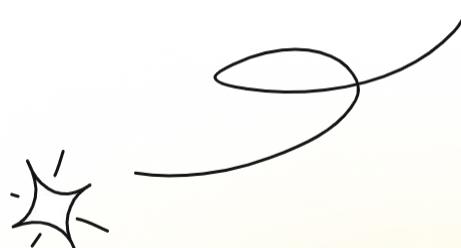
ПРИМЕР

```
while True:  
    user_input = input("Введите число: ")  
    if user_input == "exit":  
        break  
    print(int(user_input) * 2)
```

Стать IT-шником может каждый



Экспресс-опрос



Когда используется ключевое слово pass?



Ключевое слово pass —

используется в Python, когда требуется указать пустой блок кода. pass игнорируется интерпретатором Python и используется для заполнения места, где требуется наличие выражения, но не нужно выполнять никаких действий.



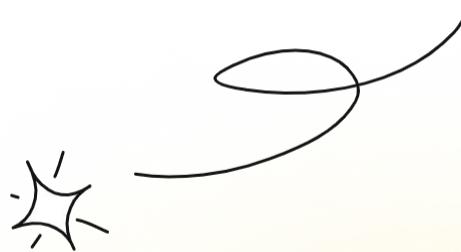
Стать IT-шником может каждый



ПРИМЕР

```
while True:  
    pass
```

Экспресс-опрос



Какие способы замера времени работы программы вы помните?

Способы замера времени работы программы

- Использование функции **time()** из модуля time.
- Команда **%%timeit** в Jupyter Notebook.



ПРИМЕР

```
import time
start_time = time.time()

# выполняемые действия

end_time = time.time()
execution_time = end_time - start_time
print("Время выполнения программы:", execution_time, "секунд")
```

Стать IT-шником может каждый



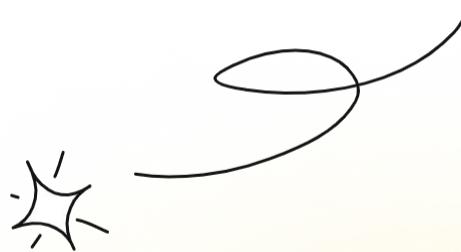
ПРИМЕР

```
%%timeit  
# выполняемые действия
```

Стать IT-шником может каждый



Экспресс-опрос



Что вы помните про модуль random?



Полезно знать



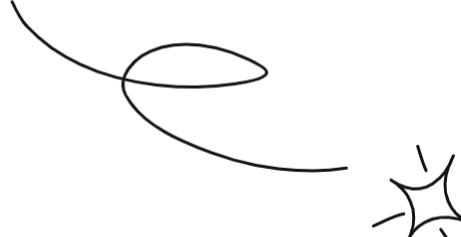
Модуль —

это **отдельные файлы**, содержащие нужные функции.



Модуль random —

дает возможность генерировать случайные числа и
выбирать случайные элементы из списка.



Модуль random

- Чтобы подключить модуль random, необходимо его импортировать с помощью команды **import random**.
- К модулю random можно подключить функции:
 - `random()`: генерирует случайное число от 0.0 до 1.0
 - `randint()`: возвращает случайное число из определенного диапазона
 - `randrange()`: возвращает случайное число из определенного набора чисел
 - `shuffle()`: перемешивает список
 - `choice()`: возвращает случайный элемент списка



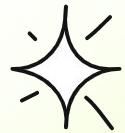
ПРИМЕР

```
import random #подключаем модуль псевдослучайных функций Python
myplaylist = ["Pink Floyd", "Santana", "The Beatles", "ELO", "Mark
Knopfler", "KennyG", "Bob Dylan"] #описание плей-листа
random.shuffle(myplaylist) #перемешиваем список
print(myplaylist) #вывод результата на экран
```

Стать IT-шником может каждый



ВОПРОСЫ



Стать IT-шником может каждый

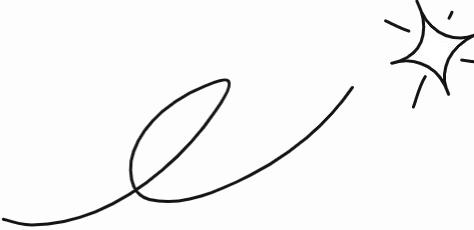


ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ





Напишите код

1. Реализуйте функцию `guess_number()`, которая принимает число и проверяет, равно ли число заданному (пусть это будет 42). Если равно, то функция должна вернуть строку 'You win!', в противном случае нужно вернуть строку 'Try again!'.

```
guess_number( 42 ) # You win!  
guess_number( 61 ) # Try again!
```



Напишите код

2. Реализуйте функцию `flip_flop()`, которая принимает на вход строку и, если эта строка равна 'flip', возвращает строку 'flop'. В противном случае функция должна вернуть 'flip'.

Примеры вызова:

```
print(flip_flop('flip')) # => 'flop'  
print(flip_flop('flop')) # => 'flip'
```



Напишите код

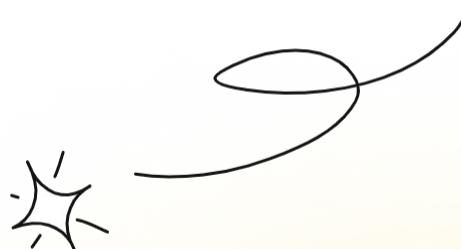
3. Модифицируйте функцию `print_numbers()` так, чтобы она выводила числа в обратном порядке. Для этого нужно идти от верхней границы к нижней. То есть счётчик должен быть инициализирован максимальным значением, а в теле цикла его нужно уменьшать до нижней границы.

Пример вызова и вывода:

```
print_numbers( 4 )
```

```
4
3
2
1
finished!
```

Экспресс-опрос



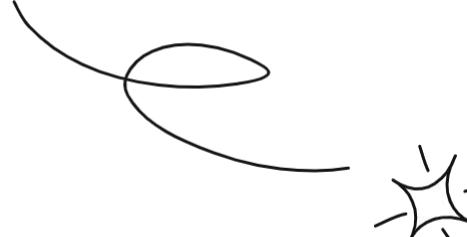
Как вы поняли, чем вещественные числа отличаются от целых чисел?

Приведите пример вещественного числа.



Вещественные числа —

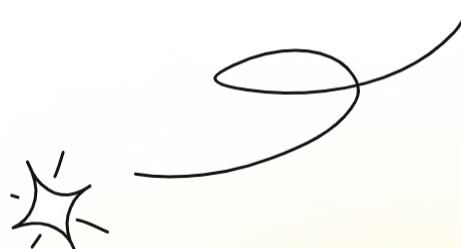
это множество чисел, которое объединяет в себе два больших класса: рациональные (представимые, как отношение натурального и целого m/n) и иррациональные (как корень из 2).



Стать IT-шником может каждый



Экспресс-опрос

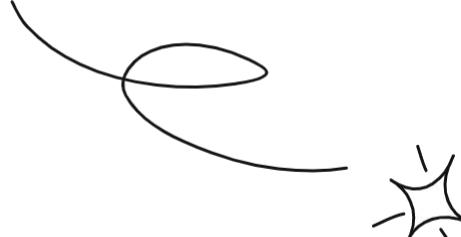


Что такое число с плавающей запятой?



Число с плавающей запятой —

это форма представления дробных чисел, в которой число хранится в форме мантиссы и показателя степени.



Мантисса —

это дробная часть числа или его логарифма.





Экспонента —

это степень, определяет положение запятой и масштаб числа.



Вещественные числа в памяти компьютера

- В Python вещественные числа представляются в памяти компьютера с использованием формата с плавающей запятой.
- Формат с плавающей запятой представляет числа в виде мантиссы и экспоненты.
- Мантисса и экспонента позволяют представить очень большие и очень маленькие числа с высокой точностью.

Вещественные числа в памяти компьютера

В языке Python используется стандарт IEEE 754 для представления вещественных чисел.

Вещественные числа

32-битные (float)

64-битные (double)

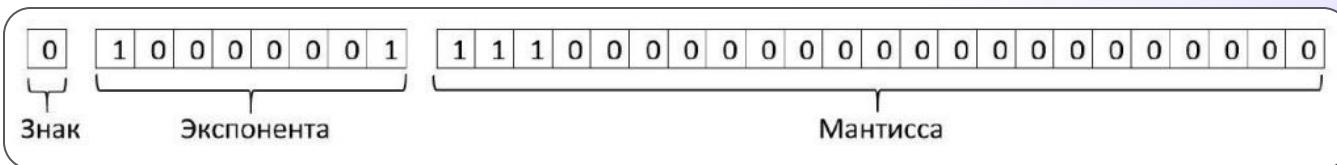
Стать IT-шником может каждый



Вещественные числа в памяти компьютера

- Формат с плавающей запятой использует определенное количество битов для представления мантиссы и экспоненты.
- Точность представления вещественных чисел ограничена и может приводить к небольшим ошибкам округления при выполнении арифметических операций.

Вещественные числа в памяти компьютера



Вещественные числа в памяти компьютера

Знак

Если этот бит равен 0, это означает положительное число, а если он равен 1, это означает отрицательное число.

Экспонента

В 32-битном формате эта группа состоит из 8 битов, а в 64-битном формате - из 11 битов.

Мантисса

В 32-битном формате мантисса занимает 23 бита, а в 64-битном формате - 52 бита.

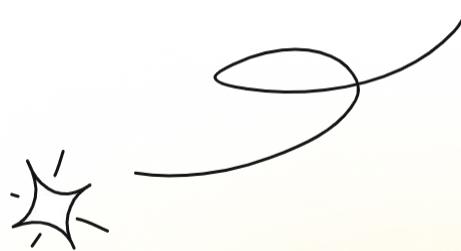
Определение значения числа в формате IEEE 754

- Знак определяется значением бита знака.
- Экспонента вычитается из смещенного значения, чтобы получить истинное значение экспоненты.
- Мантисса интерпретируется как дробное число с фиксированной запятой и складывается с помощью формулы $1.X$, где X - значение мантиссы.
- Результатом является произведение знака, мантиссы и основания экспоненты (2 в степени истинной экспоненты).

Стать IT-шником может каждый



Экспресс-опрос



Помните ли вы алгоритм перевода чисел в формат с плавающей запятой?

Стать IT-шником может каждый



Алгоритм перевода чисел в формат с плавающей запятой

Шаг 1: Приведение числа к нормализованному виду.

Шаг 2: Вычисление экспоненты.

Шаг 3: Представление мантиссы.

Шаг 4: Объединение значений.

Пример перевода в формат с плавающей запятой

Дано: число 44,102

- Переводим в двоичную систему счисления:

101100.00011010000111001011

- Переносим точку на пять знаков влево:

$$+1.0110000011010000111001011 \cdot 2^5$$

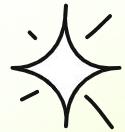
- Экспонента 5, смещение 127, смещенное значение 132 = 1000 0100
- Получаем представление:

0	1	0	0	0	0	1	0	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Стать IT-шником может каждый



ВОПРОСЫ



Стать IT-шником может каждый



ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ





Напишите код

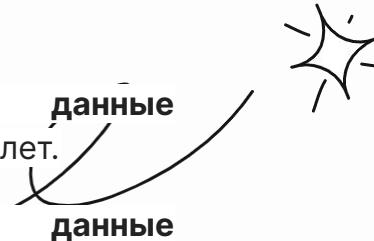
На вход программе подается число N – количество собачьих лет. Напишите программу, которая вычисляет возраст собаки в человеческих годах. В течение первых двух лет собачий год равен 10.5 человеческим годам. После этого каждый год собаки равен 4 человеческим годам.

Входные

На вход программе подается натуральное число – количество собачьих лет.

Выходные

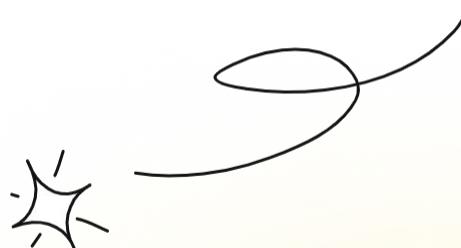
Программа должна вывести возраст собаки в человеческих годах.



Стать IT-шником может каждый

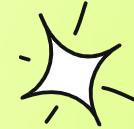


Экспресс-опрос



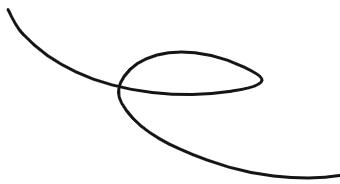
Что такое модуль в Python?





Модуль в Python —

это файл, содержащий определения функций, классов и переменных, которые могут быть использованы в других программах.



Модуль в Python

- Модули позволяют организовать код в логические блоки и повторно использовать его.
- Для использования модуля в программе необходимо выполнить операцию импорта `import this`.

Стать IT-шником может каждый



Экспресс-опрос



Какие варианты import вы помните?

Варианты import

Импорт всего модуля

```
import module_name
```

Импорт модуля с
использованием псевдонима

```
import module_name as alias
```

Импорт определенных
элементов из модуля

```
from module_name import item1, item2
```

Импорт всего модуля

- import module_name

```
import math
print(math.pi)    # Выводит значение числа п из
модуля math
```

Импорт модуля с использованием псевдонима

- import module_name as alias

```
import math as m
print(m.pi) # Выводит значение числа π из модуля
             math, но с использованием псевдонима "m"
```

Импорт определенных элементов из модуля

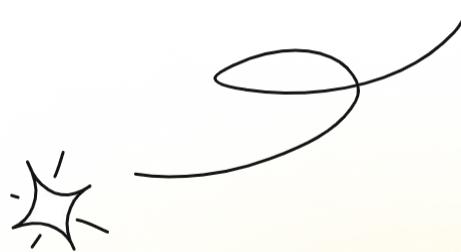
- `from module_name import item1, item2`

```
from math import pi
print(pi)  # Выводит значение числа π из модуля
           # math, но импорт только конкретного элемента "pi"
           # из модуля math
```

Стать IT-шником может каждый



Экспресс-опрос

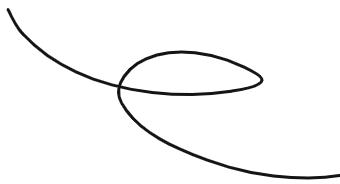


Что такое библиотека math и в чем ее польза?



Библиотека math —

это модуль, который предоставляет функции и константы для работы с математическими операциями.



ФУНКЦИИ В МОДУЛЕ math

- `sqrt()` (квадратный корень)
- `sin()` (синус)
- `cos()` (косинус)
- `log()` (логарифм)...

Большинство функций модуля `math` работают с вещественными числами

Операция импорта math

```
import math
print(math.sqrt(16)) # Выводит: 4.0
print(math.sin(math.pi/2)) # Выводит: 1.0
```

Основные математические функции

`math.sqrt(x)`

`math.pow(x, y)`

`math.exp(x)`

вычисляет квадратный корень из числа x

возводит число x в степень y

вычисляет значение экспоненты e в степени x

`math.log(x)`

`math.sin(x), math.cos(x), math.tan(x)`

вычисляет натуральный логарифм числа x

вычисляют синус, косинус и тангенс угла x в радианах

Округление чисел

`math.ceil(x)`

округляет число x до ближайшего большего целого

`math.floor(x)`

округляет число x до ближайшего меньшего целого

`math.round(x)`

округляет число x до ближайшего целого

Константы

math.pi

представляет значение числа π

math.e

представляет значение числа е
(основание натурального логарифма)

Работа с тригонометрией

math.radians(x)

math.degrees(x)

преобразует значение угла x из градусов в радианы

преобразует значение угла x из радиан в градусы

Дополнительные функции

math.fabs(x)

возвращает абсолютное значение числа x

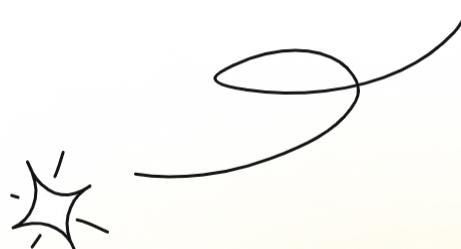
math.factorial(x)

вычисляет факториал числа x

math.isqrt(x)

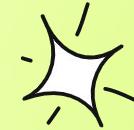
вычисляет целую часть квадратного корня числа x

Экспресс-опрос



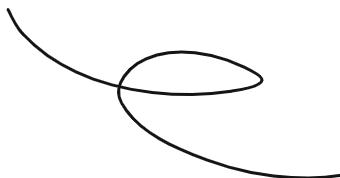
Что такое библиотека decimal и в чем ее польза?





Библиотека decimal —

это модуль, который предоставляет возможность работы с вещественными числами с фиксированной точностью.



Библиотека decimal

Возможность работы с вещественными числами с фиксированной точностью

```
from decimal import Decimal

number1 = Decimal('0.1')
number2 = Decimal('0.2')
result = number1 + number2
print(result) # Выводит: 0.3
```

Библиотека decimal

Возможность выполнения арифметических операций с вещественными числами с высокой точностью и контролем округления.

```
from decimal import Decimal, getcontext

getcontext().prec = 10 # Установка точности на 10
 знаков после запятой
number1 = Decimal('1') / Decimal('7')
print(number1) # Выводит: 0.1428571429
```

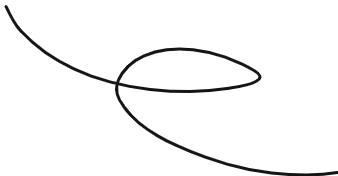
Библиотека decimal

Операция импорта	<code>import decimal</code>
Создание и операции с числами Decimal	<code>Decimal()</code>
Установка точности	<code>decimal.getcontext().prec = n</code>
Округление чисел	<code>decimal.quantize()</code> и <code>decimal.round()</code>
Контексты вычислений	<code>decimal.Context</code>
Конвертация вещественных чисел в Decimal	<code>decimal.Decimal.from_float()</code>
Обработка ошибок и исключений	Возможны ошибки и исключения, связанные с превышением точности, делением на ноль итд



Работа с вещественными числами в формате с фиксированной запятой —

этот **формат** позволяет задать фиксированное количество знаков до и после запятой, обеспечивая точность вычислений в конкретной предметной области.



Стать IT-шником может каждый



Экспресс-опрос

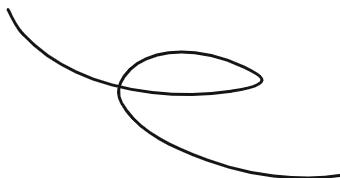


С какой точностью представляют числа класса decimal?



Класс Decimal —

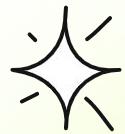
это класс, представляющий вещественные числа с произвольной точностью.



Стать IT-шником может каждый



ВОПРОСЫ



Стать IT-шником может каждый



ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ





Напишите код

Даны длины сторон треугольника. Вычислите площадь треугольника.

Входные

Вводятся три положительных числа.

Выходные

Выведите ответ на задачу.

данные

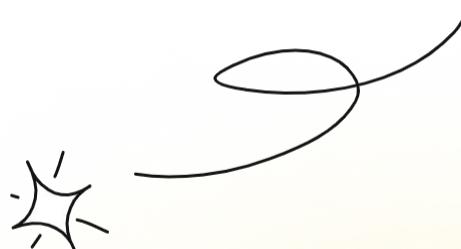
данные



Стать IT-шником может каждый



Экспресс-опрос



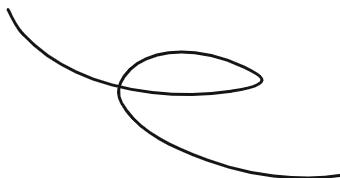
Как вы можете объяснить, что такое строки?





Строки в Python —

это упорядоченная последовательность символов, которые используются для хранения и представления текстовой информации. С их помощью можно описать все, что представлено в текстовой форме.



Пример строк

'Hello'

'Goodbye'

'G'

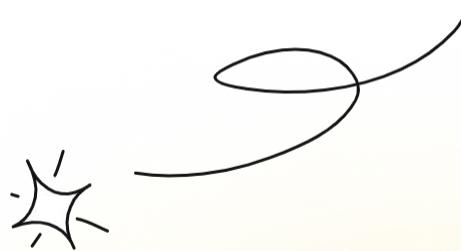
''

"

Стать IT-шником может каждый



Экспресс-опрос



С помощью чего могут быть записаны строки?

Запись строк

Строки в Python могут быть записаны с использованием кавычек:

- одинарных ('')
- двойных ("")
- тройных кавычек (""""")

Строки - неизменяемые, после создания их содержимое не может быть изменено.

Стать IT-шником может каждый



Пример строк

```
name = "John"
```

Стать IT-шником может каждый



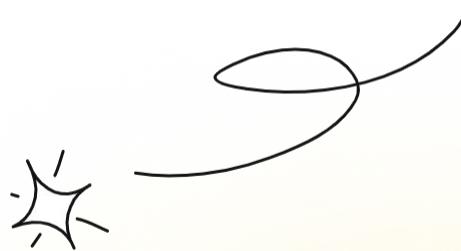
Создание строки с кавычками в тексте

Dragon's mother said "No"

Создание строки с кавычками в тексте

```
# Экранируем кавычки вокруг No, чтобы интерпретатор
# распознал их как часть строки
print("Dragon's mother said \"No\"")
# => Dragon's mother said "No"
```

Экспресс-опрос



С помощью каких функций может быть выполнено приведение типов между строками?



Типы str, int, float, bool

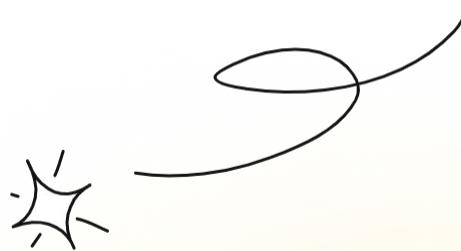
Приведение типов между строками и другими базовыми типами данных в Python может быть выполнено с помощью функций:

- `str()`
- `int()`
- `float()`
- `bool()`

Приведение числа к строке

```
number = 10
number_str = str(number) # Приведение числа к строке
```

Экспресс-опрос



Что позволяют сделать операции сложение и умножение со строками?

Арифметические операции над строками

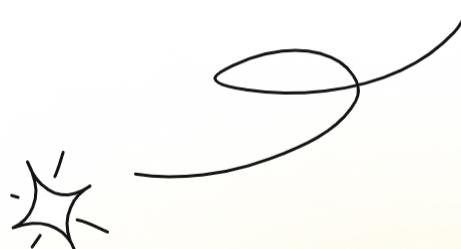
Строки поддерживают операции сложения и умножения.

- Операция сложения позволяет объединять строки
- Операция умножения позволяет повторять строку определенное количество раз

Арифметические операции над строками

```
greeting = "Hello, "
name = "John"
message = greeting + name # Конкатенация строк
multiplied_string = name * 3 # Повторение строки
```

Экспресс-опрос



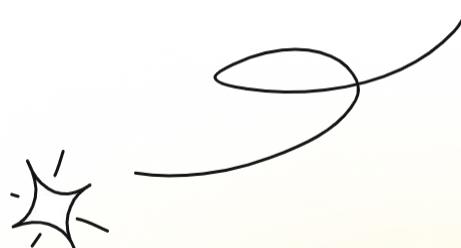
Какая функция используется для вычисления длины строки - количество символов в ней?

Функция len()

В Python функция `len()` используется для вычисления длины строки - количество символов в ней.

```
text = "Hello"  
length = len(text) # Длина строки
```

Экспресс-опрос



Какая функция возвращает числовое представление символа?

А какая выполняет обратное преобразование - возвращает символ по его числовому представлению?

ФУНКЦИИ ord(), chr()

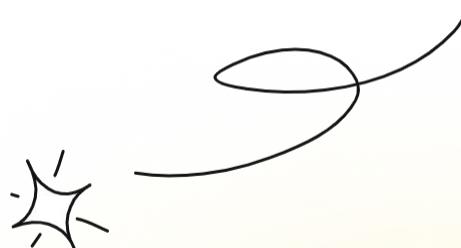
- Функция `ord()` возвращает числовое представление символа.
- Функция `chr()` выполняет обратное преобразование - возвращает символ по его числовому представлению.

```
code = ord('A') # Числовое представление символа 'A'  
character = chr(65) # Символ по числовому представлению 65
```

Стать IT-шником может каждый



Экспресс-опрос



Что хранится в таблице ASCII?





Таблица ASCII —

стандартная [таблица символов](#), где каждому символу соответствует свой уникальный числовый код.

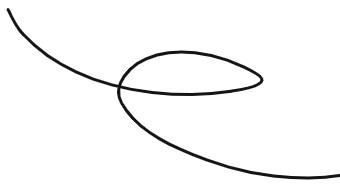


Таблица ASCII

Dec	Hex	Char	Cmd
0	00		NUL
1	01	␀	SOH
2	02	␁	STX
3	03	␂	ETX
4	04	␃	EOT
5	05	␄	ENQ
6	06	␅	ACK
7	07	␆	BEL
8	08	␇	BS
9	09	␈	TAB
10	0A	␉	LF
11	0B	␊	VT
12	0C	␋	FF
13	0D	␌	CR
14	0E	␍	SO
15	0F	␎	SI
16	10	␏	DLE
17	11	␐	DC1
18	12	␑	DC2
19	13	␒	DC3
20	14	␓	DC4
21	15	␔	NAK
22	16	␕	SYN
23	17	␖	ETB
24	18	␗	CAN
25	19	␘	EM
26	1A	␙	SUB
27	1B	␚	ESC
28	1C	␛	FS
29	1D	␜	GS
30	1E	␝	RS
31	1F	␞	US

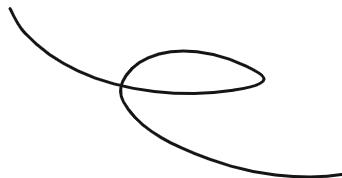
Dec	Hex	Char	Cmd
32	20	!	(sp)
33	21	"	
34	22	#	
35	23	\$	
36	24	%	
37	25	&	
38	26	'	
39	27	(
40	28)	
41	29	*	
42	2A	+	
43	2B	,	
44	2C	-	
45	2D	.	
46	2E	/	
47	2F	0	
48	30	1	
49	31	2	
50	32	3	
51	33	4	
52	34	5	
53	35	6	
54	36	7	
55	37	8	
56	38	9	
57	39	:	
58	3A	;	
59	3B	<	
60	3C	=	
61	3D	>	
62	3E	?	
63	3F	~	

Dec	Hex	Char	Cmd
64	40	@	
65	41	A	
66	42	B	
67	43	C	
68	44	D	
69	45	E	
70	46	F	
71	47	G	
72	48	H	
73	49	I	
74	4A	J	
75	4B	K	
76	4C	L	
77	4D	M	
78	4E	N	
79	4F	O	
80	50	P	
81	51	Q	
82	52	R	
83	53	S	
84	54	T	
85	55	U	
86	56	V	
87	57	W	
88	58	X	
89	59	Y	
90	5A	Z	
91	5B	[
92	5C	\	
93	5D]	
94	5E	^	
95	5F	_	DEL

Dec	Hex	Char	Cmd
96	60	`	
97	61	a	
98	62	b	
99	63	c	
100	64	d	
101	65	e	
102	66	f	
103	67	g	
104	68	h	
105	69	i	
106	6A	j	
107	6B	k	
108	6C	l	
109	6D	m	
110	6E	n	
111	6F	o	
112	70	p	
113	71	q	
114	72	r	
115	73	s	
116	74	t	
117	75	u	
118	76	v	
119	77	w	
120	78	x	
121	79	y	
122	7A	z	
123	7B	{	
124	7C		
125	7D	}	
126	7E	~	
127	7F	¤	DEL

UTF-8 —

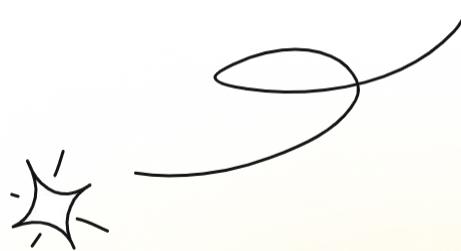
кодировка символов, которая позволяет представлять символы разных языков в компьютерной памяти.



UTF-8

	0	1	2	3	4	5	6	7	8	9	А	В	С	Д	Е	Ф
	D082	D083	E2809A	D193	E2809E	E280A6	E280A0	E280A1	E282AC	E280B0	D089	E280B9	D08A	D08C	D08B	D08F
80	Ђ	Ѓ	,	ѓ	„	...	†	‡	€	%	Љ	(Њ	Ќ	Ћ	Џ
	D192	E28098	E28099	E2809C	E2809D	E280A2	E28093	E28094		E284A2	D199	E280BA	D19A	D19C	D19B	D19F
90	Ѡ	Ѽ	‘	’	“	”	•	-	—	□	TM	Љ)	Њ	ќ	Ѡ
	C2A0	D08E	D19E	D088	C2A4	D089	C2A6	C2A7	D081	C2A9	D084	C2AB	C2AC	C2AD	C2AE	D087
A0	Ӯ	ӯ	J	#	Ӯ	I	Ӯ	Ӯ	Ӯ	Ӯ	©	Ӯ	«	Ӯ	-	®
	C2B0	C2B1	D086	D196	D291	C2B5	C2B6	C2B7	D191	E28496	D194	C2BB	D198	D085	D195	D197
B0	°	±	I	i	Ր	Ը	Ա	Ա	.	ë	№	€)	j	S	Ւ
	D090	D091	D092	D093	D094	D095	D096	D097	D098	D099	D09A	D09B	D09C	D09D	D09E	D09F
C0	Ա	Բ	Վ	Ղ	Ճ	Ե	Ժ	Յ	Ի	Ի	Կ	Լ	Մ	Հ	Օ	Ո
	D0A0	D0A1	D0A2	D0A3	D0A4	D0A5	D0A6	D0A7	D0A8	D0A9	D0AA	D0AB	D0AC	D0AD	D0AE	D0AF
D0	Պ	Ծ	Տ	Յ	Փ	Խ	Ծ	Չ	Շ	Щ	՚	Յ	՚	Յ	Յ	Յ
	D0B0	D0B1	D0B2	D0B3	D0B4	D0B5	D0B6	D0B7	D0B8	D0B9	D0BA	D0BB	D0BC	D0BD	D0BE	D0BF
E0	ա	բ	վ	ղ	ճ	ե	ժ	Յ	ի	ի	կ	լ	մ	հ	օ	ո
	D180	D181	D182	D183	D184	D185	D186	D187	D188	D189	D18A	D18B	D18C	D18D	D18E	D18F
F0	ր	ս	տ	յ	փ	խ	Ծ	Ч	Շ	Щ	՚	Յ	՚	Յ	Յ	Յ

Экспресс-опрос



С помощью каких операторов могут сравниваться строки?

Сравнение строк

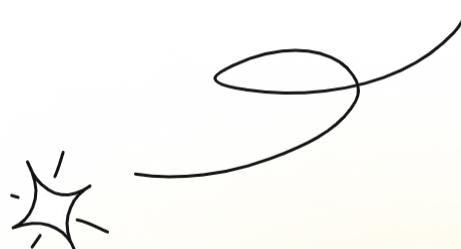
В Python строки могут быть сравнены с помощью операторов сравнения:

- ==
- !=
- <
- >
- <=
- >=

Стать IT-шником может каждый



Экспресс-опрос

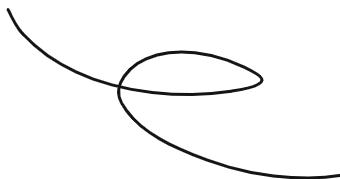


Что такое лексикографическое сравнение?



Лексикографическое сравнение —

строки сравниваются посимвольно, сравнивая их числовые значения кодов символов в таблице символов.



Сравнение строк

```
str1 = "apple"
str2 = "banana"
result = str1 < str2 # Сравнение строк по лексикографическому порядку
```

Стать IT-шником может каждый



Экспресс-опрос



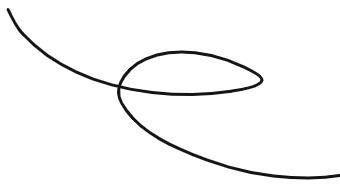
Для чего нужен метод `in` ?





Метод `in` —

метод в Python, который позволяет проверить, содержится ли одна строка в другой.

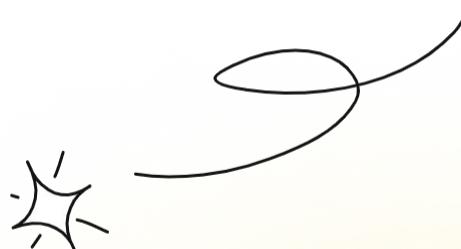


Метод `in`

Метод `in` возвращает значение **True**, если подстрока найдена, и **False**, если подстрока отсутствует.

```
text = "Hello, world!"  
is_found = "world" in text # Проверка наличия подстроки
```

Экспресс-опрос



Как осуществляется доступ к отдельным символам строки?

Доступ к элементу по индексу

Доступ к отдельным символам строки осуществляется по индексу.

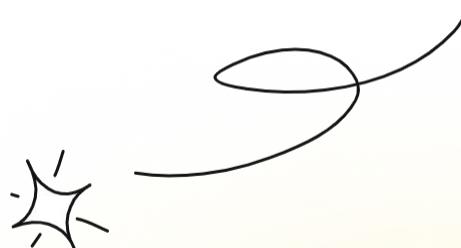
- Положительные индексы начинаются с 0 и указывают на позицию символа в строке.
- Отрицательные индексы считаются с конца строки, где -1 соответствует последнему символу.

```
text = "Hello"
first_char = text[0] # Получение первого символа
last_char = text[-1] # Получение последнего символа
```

Стать IT-шником может каждый



Экспресс-опрос



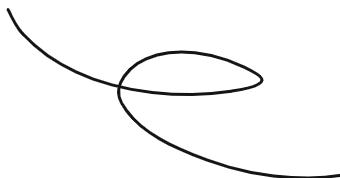
Что значит неизменяемость строк?





Неизменяемость строк —

строки в Python являются **неизменяемыми объектами**, что означает, что после создания их содержимое не может быть изменено.



Неизменяемость строк

Демонстрация неизменяемости может быть выполнена с помощью функции `id()`, которая возвращает идентификатор объекта.

```
text = "Hello"  
text_id = id(text) # Получение идентификатора объекта
```

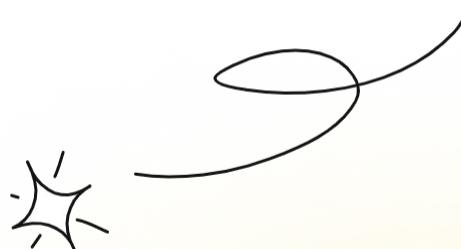
Неизменяемость строк

```
a={"1","2"}  
print(id(a))  
print(a)  
a.update({"3"})  
print(id(a))  
print(a)  
s="Hello"  
print(id(s))  
print(s)  
s=s.lower()  
print(id(s))  
print(s)
```

Стать IT-шником может каждый



Экспресс-опрос

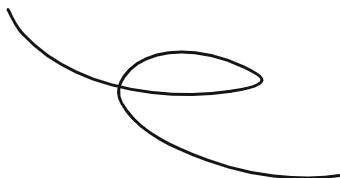


Что такое взятие среза от строки?



Взятие среза от строки —

[операция](#), которая используется для извлечения подстроки из строки.



Взятие среза от строки

С помощью срезов можно выбрать определенный диапазон символов.

Варианты срезов включают использование одного, двух или трех аргументов:

- Один аргумент указывает индекс символа, с которого начинается срез.
- Два аргумента указывают начальный и конечный индексы среза.
- Три аргумента позволяют указать шаг, с которым будут выбраны символы.

Взятие среза от строки

```
text = "Hello, world!"  
substring1 = text[0:5] # Взятие среза с 1-го по 5-й символы  
substring2 = text[7:] # Взятие среза с 8-го символа до конца строки  
substring3 = text[::-2] # Взятие среза с шагом 2
```

Копирование, разворот строки

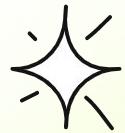
- С помощью срезов можно создавать копии строк.
- Можно использовать срезы с отрицательным шагом для разворота строки.

```
text = "Hello"
copy = text[:] # Создание копии строки
reversed_text = text[::-1] # Разворот
```

Стать IT-шником может каждый



ВОПРОСЫ



Стать IT-шником может каждый



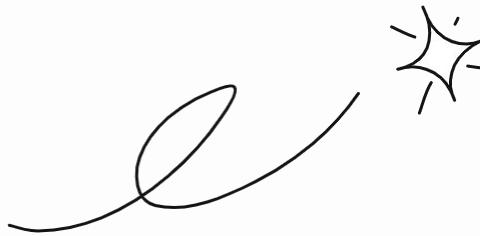
ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ



Подумайте:

- Что покажет приведенный ниже фрагмент кода?

```
mystr = 'да'  
mystr = mystr + 'нет'  
mystr = mystr + 'да'  
print(mystr)
```

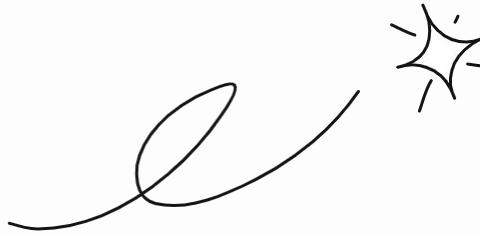




Подумайте:

2. Что покажет приведенный ниже фрагмент кода?

```
str1 = '1'  
str2 = str1 + '2' + str1  
str3 = str2 + '3' + str2  
str4 = str3 + '4' + str3  
print(str4)
```

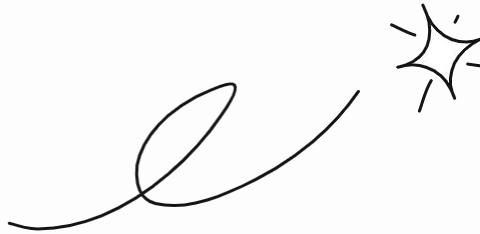




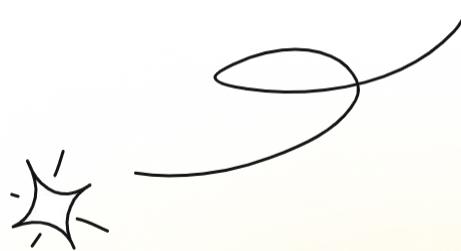
Подумайте:

3. Что покажет приведенный ниже фрагмент кода?

```
mystr = '123' * 3 + '456' * 2 + '789' * 1
print(mystr)
```



Экспресс-опрос



Что делают методы `find()` и `index()`?



Поиск подстроки в строке

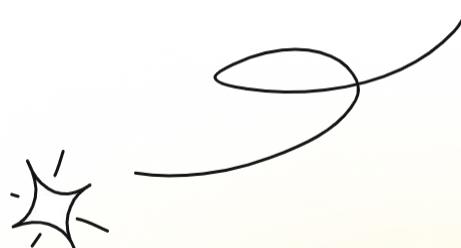
Для поиска подстроки в строке можно использовать метод **find()**.

- **find()** возвращает индекс первого вхождения подстроки или -1, если подстрока не найдена.
- **index()**, работает аналогично методу `find()`, но выбрасывает исключение, если подстрока не найдена.

Поиск подстроки в строке

```
text = "Hello, world!"  
index = text.find("world") # Поиск подстроки "world"
```

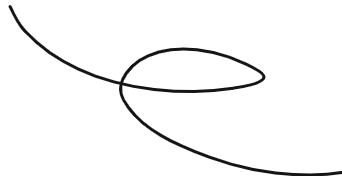
Экспресс-опрос



Какие методы позволяют проверить определенное условие относительно символов строки?

isalpha(), islower(), isupper() —

методы, которые позволяют проверить определенное условие относительно символов строки.



Метод `isalpha()`

Метод `isalpha()` возвращает значение **True**, если все символы в строке являются буквами, и **False** в противном случае.

```
text = "Hello"  
is_alpha = text.isalpha() # Проверка, содержит ли строка только буквы
```

Метод `islower()`

Метод `islower()` возвращает значение **True**, если все буквы в строке являются строчными, и **False** в противном случае.

```
text = "hello"
is_lower = text.islower() # Проверка, содержит ли строка только строчные
буквы
```

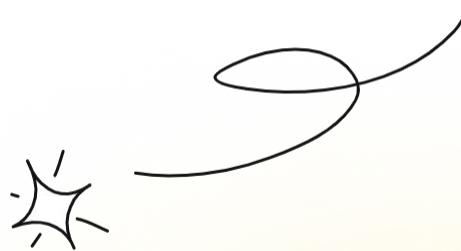
Стать IT-шником может каждый



Метод isupper()

Метод `isupper()` проверяет, состоит ли строка только из прописных букв.

Экспресс-опрос



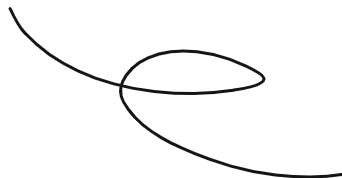
Что делают методы startswith(), endswith() ?





startswith(), endswith() —

методы, которые позволяют проверить определенное условие относительно строки.



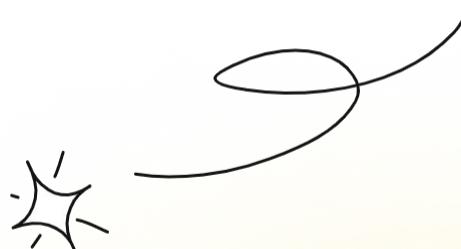
Метод `startswith()`, `endswith()`

Метод **`startswith()`** возвращает значение `True`, если строка начинается с определенной подстроки, и `False` в противном случае.

Метод **`endswith()`** проверяет, заканчивается ли строка определенной подстрокой.

```
text = "Hello, world!"  
starts_with_hello = text.startswith("Hello") # Проверка, начинается ли строка с  
"Hello"
```

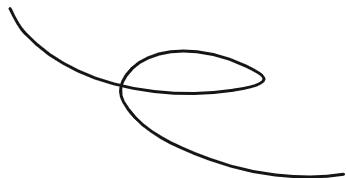
Экспресс-опрос



Какие методы возвращают новую строку, преобразованную к нижнему или верхнему регистру соответственно?

lower() и upper() —

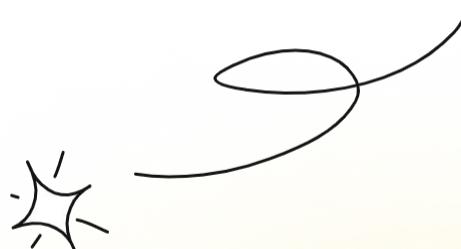
методы, которые возвращают новую строку, преобразованную к нижнему или верхнему регистру соответственно.



Методы lower() и upper()

```
text = "Hello, World!"  
lower_text = text.lower() # Преобразование строки к нижнему регистру
```

Экспресс-опрос

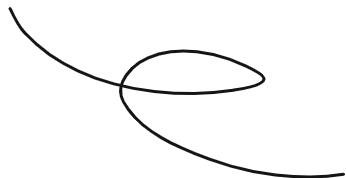


Для чего используется метод replace()?



replace() —

метод, который создает новую строку, заменяя все вхождения указанной подстроки на другую подстроку.



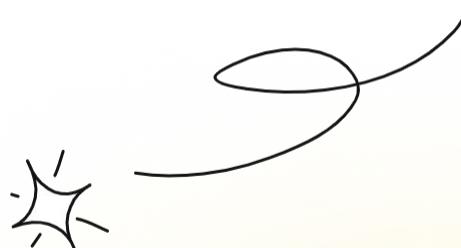
Метод replace()

```
text = "Hello, World!"  
new_text = text.replace("Hello", "Hi") # Замена подстроки "Hello" на "Hi"
```

Стать IT-шником может каждый



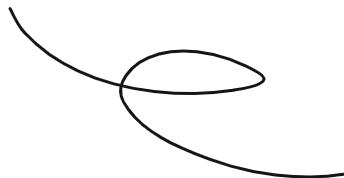
Экспресс-опрос



Какие методы разделяют/соединяют строки?

split() и join() —

[методы](#), которые разделяют/соединяют строки.



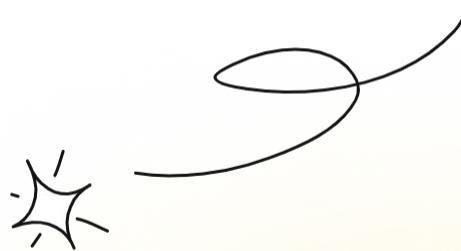
Методы `split()` и `join()`

Метод **split()** разделяет строку на подстроки по указанному разделителю и возвращает список подстрок.

Метод **join()** объединяет список строк в одну строку, используя указанный разделитель.

```
text = "Hello, world!"  
words = text.split(", ") # Разделение строки на подстроки  
joined_text = ", ".join(words) # Объединение списка подстрок в строку
```

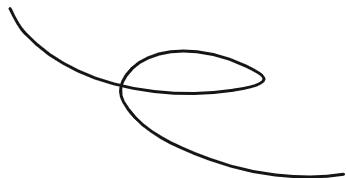
Экспресс-опрос



Для чего используются ljust(), rjust(), center()?

ljust(), rjust(), center() —

[методы](#), которые выравнивают строки.



Выравнивание строк

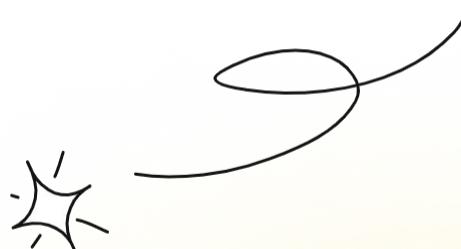
Метод **ljust()** выравнивает строку по левому краю, добавляя пробелы в конец строки, чтобы достичь заданной ширины.

Метод **rjust()** выравнивает строку по правому краю, добавляя пробелы в начало строки.

Метод **center()** выравнивает строку по центру, добавляя пробелы с обеих сторон.

```
text = "Hello"  
left_aligned = text.ljust(10) # Выравнивание по левому краю
```

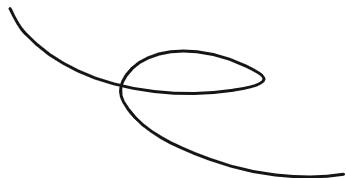
Экспресс-опрос



Для чего используются strip(), lstrip(), rstrip()?

strip(), lstrip(), rstrip() —

[методы](#), которые удаляют ненужные символы.



Удаление ненужных символов

Метод **strip()** удаляет пробелы и другие символы из начала и конца строки.

Метод **lstrip()** удаляет символы только из начала строки.

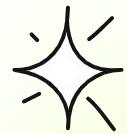
Метод **rstrip()** удаляет символы только из конца строки.

```
text = "Hello, world! "
stripped_text = text.strip() # Удаление пробелов из начала и конца строки
```

Стать IT-шником может каждый



ВОПРОСЫ



Стать IT-шником может каждый



ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ



Какие значения может принимать строковая переменная `s`, чтобы в результате выполнения кода было выведено слово «YES»? Выберите все подходящие ответы:



```
if s in 'abc123abc':  
    print('YES')  
else:  
    print('NO')
```

- a. `s = 'bca'`
- b. `s = '1'`
- c. `s = '123abc'`
- d. `s = 'bc2'`
- e. `s = 'a'`
- f. `s = 'abcabc'`
- g. `s = '23'`
- h. `s = '321'`
- i. `s = '3ab'`



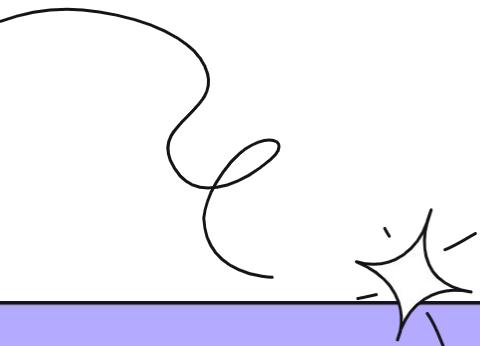
Стать IT-шником может каждый



РАЗБОР ДОМАШНЕГО ЗАДАНИЯ



Домашнее задание 6

- 
- 
1. Напишите программу, которая генерирует случайное число от 1 до 100, а затем предлагает пользователю угадать это число. Если пользователь угадывает число, программа выводит сообщение о победе. Если пользователь не угадывает число, программа сообщает, больше или меньше загаданное число и предлагает попробовать снова. Используйте цикл с инструкцией `break`, чтобы остановить выполнение цикла, когда число угадано.

Пример вывода:

Угадайте число от 1 до 100: 50

Загаданное число меньше.

Попробуйте снова: 75

Загаданное число больше.

Попробуйте снова: 63

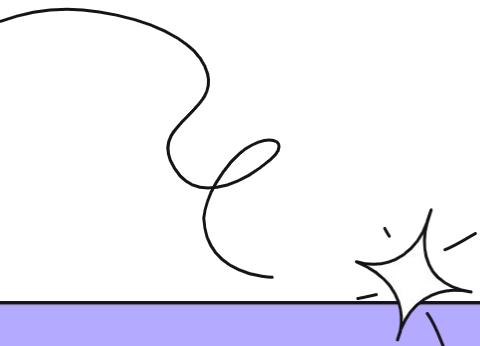
Поздравляю! Вы угадали число 63!

Домашнее задание 6

```
from random import randint

x=randint(1,100)
t=int(input("Угадайте число от 1 до 100: "))
while t!=x:
    if t<x:
        print("Загаданное число больше")
    else:
        print("Загаданное число меньше")
    t=int(input("Попробуйте снова: "))
else:
    print("Поздравляю, вы угадали. Это число", x)
```

Домашнее задание 6

- 
2. Напишите программу, которая запрашивает у пользователя число N и выводит первые N чисел Фибоначчи. Числа Фибоначчи - это последовательность чисел, где каждое следующее число является суммой двух предыдущих чисел (начиная с 0 и 1). Используйте цикл while для решения задачи. Выведите числа через запятую с помощью команды print.



Пример вывода:

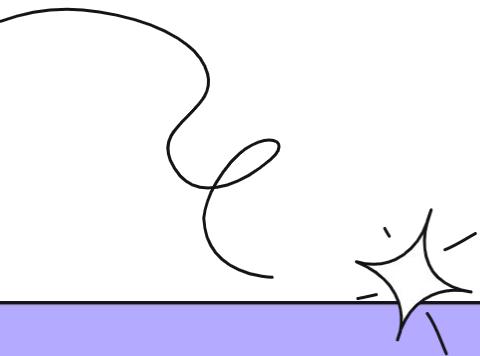
Введите число N: 7

Первые 7 чисел Фибоначчи: 0, 1, 1, 2, 3, 5, 8

Решение 2

```
N=int(input("Введите N:"))
a=0
print("Первые", N, "чисел Фибоначчи: ", end="")
b=1
count=2
while count<N:
    print(a, end=", ")
    c=a+b
    a=b
    b=c
    count+=1
else:
    print(a)
```

Домашнее задание 6

- 
3. Напишите программу, которая запрашивает у пользователя целое положительное число и проверяет, является ли оно простым. Простое число - это число, которое делится только на 1 и на само себя без остатка. Используйте цикл while и проверку деления числа на все числа от 2 до N-1 для решения задачи. Выведите соответствующее сообщение на экран с помощью команды print.



Пример вывода:

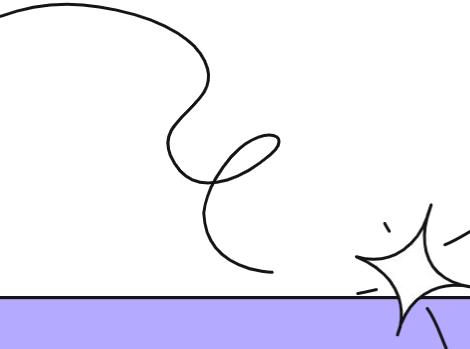
Введите целое положительное число: 17

Число 17 является простым.

Домашнее задание 6

```
N=int(input("Введите N:"))
i=2
while i<N:
    if N%i==0:
        print("Это число не простое")
        break
    else:
        i+=1
else:
    print("Это число простое")
```

Домашнее задание 7



1. Напишите программу, которая запрашивает у пользователя натуральное десятичное число и выводит его двоичное представление. Реализуйте алгоритм перевода числа в двоичную систему счисления, используя основные концепции представления чисел (подсказка: через деление с остатком). Выведите полученное представление числа на экран.



Пример вывода:

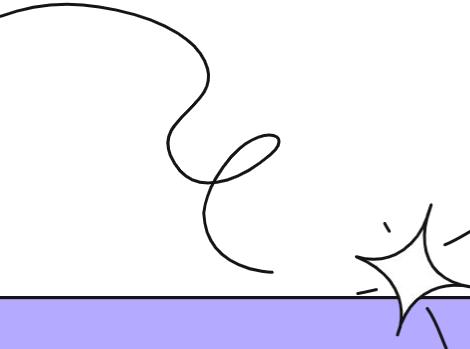
Введите натуральное десятичное число: 123

Двоичное представление числа: 1111011

Домашнее задание 7

```
N=int(input("Введите натуральное число: "))
res=""
while N>0:
    res=str(N%2)+res
    N//=2
print("Двоичное представление:",res)
```

Домашнее задание 7



2. Напишите программу, принимающую число с плавающей точкой и округляющую его до целого числа в соответствии с правилами школьной математики. Использовать модуль `math` и методы из него нельзя. Учесть, что программа должна корректно работать с отрицательными числами.

Пример вывода:

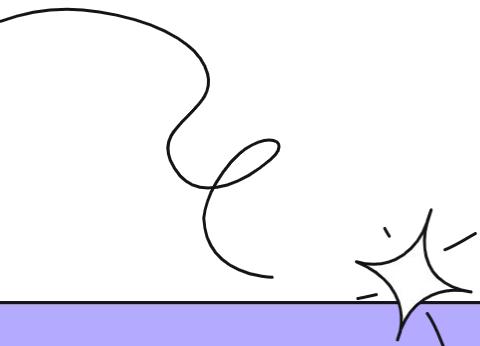
Введите вещественное число: -3.14

Округленное значение: -3

Домашнее задание 7

```
F=float(input("Введите действительное число: "))
if F>0:
    FF=int(F+0.5)
else:
    FF=int(F-0.5)
print("Результат округления:", FF)
```

Домашнее задание 8

- 
1. Напишите программу, которая запрашивает у пользователя целое число и определяет, является ли оно палиндромом. Число является палиндромом, если оно читается одинаково слева направо и справа налево. Выведите соответствующее сообщение на экран с помощью команды `print`. Используйте математические операции. Работу со строками использовать нельзя.



Пример вывода:

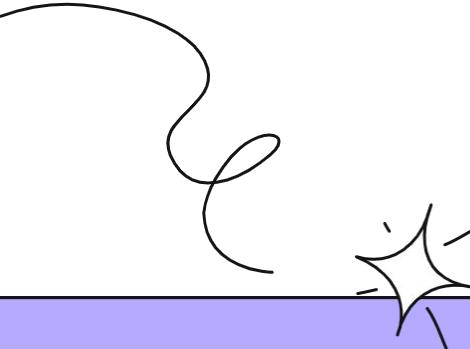
Введите целое число: 12321

Число является палиндромом.

Домашнее задание 8

```
N=int(input("Введите натуральное число:"))
saveN=N
reverseN=0
while N>0:
    reverseN*=10
    reverseN+=N%10
    N/=10
if saveN==reverseN:
    print("Палиндром")
else:
    print("Не палиндром")
```

Домашнее задание 8



2. Напишите программу, которая запрашивает у пользователя целое число и проверяет, является ли оно числом Армстронга. Число Армстронга - это число, которое равно сумме своих цифр, введенных в степень, равную количеству цифр в числе.

Выведите соответствующее сообщение на экран с помощью команды print.

Пример вывода:

Введите целое число: 153

Число является числом Армстронга.

Домашнее задание 8

```
N=int(input("Введите натуральное число:"))
saveN=N
digits=0
while N>0:
    N/=10
    digits+=1
arm=0
N=saveN
while N>0:
    arm+=(N%10)**digits
    N/=10
if saveN==arm:
    print("Число Армстронга")
else:
    print("Не число Армстронга")
```

Домашнее задание 9

1. Напишите программу, которая запрашивает у пользователя строку и определяет, является ли она панграммой.



Панграмма - это фраза, содержащая все буквы алфавита.

Программа должна игнорировать регистр букв и пробелы при проверке панграммы.

Выведите соответствующее сообщение на экран с помощью команды print.

Решить задачу для латиницы.

Пример вывода:

Введите строку: The quick brown fox jumps over the lazy dog

Строка является панграммой.

Решение 1

```
alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZAA"  
s=input("Введите строку латиницей:")  
i=0  
while i<len(alphabet):  
    if alphabet[i] in s.upper():  
        i+=1  
    else:  
        print("Это не панграмма")  
        break  
else:  
    print("Это панграмма")
```

Домашнее задание 9

2. Напишите программу, которая запрашивает у пользователя строку и выводит на экран количество гласных и согласных букв в ней.



Используйте функцию `len()` для подсчета количества букв.

Выведите результат на экран с помощью команды `print`.

Решить задачу для латиницы.

Пример вывода:

Введите строку: Hello World

Количество гласных букв: 3

Количество согласных букв: 7

Решение 2

```
aei="AEIOUY"  
bcd="BCDFGJHKLMNPQRSTVWXZ"  
s=input("Введите строку латиницей:").upper()  
i=0  
s1=""  
s2=""  
while i<len(s):  
    if aei.find(s[i])>=0:  
        s1+=s[i]  
    if bcd.find(s[i])>=0:  
        s2+=s[i]  
    i+=1  
print("Гласных букв",len(s1))  
print("Согласных букв",len(s2))
```



Домашнее задание 10

1. Напишите программу, которая запрашивает у пользователя строку и преобразует ее, удаляя все гласные буквы из строки. Используйте метод `replace()` для замены гласных букв на пустую строку. Выведите преобразованную строку на экран с помощью команды `print`.

Пример вывода:

Введите строку: Hello, world!

Результат: Hll, wrld!



Решение 1

```
aei="AEIOUY"  
s=input("Введите строку латиницей:")  
i=0  
while i<len(aei):  
    s=s.replace(aei[i],"")  
    s=s.replace(aei[i].lower(),"")  
    i+=1  
print("Результат: ",s)
```



Домашнее задание 10

2. Напишите программу, которая запрашивает у пользователя строку и определяет, содержит ли она только уникальные символы. Если все символы в строке уникальны, выведите соответствующее сообщение на экран. В противном случае выведите сообщение о том, какие символы повторяются. Не используйте множества и подобные структуры данных, которые мы пока не изучали, для проверки уникальности символов.



Пример вывода:

Введите строку: Python

Все символы в строке уникальны.

Введите строку: Hello

Символы 'l' и 'o' повторяются.



Решение 2



```
s=input("Введите строку:").lower()
i=0
while i<len(s):
    if s[i+1:].find(s[i])==-1:
        i+=1
    else:
        print("Символы не уникальны")
        break
else:
    print("Символы уникальны")
```



Домашнее задание 10

3. Напишите программу, которая запрашивает у пользователя строку и выравнивает ее по центру с заданной шириной. Если строка не может быть выровнена по центру из-за нечетной ширины, она должна быть выровнена смещением вправо. Используйте методы center() и rjust() для выравнивания строки.



Пример вывода:

Введите строку: Python

Введите ширину: 10

Результат:

Python



Решение 3

```
s=input("Введите строку: ")  
  
d=int(input("Введите ширину: "))  
  
while d<=len(s):  
  
    d=int(input("Ширина должна быть больше длины строки. Введите ширину:"))  
  
    if len(s)%2==d%2:  
  
        s=s.center(d)  
  
    else:  
  
        s=s.rjust(d)  
  
print(s)
```



Стать IT-шником может каждый



ОСТАВШИЕСЯ ВОПРОСЫ



Стать IT-шником может каждый



ЗАКЛЮЧЕНИЕ

