PRODUCT
STAR.RU

Строки, условия, циклы

Содержание урока

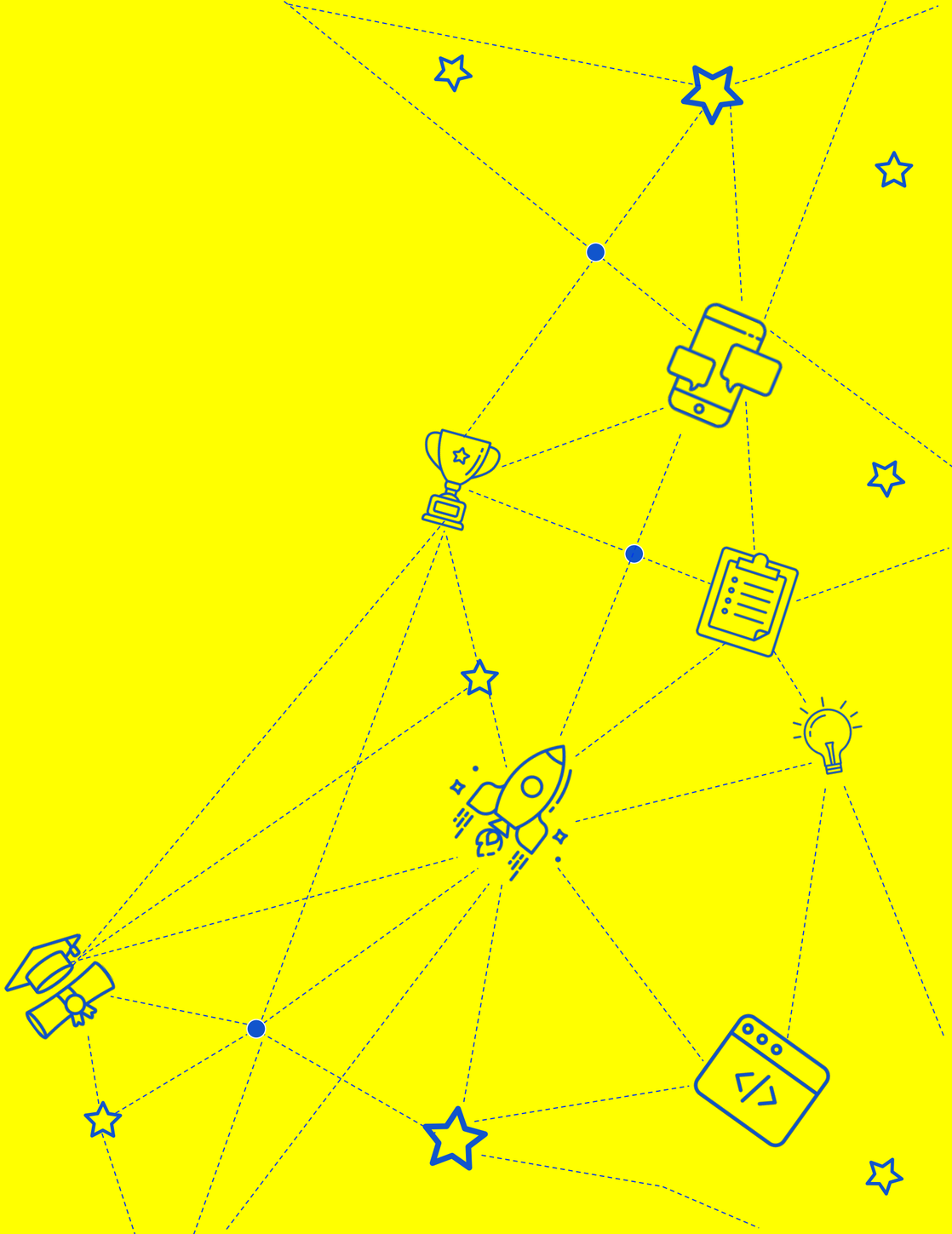
- Строки
- Строки*
- Bytes
- Условия
- Циклы

НИКИТА СОБОЛЕВ

CTO, wemake.services

- Занимаюсь OpenSource полный рабочий день
- Топ-60 по коммитам в CPython
- Первый GitHub Star в России

Строки



'Hello, world!'

"Привет, мир!"

'Привет, 🌍'

'Hello, world!'

"Привет, мир!"

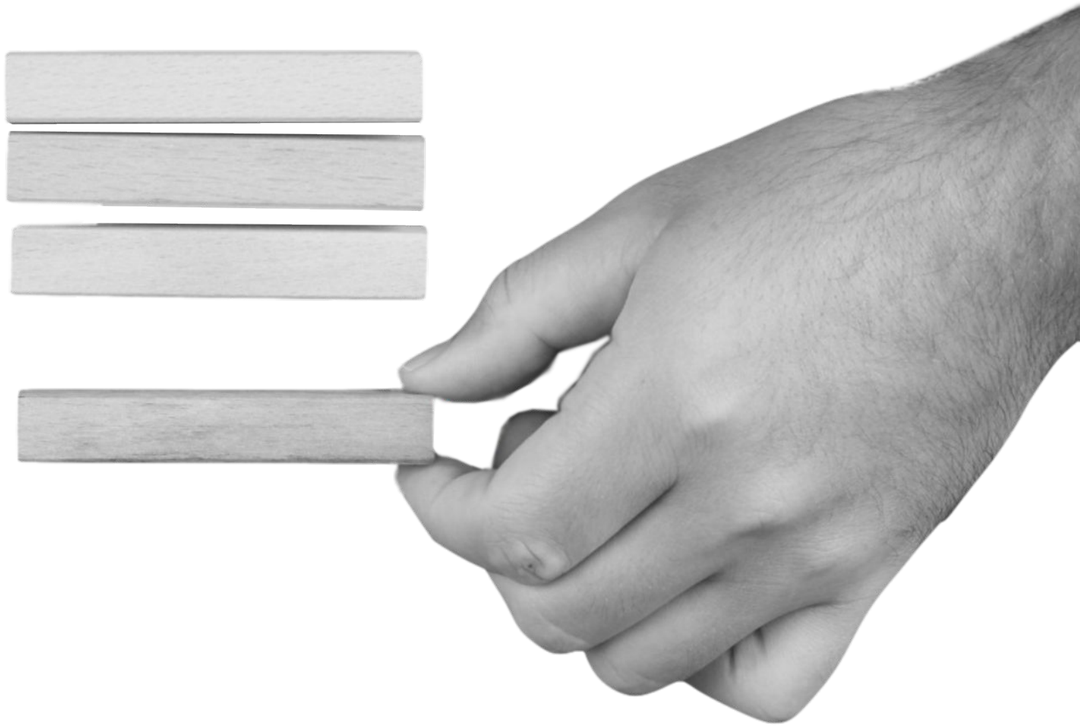
'Привет, 🌍'

'Hello, world!'

"Привет, мир!"

'Привет, 🌍'

Что можно делать со строками?



Что можно делать со строками?

Складывать или конкатенировать

```
'Hello, ' + 'world!'
```

Что можно делать со строками?

Сравнивать

```
'Hello, ' + 'world!' == 'Hello, world!'
# True
```

Что можно делать со строками?

Проверять вхождение

```
'a' in 'abc'
# True


'yz' not in 'abc'
# True
```

Что можно делать со строками?

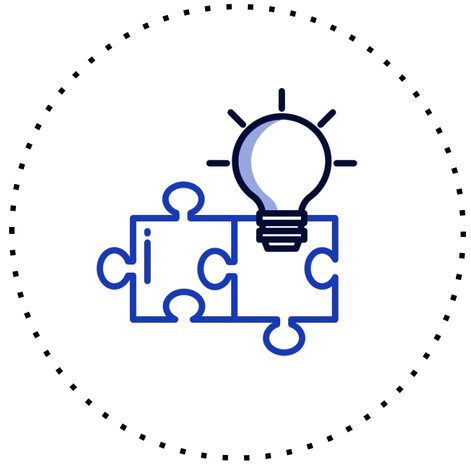
Узнавать длину

```
len('7 chars')
# 7
```

Что можно делать со строками?

Брать отдельные буквы (или их наборы)

```
'abc'[0]
# 'a'



'abcde'[1:4]
# 'bcd'
```

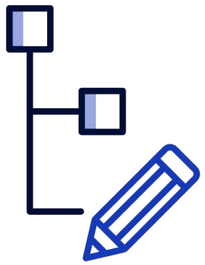


Что можно делать со строками?

Форматировать

```
'Hello, {0}. You are learning {1}'.format('Nikita', 'Python')

# Hello, Nikita. You are learning Python
```

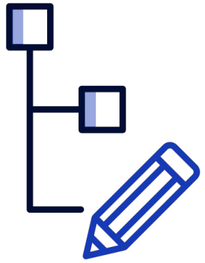
Что можно делать со строками?



Создать новую строку в новом регистре

```
'Hello, world!'.lower()
# hello, world!


'Hello, world!'.upper()
# HELLO, WORLD!


'Hello, world!'.title()
# Hello, World!
```

Что можно делать со строками?



Создать новую строку в новом регистре

```
'Hello, world!'.lower()
# hello, world!
```
← **все буквы маленькие**

```
'Hello, world!'.upper()
# HELLO, WORLD!
```
← **все буквы большие**

```
'Hello, world!'.title()
# Hello, World!
```
← **слова начинаются с заглавной буквы**

Что можно делать со строками?

Превращать другие объекты в строки и обратно

```
str(1 + 1)
# '2'



int('15')
# 15
```

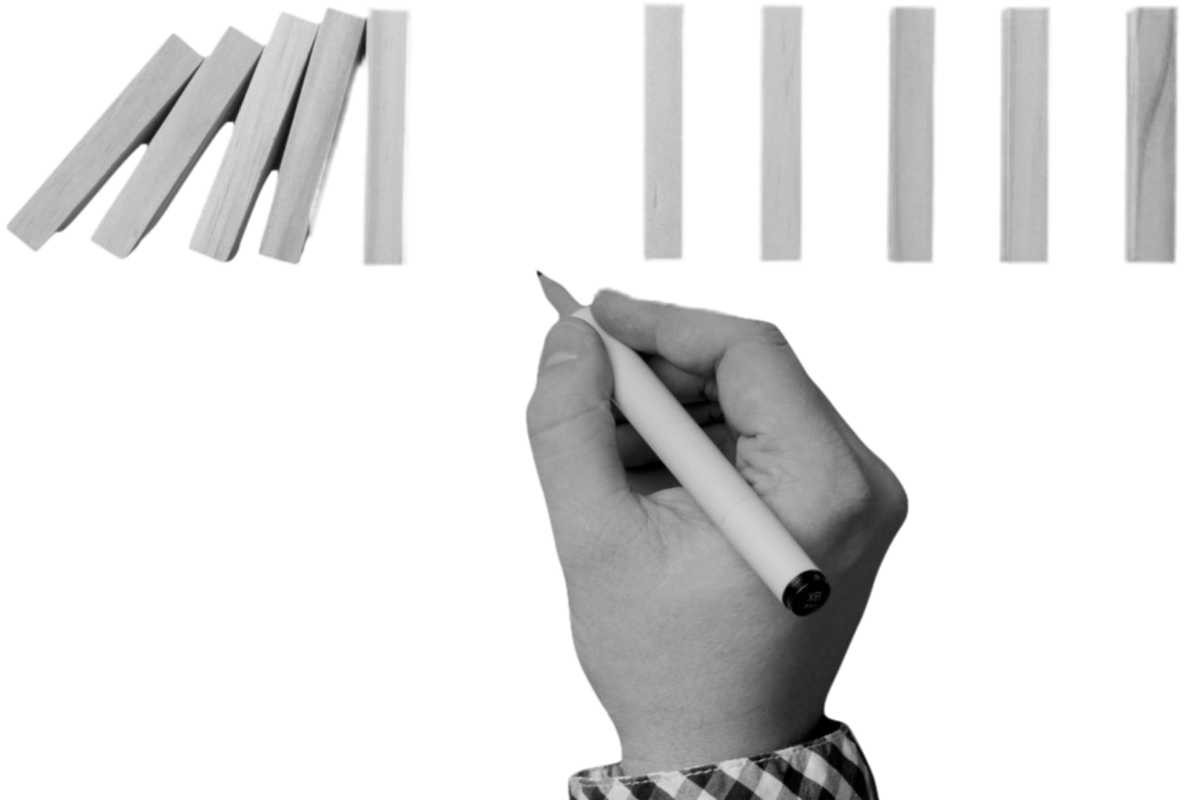
В чем разница между 1 и ‘1’?

1 – число, int

‘1’ – строка, str

```
1 + 1
# 2


'1' + '1'
# ‘11’

1 + '1'
# TypeError: unsupported operand type(s) for +:
'int' and 'str'
```

Чего нельзя делать со строками?

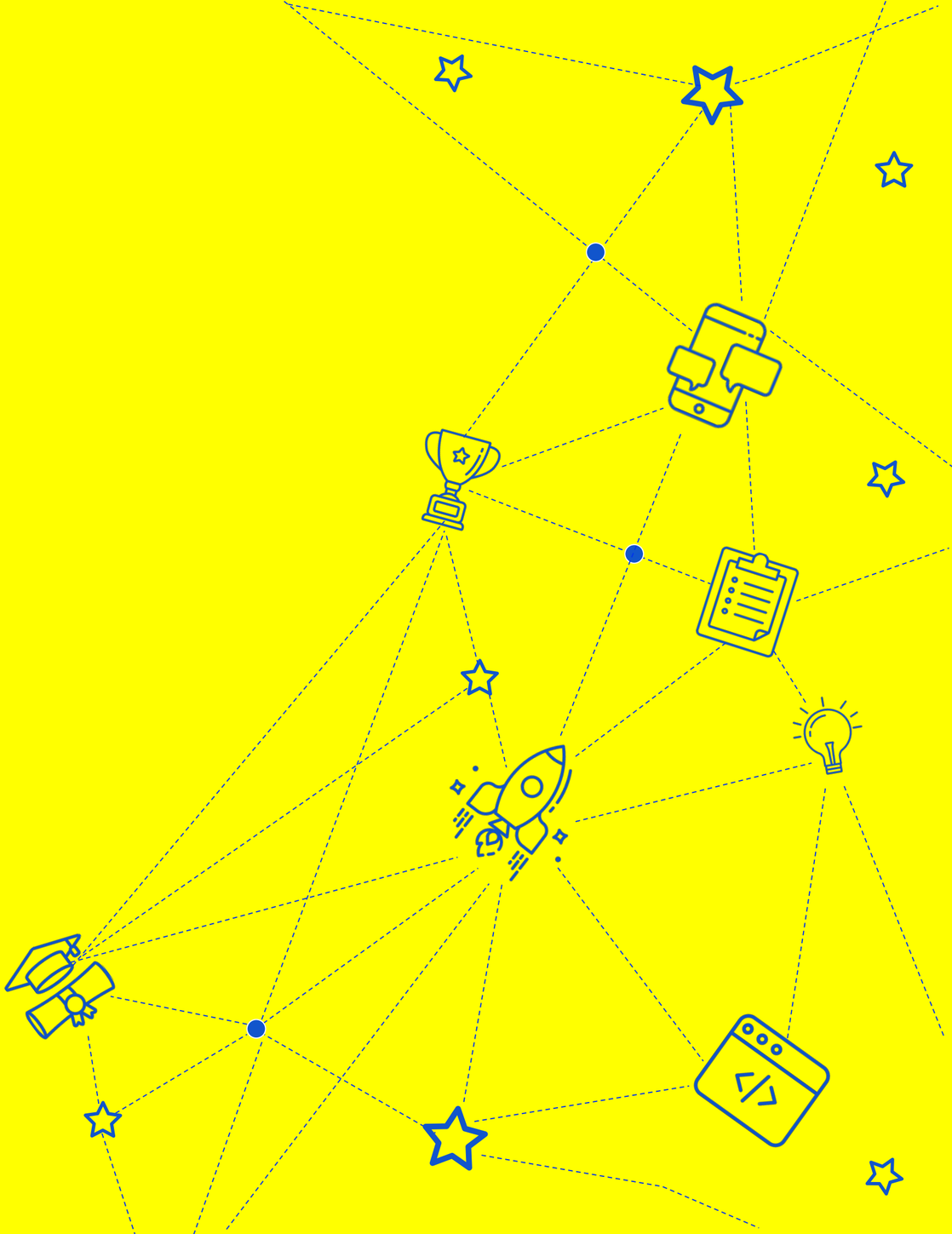


Чего нельзя делать со строками?

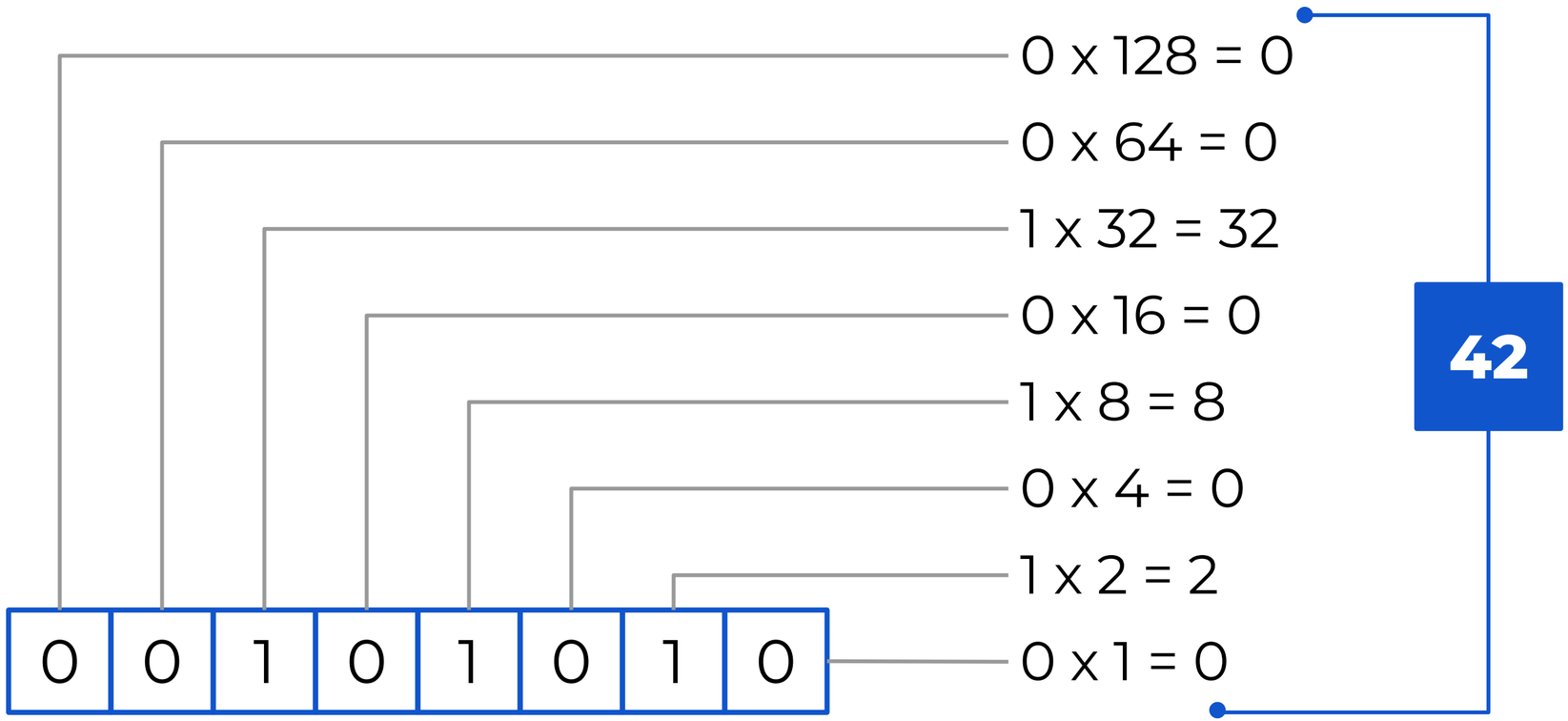
Забывать их документацию

https://docs.python.org/3/library/stdtypes.html#str

Строки*

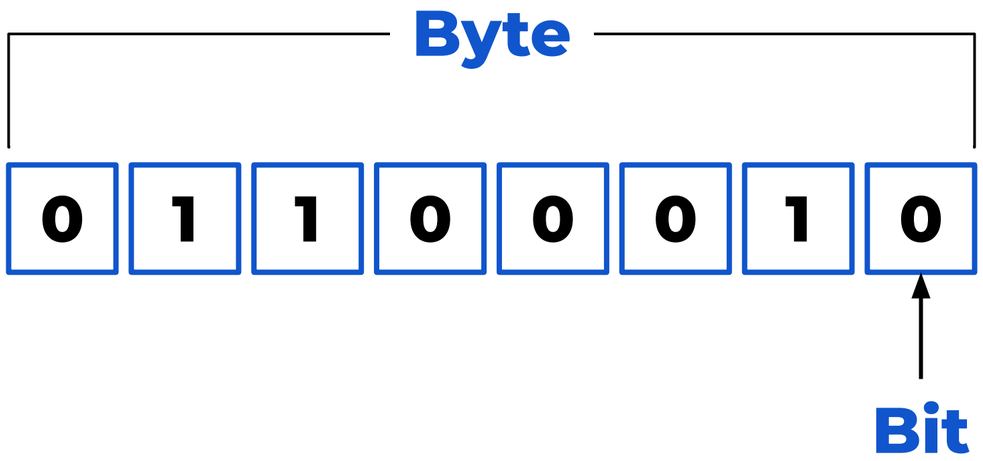


Самое важное в любой структуре – знать, как она хранится в памяти

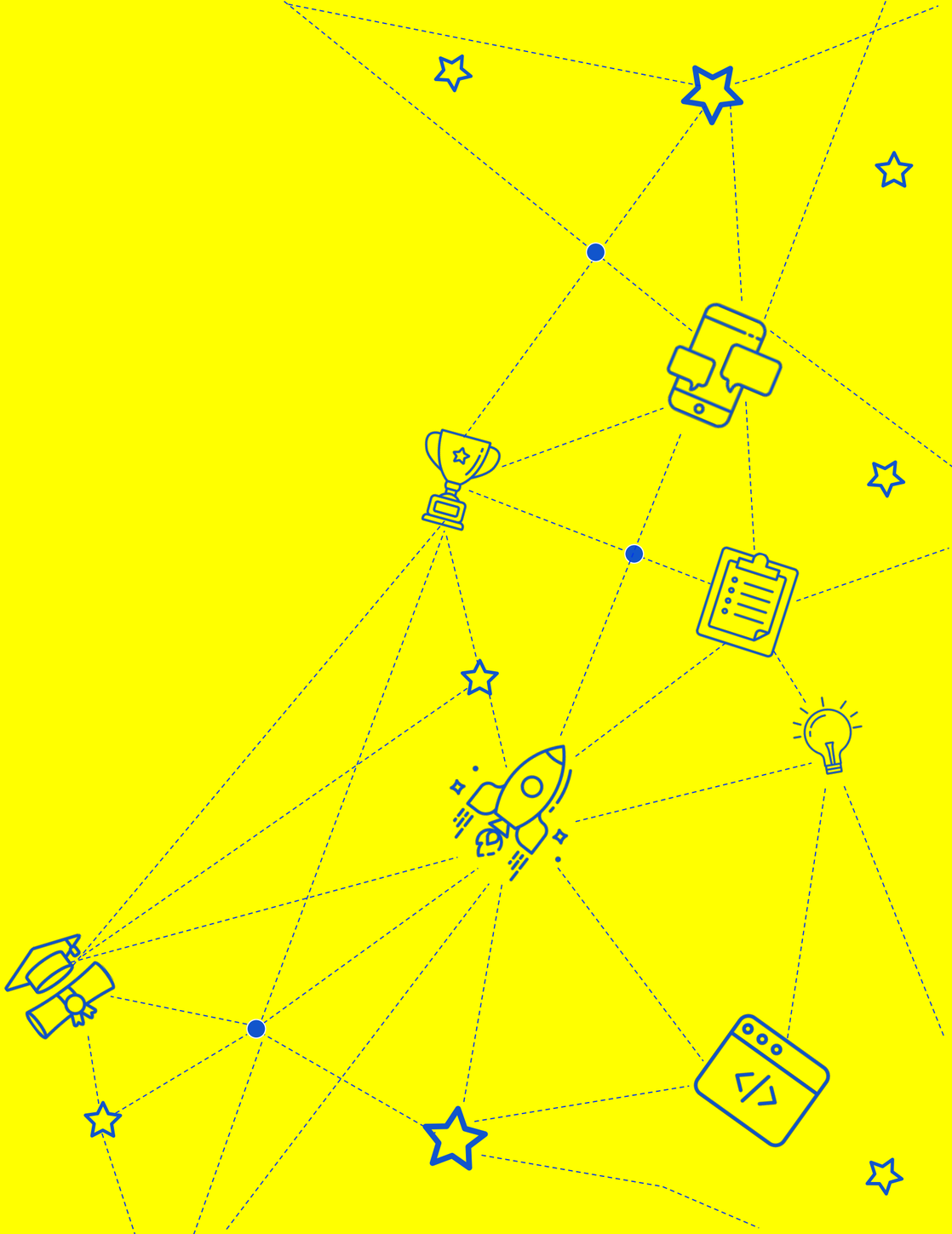


Внутри памяти компьютера все хранится как числа

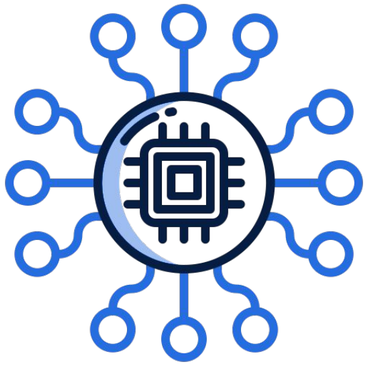
И строки тоже!



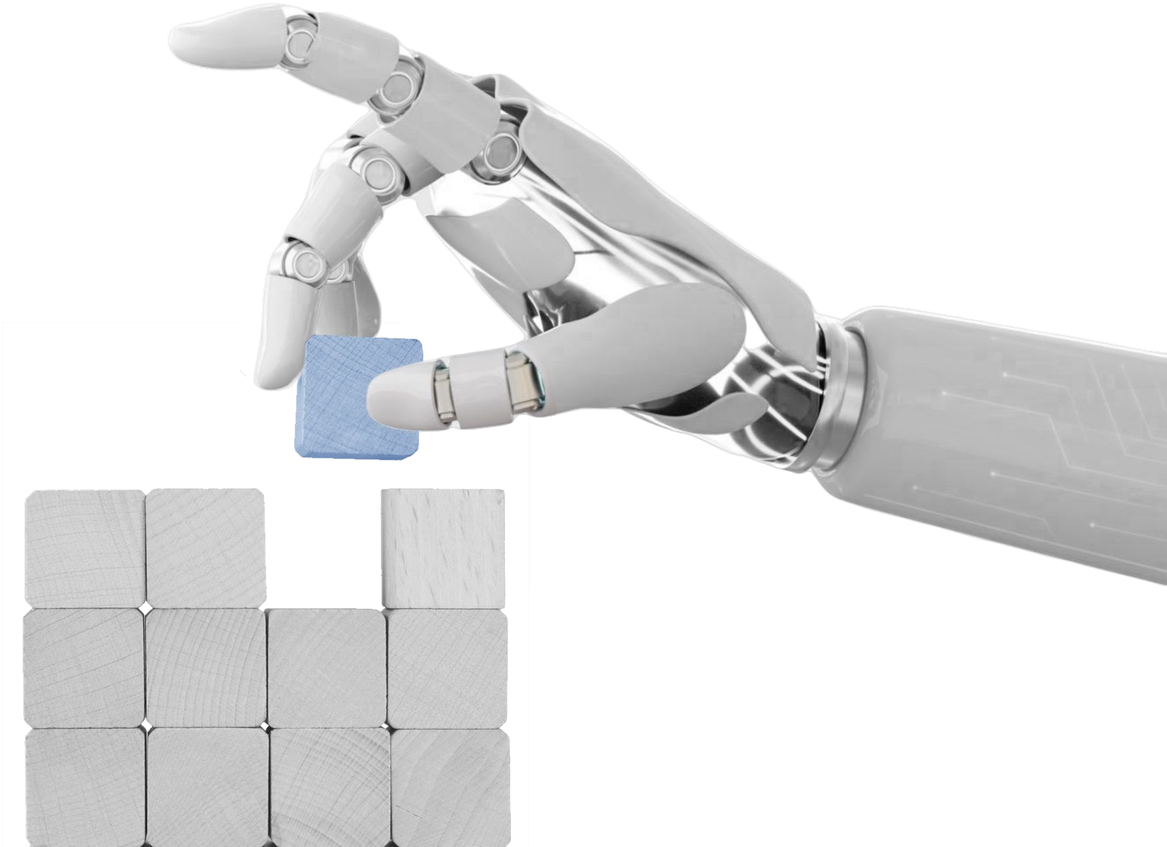
bytes



`str`
не мог бы существовать
без `bytes`!

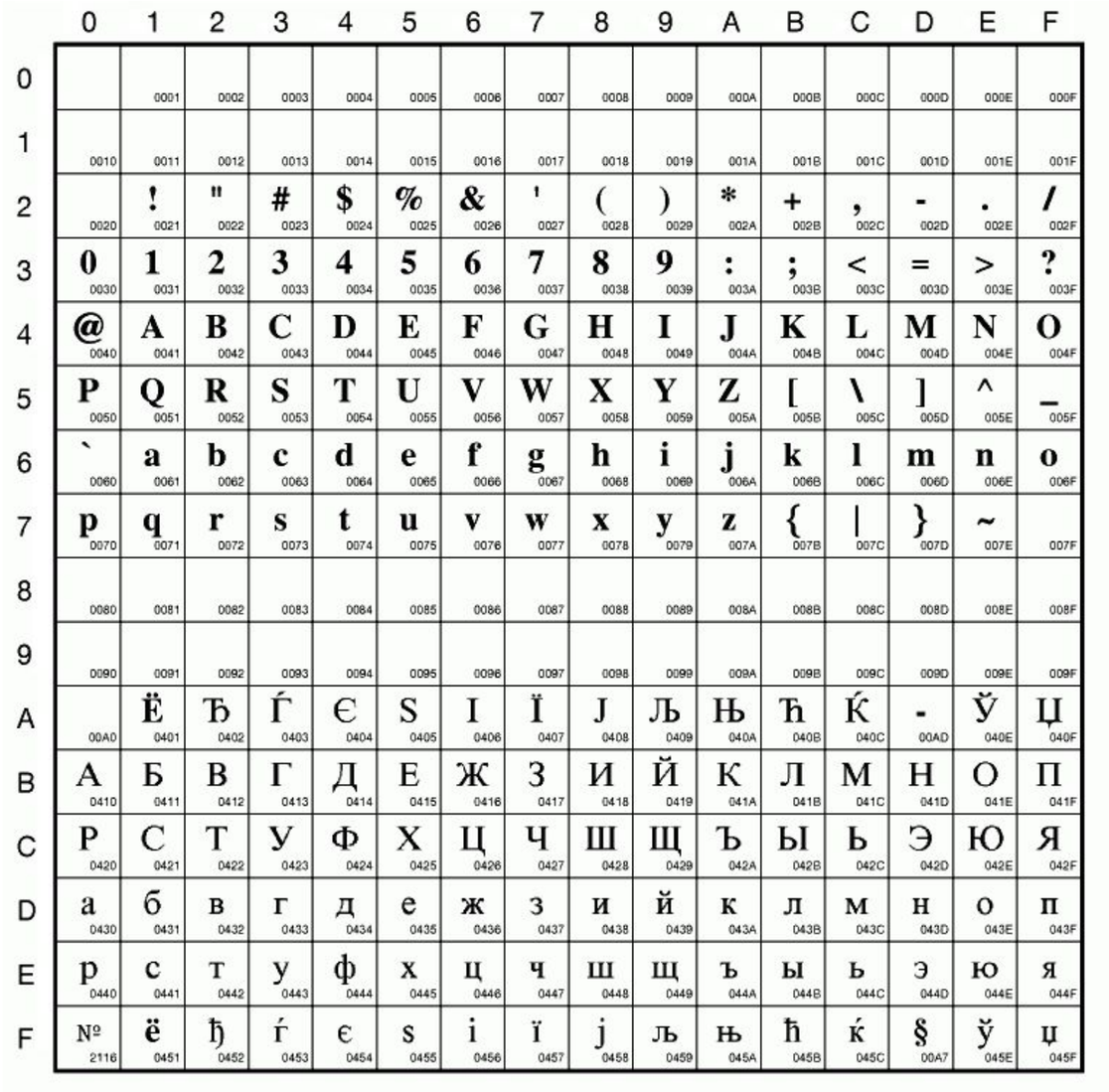


bytes = набор чисел, которые представляют строки в различных кодировках



Unicode = таблица символов с их цифровыми кодами

ISO-8859-5



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	008A	008B	008C	008D	008E	008F
9	0090	0091	0092	0093	0094	0095	0096	0097	0098	0099	009A	009B	009C	009D	009E	009F
A	00A0	Ё 0401	Ђ 0402	Ѓ 0403	Є 0404	S 0405	I 0406	Ї 0407	J 0408	Љ 0409	Њ 040A	Ћ 040B	Ќ 040C	- 00AD	Ў 040E	Џ 040F
B	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
C	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
D	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
E	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F
F	№ 2116	ё 0451	ђ 0452	ѓ 0453	є 0454	s 0455	і 0456	ї 0457	j 0458	љ 0459	њ 045A	ћ 045B	ќ 045C	§ 00A7	ў 045E	џ 045F

```
ord('!')
# 33

ord('я')
# 1103
```

Числовое представление

```
ord('!')
# 33

ord('я')
# 1103
```

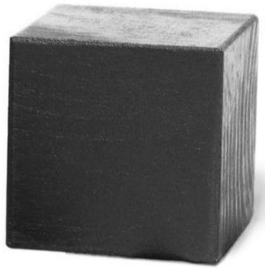
```
int.from_bytes(b'!')
# 33

int.from_bytes('я'.encode('utf8'))
# 1103
```

utf8 –

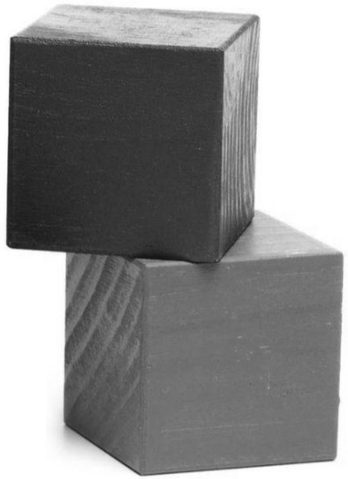
самая популярная кодировка

Но есть и другие:

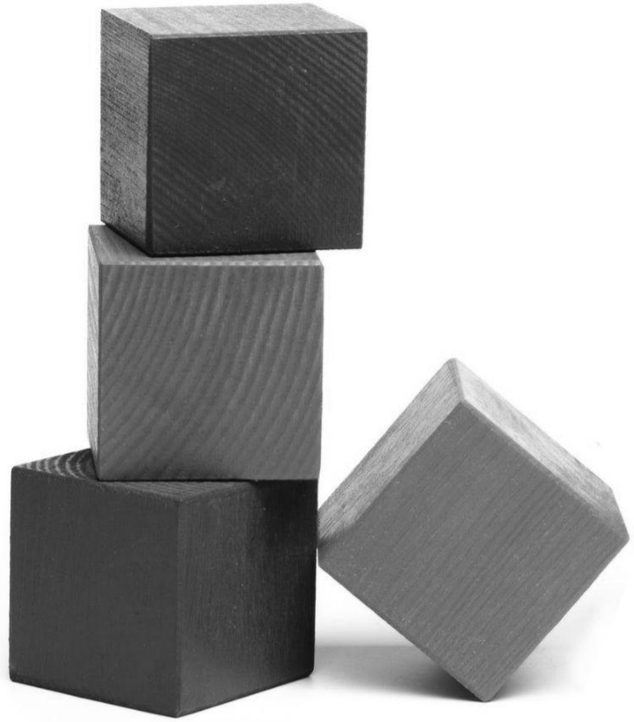
utf8



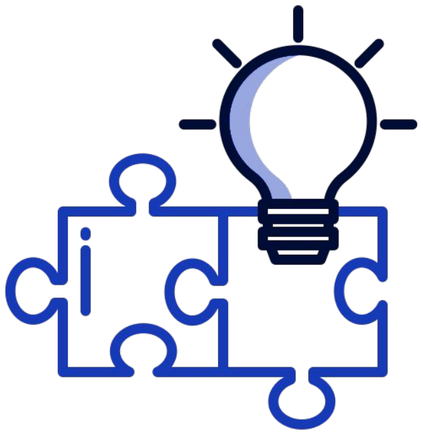
utf16



utf32



UTF-8 and UTF-16



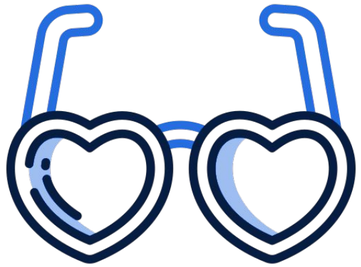
1. UTF-8 takes 1 to 4 bytes
2. UTF-16 takes 2 or 4 bytes
3. UTF-32 takes 4 bytes

4. UTF-8 – variable length
5. UTF-16 – variable length
6. UTF-32– fixed length

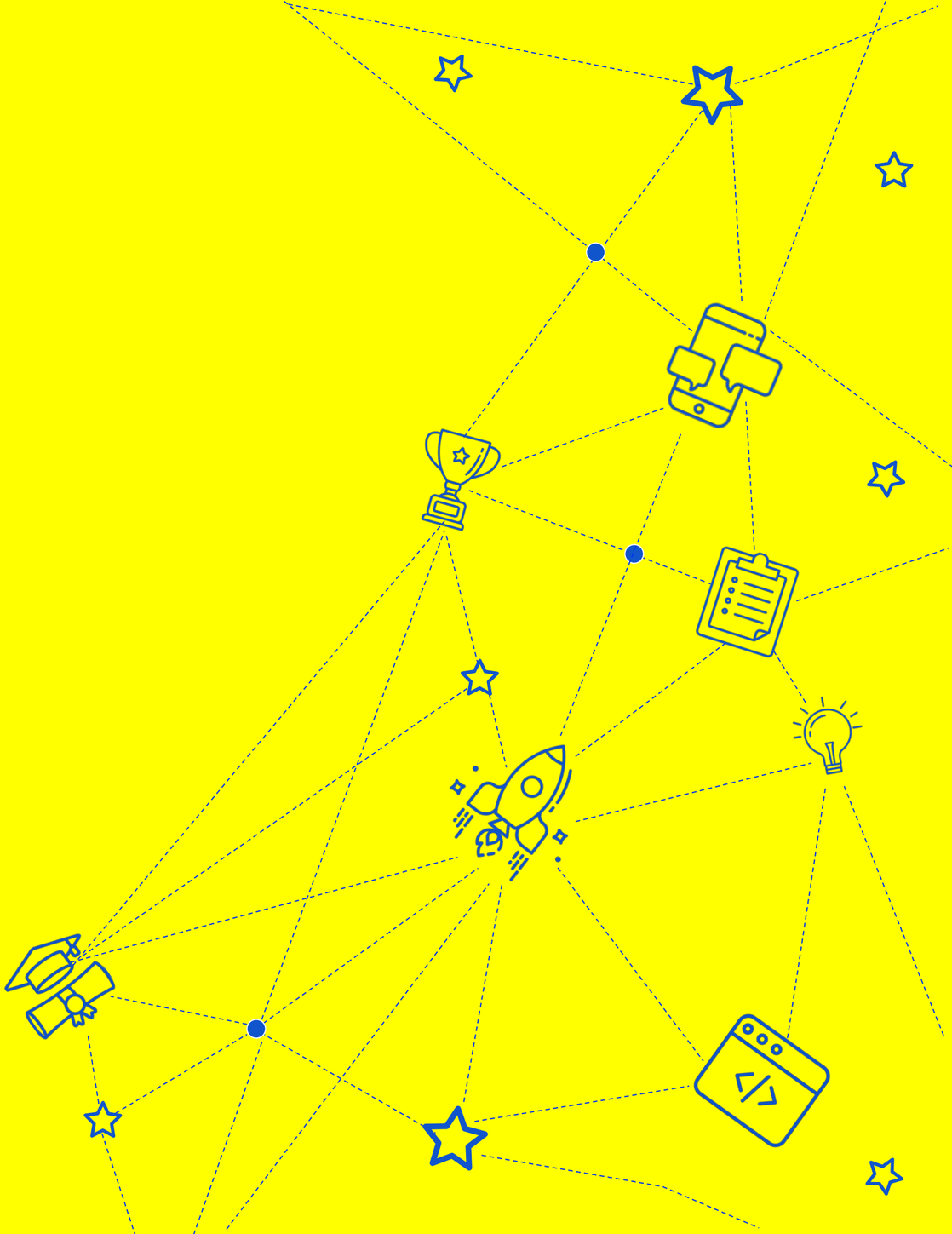
С кодировкой можно **ошибиться**

```
'Привет'.encode('utf8').decode('utf16')

# 뿐룐닄똈苑
```

Знайте
и любите ваши строки



Условия



if

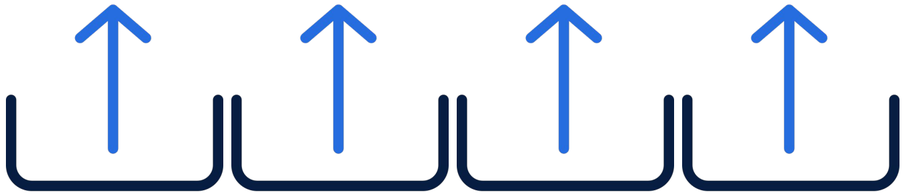
```
if x == 5:
    print('x is 5')
```

Важно! Отступы!

Так делать нельзя:

```
if x == 5:
print('x is 5')
```

Все тело условия должно иметь +4 пробела



Что может быть в условии `if`?

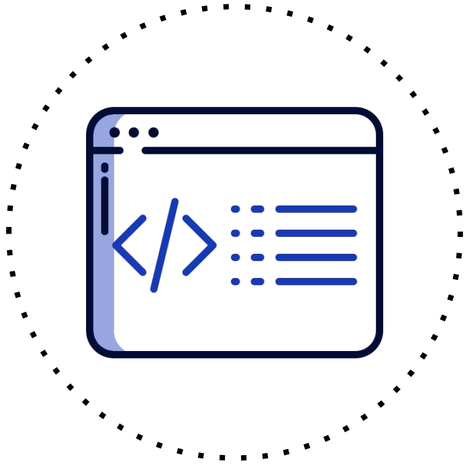
Любое выражение, которое возвращает `True` или `False`

```
'a' in 'abc'
# True

'z' not in 'abc'
# True
```

```
'Hello, ' + 'world!' == 'Hello, world!'
# True
```

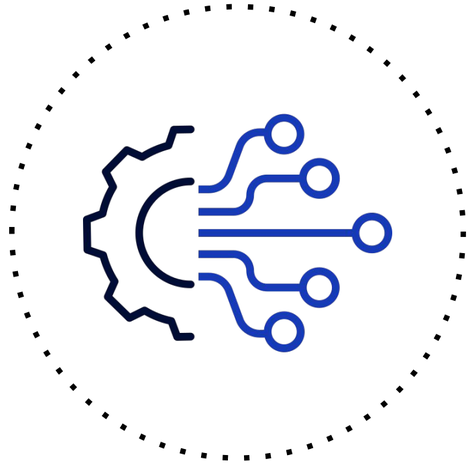
if / else

```
if x == 5:
    print('x is 5')
else:
    print('x is not 5 :(')
```

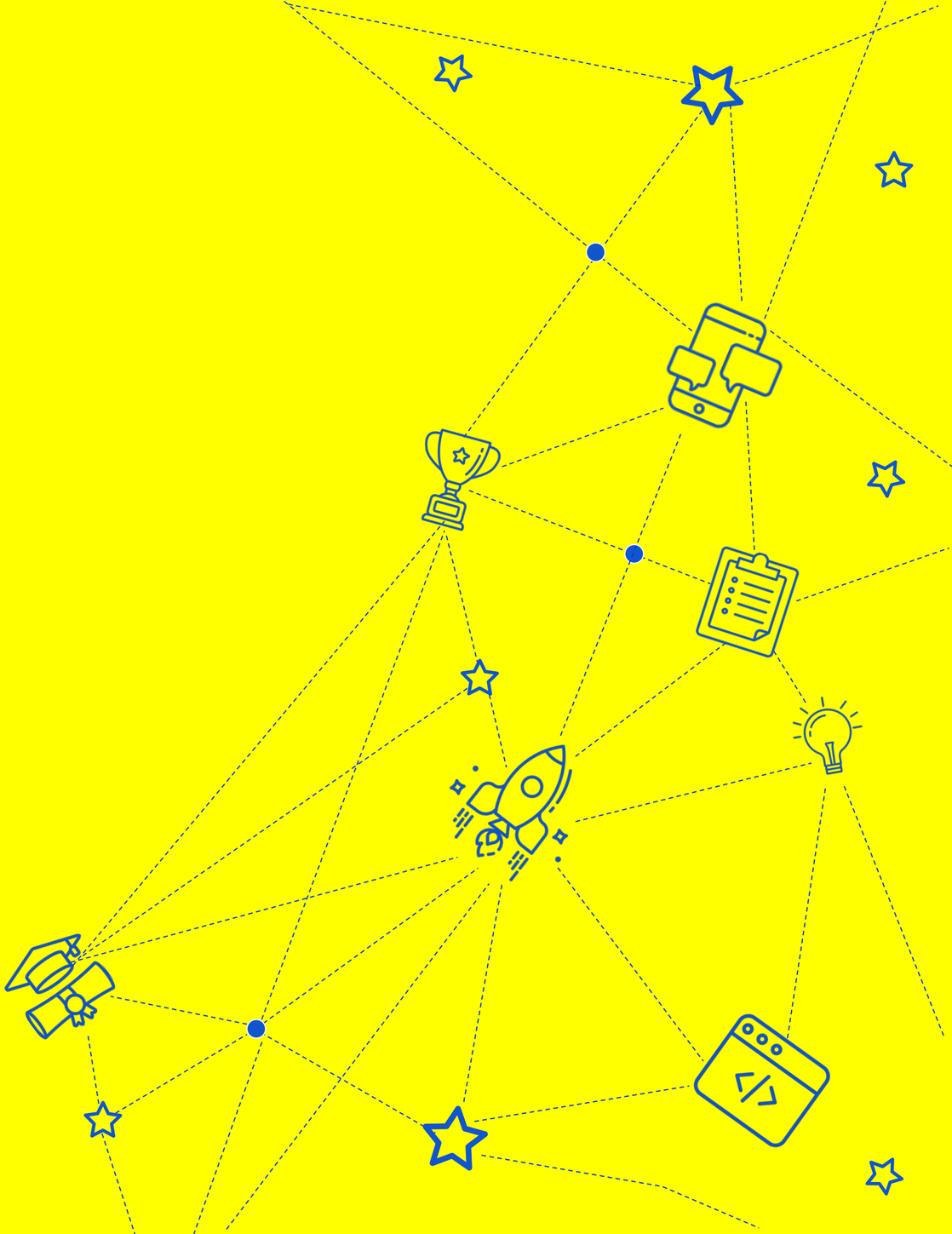


elif

```
if x == 5:
    print('x is 5')
elif x == 6:
    print('x is 6')

    print(':)')
else:
    print('x is not 5 and not 6')
```



Циклы



Цикл с предусловием

```
# Print all numbers from [0 to 2]
number = 0
while number < 3:
    print('Number', number)
    number = number + 1

# Number 0
# Number 1
# Number 2
```

Что может быть в условии `while`?

Любое выражение, которое возвращает `True` или `False`

```
x <= 5
```

```
True
```

```
my_string != ''
```

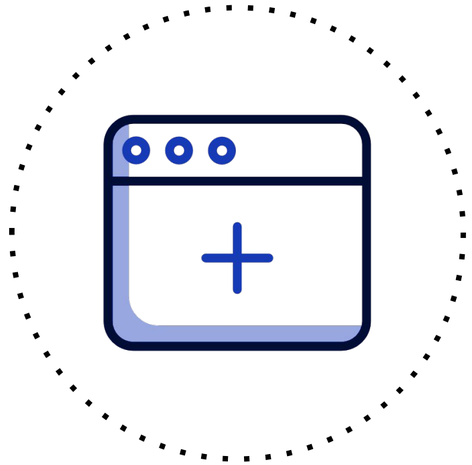
«Перебирающий итератор»

```
for char in 'abc':
    print(char)


# a
# b
# c
```

Что может быть в правой части `for`?

Только специальные «итерабельные» объекты (строки, списки, кортежи, т.д.)

```
'My string'
```

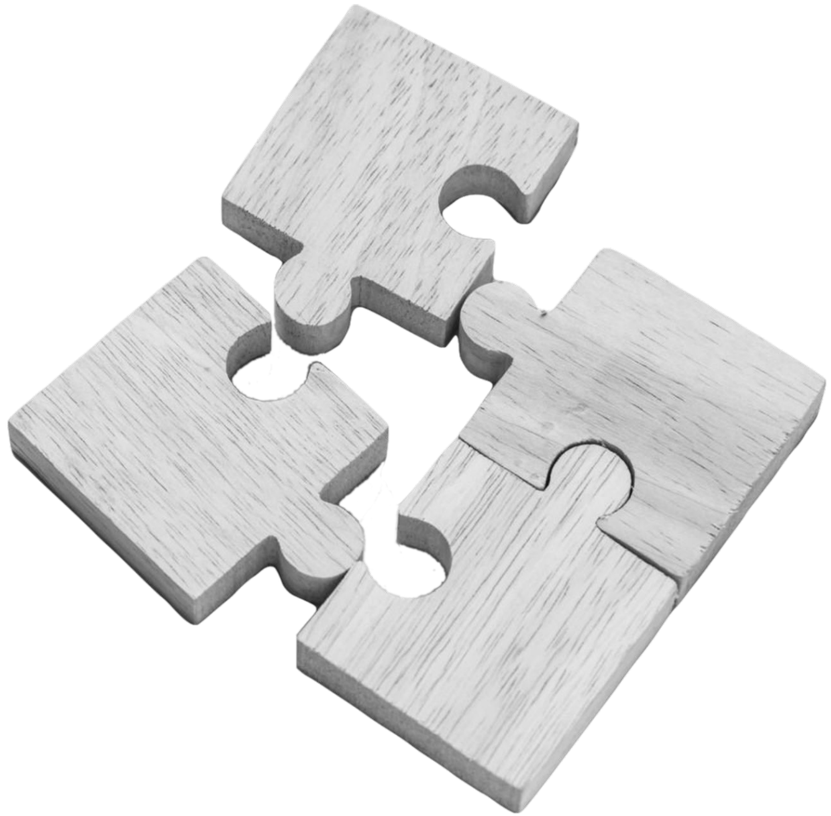


Вложенные циклы

```
for char1 in 'ab':
    for char2 in 'xy':
        print(char1, char2)


# a x
# a y
# b x
# b y
```

В Python почти
не используют `while`.
В основном только
для сложных алгоритмов.

В остальных случаях – `for`.



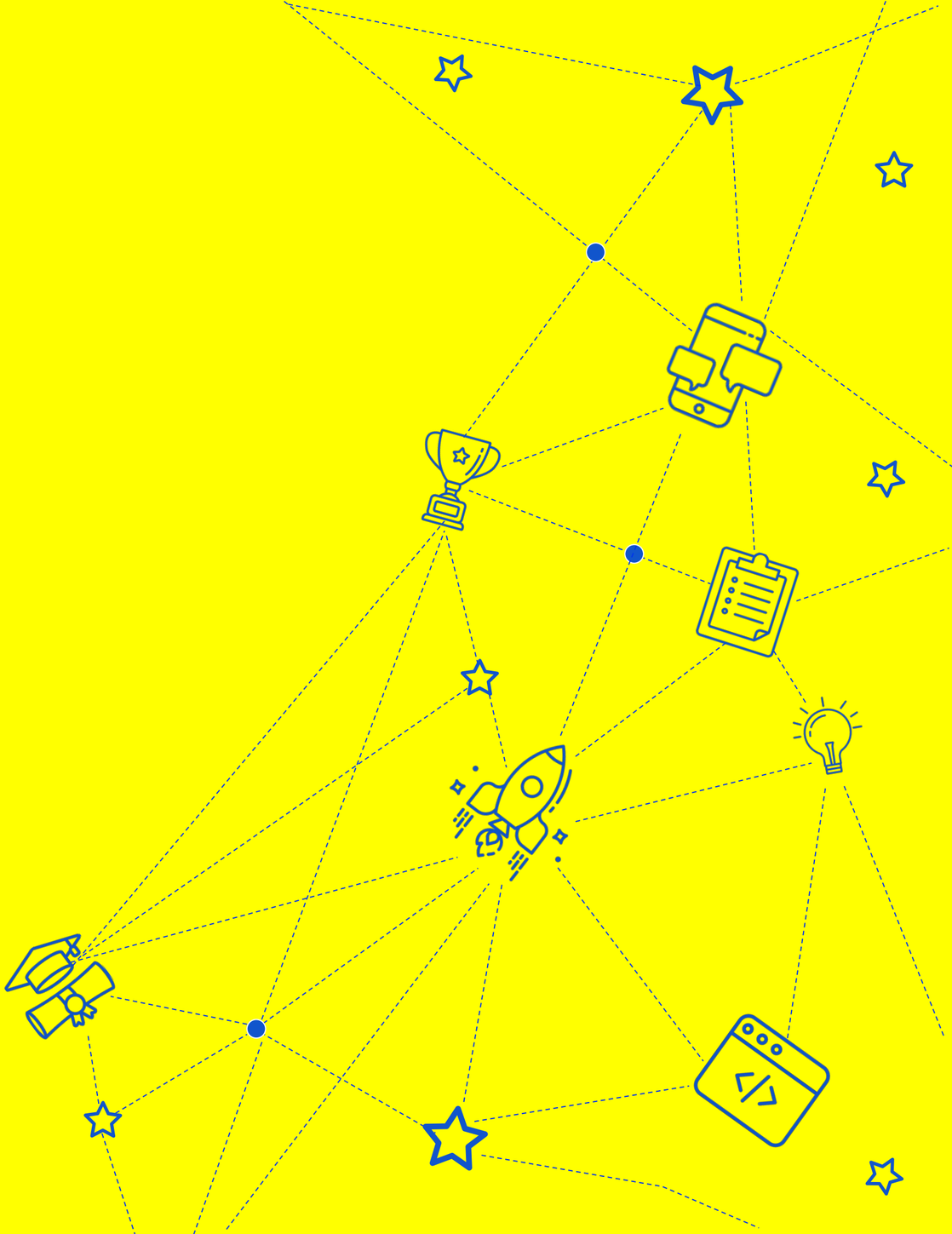
Что мы не упомянули про циклы в презентации

Ключевые слова:

- `break`
- `continue`
- `pass`
- `for/else`
- `while/else`

https://docs.python.org/3/tutorial/controlflow.html#break-and-continue-statements-and-else-clauses-on-loops

Домашнее задание

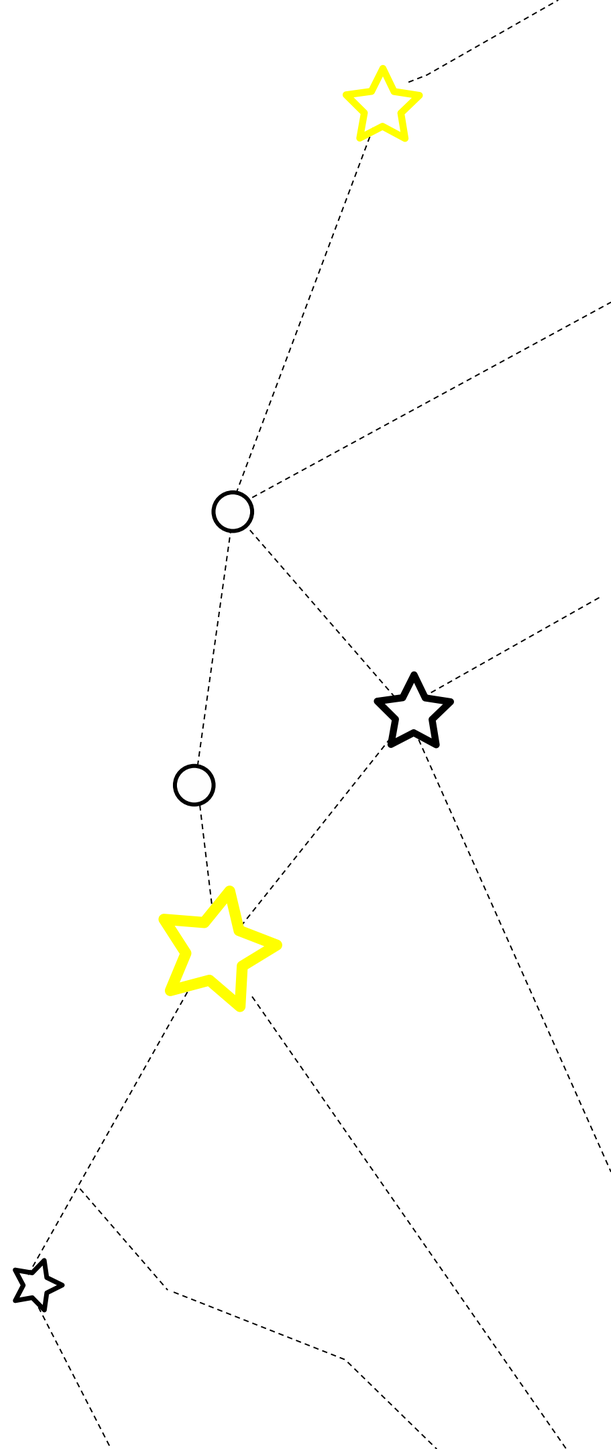


Домашнее задание

Задача: реализовать тестирование пользователя по пройденному материалу.

Требования:

- ★ Программа выводит в консоль текст загадок по одной и ждет ввода ответа от пользователя.
- ★ Программа после ввода ответа пользователя должна вывести в консоль результат. Если пользователь дал правильный ответ, то программа должна написать **"Ответ {ОТВЕТ_ПОЛЬЗОВАТЕЛЯ} верен"**. Вместо **"{ОТВЕТ_ПОЛЬЗОВАТЕЛЯ}"** необходимо подставить ответ, введенный пользователем. Если пользователь ввел неправильный ответ, то программа должна написать **"Неверный ответ."** и задать вопрос снова, пока пользователь не введет правильный ответ.
- ★ Программа задает пользователю 10 вопросов.



Домашнее задание

Примеры вопросов и правильных ответов для работы:

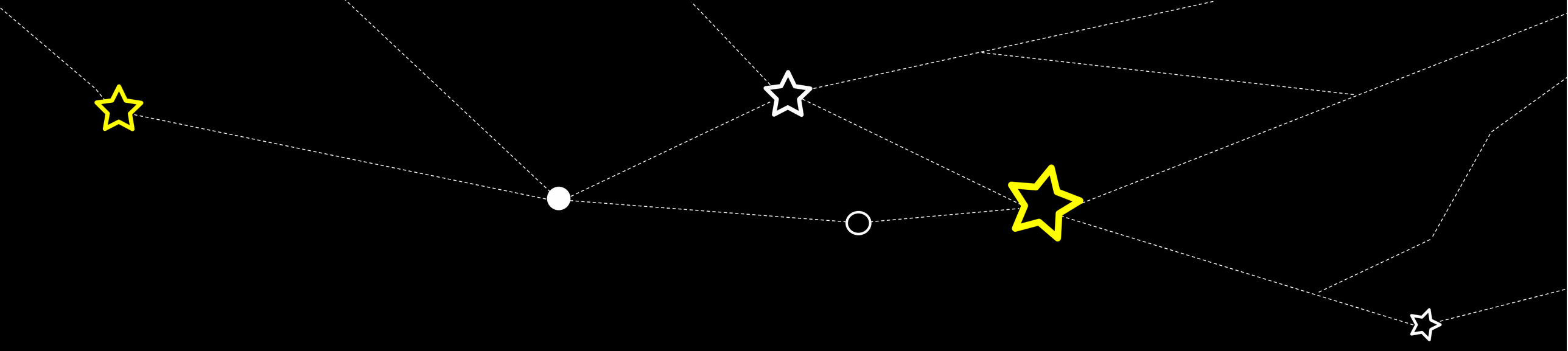
- «Какая версия языка сейчас актуальна?» – **«Python3»**
- «Какая кодировка используется в строках?» – **«UTF8»**
- «Сколько значений есть у bool?» – **«2»**

Остальные, нужно придумать самостоятельно
(из пройденного материала).

Домашнее задание

Дополнительные требования:

- Программа должна в конце игры сказать, сколько ответов дал пользователь: сколько из них было верных.
- Программа должна не учитывать регистр ответа пользователя, например: "utf8" и "UTF8" оба должны быть правильным ответом на вопрос "Какая кодировка используется в строках?"



СПАСИБО ЗА ВНИМАНИЕ

Никита Соболев

sobolevn