Основы языка программирования Python

Строки
(класс `str`)

Кодировки символов

ASCII - American Standard Code for Information Interchange

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Другие кодировки

CP866

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
9	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
B	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
C	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
D	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	▀	▄	▌	▐	■
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
F	Ё	ё	Є	є	Ї	ї	Ў	ў	°	∙	·	√	№	¤	■	

CP1251

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	Ъ	Ѓ	‚	ѓ	„	…	†	‡	□	‰	Љ	‹	Њ	Ќ	Ћ	Џ
9	ђ	‘	’	“	”	•	–	—		™	љ	›	њ	ќ	ћ	џ
A		Ў	ў	Ј	¤	Ґ	¦	§	Ё	©	Є	«	¬	-	®	Ї
B	°	±	І	і	ґ	µ	¶	·	ё	№	є	»	j	S	s	ї
C	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
D	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
E	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
F	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

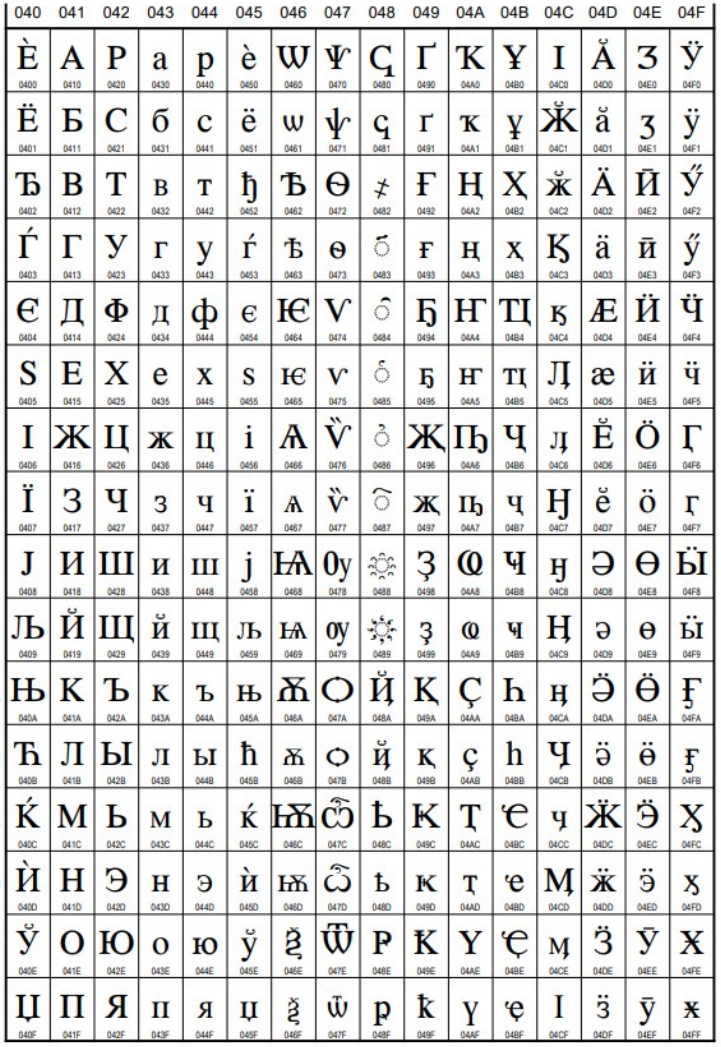
Unicode

Unicode 16.0 Character Code Charts
https://www.unicode.org/charts/

Текущая версия: Unicode 16.0 (сентябрь 2024 г.)

Количество символов: 155 063

Количество письменностей: 168


040 041 042 043 044 045 046 047 048 049 04A 04B 04C 04D 04E 04F
0 È А Р а р è Ѡ Ѱ Ҫ Ґ Ҟ Ұ І Ӑ З Ӱ
1 Ё Б С б с ё ѡ ѱ ҫ г ҟ ұ Ӂ ӑ з ӱ
2 Ђ В Т в т ђ Ђ Ө ≠ Ғ Ң Ҳ ӂ Ä Ӣ Ӳ
3 Ѓ Г У г у ѓ ђ ө ő ғ ң ҳ Ҷ ä ӣ ӳ
4 Є Д Ф д ф є Ю Ѵ ő Ҕ Н Ҵ ҷ Æ Й Ӵ
5 S Е Х е х s ю ѵ ő ҕ н ҵ Ӆ æ й ӵ
6 І Ж Ц ж ц і Ѧ Ѷ ѳ Ж Җ Ч ӆ Ё Ö Ҁ
7 Ї З Ч з ч ї ѧ ѷ ѳ ж җ ч Ӈ ё ö ҁ
8 Ј И Ш и ш ј Ҍ Ꙋ ҉ Ҙ ҈ Ҷ ӈ Э Ө Ы
9 Љ Й Щ й щ љ ҍ ꙋ ҉ ҙ ҉ ҷ ӈ ə ө ы
A Њ К Ъ к ъ њ Ӂ Ѻ Ӣ Ҟ Ҫ Һ ҥ Ә Ө Ғ
B Ћ Л Ы л ы ћ ӂ ѻ ӣ ҟ ҫ h Ҷ ә ө ғ
C Ќ М Ь м ь ќ Ҍ Ѽ Ь К Ҭ Ҿ ҷ Ӝ Ҽ Ҳ
D Ѝ Н Э н э ѝ ҍ ѽ ь к ҭ ҿ Ӎ ӝ ҽ ҳ
E Ў О Ю о ю ў Ҙ Ѿ Ҏ К Ү Ҿ ӎ Ӟ Ӯ Ҳ
F Џ П Я п я џ ҙ ѿ ҏ ҟ ү ҿ І ӟ ӯ ҳ


130E 130F 1310 1311 1312 1313 1314 1315 1316 1317 1318 1319 131A 131B
0 130E0 130F0 13100 13110 13120 13130 13140 13150 13160 13170 13180 13190 131A0 131B0
1 130E1 130F1 13101 13111 13121 13131 13141 13151 13161 13171 13181 13191 131A1 131B1
2 130E2 130F2 13102 13112 13122 13132 13142 13152 13162 13172 13182 13192 131A2 131B2
3 130E3 130F3 13103 13113 13123 13133 13143 13153 13163 13173 13183 13193 131A3 131B3
4 130E4 130F4 13104 13114 13124 13134 13144 13154 13164 13174 13184 13194 131A4 131B4
5 130E5 130F5 13105 13115 13125 13135 13145 13155 13165 13175 13185 13195 131A5 131B5
6 130E6 130F6 13106 13116 13126 13136 13146 13156 13166 13176 13186 13196 131A6 131B6
7 130E7 130F7 13107 13117 13127 13137 13147 13157 13167 13177 13187 13197 131A7 131B7
8 130E8 130F8 13108 13118 13128 13138 13148 13158 13168 13178 13188 13198 131A8 131B8
9 130E9 130F9 13109 13119 13129 13139 13149 13159 13169 13179 13189 13199 131A9 131B9
A 130EA 130FA 1310A 1311A 1312A 1313A 1314A 1315A 1316A 1317A 1318A 1319A 131AA 131BA
B 130EB 130FB 1310B 1311B 1312B 1313B 1314B 1315B 1316B 1317B 1318B 1319B 131AB 131BB
C 130EC 130FC 1310C 1311C 1312C 1313C 1314C 1315C 1316C 1317C 1318C 1319C 131AC 131BC
D 130ED 130FD 1310D 1311D 1312D 1313D 1314D 1315D 1316D 1317D 1318D 1319D 131AD 131BD
E 130EE 130FE 1310E 1311E 1312E 1313E 1314E 1315E 1316E 1317E 1318E 1319E 131AE 131BE
F 130EF 130FF 1310F 1311F 1312F 1313F 1314F 1315F 1316F 1317F 1318F 1319F 131AF 131BF

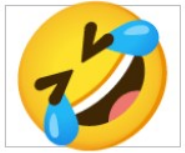
U+1F600 : 
U+1F603 : 
U+1F604 : 
U+1F601 : 
U+1F606 : 
U+1F605 : 
U+1F923 : 
U+1F602 : 
U+1F642 : 
U+1F643 : 
U+1F609 : 
U+1F60A : 
U+1F607 : 
U+1F60D : 
U+1F929 : 
U+1F618 : 
U+1F617 : 

U+263A : 
U+1F61A : 
U+1F619 : 
U+1F60B : 
U+1F61B : 
U+1F61C : 
U+1F92A : 
U+1F61D : 
U+1F911 : 
U+1F917 : 
U+1F92D : 
U+1F92B : 
U+1F914 : 
U+1F60E : 
U+1F913 : 
U+1F9D0 : 
U+1F615 : 

U+1F61F : 
U+1F641 : 
U+2639 : 
U+1F62E : 
U+1F62F : 
U+1F632 : 
U+1F633 :
U+1F626 :
U+1F627 :
U+1F628 :
U+1F630 :
U+1F625 :
U+1F622 :
U+1F62D :
U+1F631 :
U+1F616 :
U+1F623 :

U+1F61E :
U+1F648 :
U+1F649 :
U+1F64A :
U+1F44B :
U+1F91A :
U+1F590 :
U+270B :
U+1F596 :
U+1F44C :
U+270C :
U+1F91E :
U+1F91F :
U+1F918 :
U+1F919 :
U+1F44D :
U+1F44E :

Unicode Character “🤣” (U+1F923)



Name:	Rolling On the Floor Laughing[1]
Unicode Version:	9.0 (June 2016)[2]
Block:	Supplemental Symbols and Pictographs, U+1F900 - U+1F9FF[3]
Plane:	Supplementary Multilingual Plane, U+10000 - U+1FFFF[3]
Script:	Code for undetermined script (Zyyy) [4]
Category:	Other Symbol (So) [1]
Bidirectional Class:	Other Neutral (ON) [1]
Combining Class:	Not Reordered (0) [1]
Character is Mirrored:	No [1]
HTML Entity:	`&#129315;` `&#x1F923;`
UTF-8 Encoding:	`0xF0 0x9F 0xA4 0xA3`
UTF-16 Encoding:	`0xD83E 0xDD23`
UTF-32 Encoding:	`0x0001F923`

Кодировки Unicode

- UTF-32 — 32 бита на символ
- UTF-16 — 16 или 32 бита на символ
- UTF-8 — от 8 до 32 бит на символ

UTF — Unicode Transformation Format

Способы создания строк

```python
# Однострочные (single line) строки
foo = "Hello"
bar = 'Привет'

print("type(foo):", type(foo))
print("type(bar):", type(bar))
```

```
type(foo): <class 'str'>
type(bar): <class 'str'>
```

```python
# Многострочные (multiline) строки

foo = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""

bar = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''

print("type(foo):", type(foo))
print("type(bar):", type(bar))

print()
print("foo:", foo)
print()
print("bar:", bar)
```

```
type(foo): <class 'str'>
type(bar): <class 'str'>

foo: Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.

bar: Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

```python
# Отступы и многострочный текст

if True:
    foo = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""

    bar = """Lorem ipsum dolor sit amet,
    consectetur adipiscing elit,
    sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua."""

    print("foo:", foo)
    print()
    print("bar:", bar)
```

```
foo: Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.

bar: Lorem ipsum dolor sit amet,
    consectetur adipiscing elit,
    sed do eiusmod tempor incididunt
    ut labore et dolore magna aliqua.
```

```
# Использование кавычек в тексте

foo = "Этот текст содержит 'одинарные' кавычки"
bar = 'Этот текст содержит "двойные" кавычки'
spam = 'Этот текст содержит \'одинарные\' кавычки'
eggs = "Этот текст содержит \"двойные\" кавычки"

print("foo: ", foo)
print("bar: ", bar)
print("spam:", spam)
print("eggs:", eggs)
```

```
foo:  Этот текст содержит 'одинарные' кавычки
bar:  Этот текст содержит "двойные" кавычки
spam: Этот текст содержит 'одинарные' кавычки
eggs: Этот текст содержит "двойные" кавычки
```

```
# Использование кавычек в тексте

foo = '''Этот текст содержит 'одинарные' кавычки'''
bar = """Этот текст содержит "двойные" кавычки"""

print("foo: ", foo)
print("bar: ", bar)
```

```
foo:  Этот текст содержит 'одинарные' кавычки
bar:  Этот текст содержит "двойные" кавычки
```

```
# Использование обратных слешей в тексте

foo = "Этот текст содержит обратные \слеши\\"
bar = "Этот текст содержит обратные \\слеши\\"

print("foo: ", foo)
print("bar: ", bar)
```

```
foo:  Этот текст содержит обратные \слеши\
bar:  Этот текст содержит обратные \слеши\
```

```
foo = "Lorem ipsum dolor sit amet,\nconsectetur adipiscing elit"
bar = "Lorem ipsum dolor sit amet,\tconsectetur adipiscing elit"

print("foo: ", foo)
print()
print("bar: ", bar)
```

```
foo:  Lorem ipsum dolor sit amet,
consectetur adipiscing elit

bar:  Lorem ipsum dolor sit amet,    consectetur adipiscing elit
```

```
# Использование "сырых" (raw) строк

foo = "Этот текст\\tсодержит \\\\ слишком много \\\\ обратных слешей\\n"
bar = r'Этот текст\tсодержит \\ слишком много \\ обратных слешей\n'

print("foo: ", foo)
print("bar: ", bar)
```

```
foo:  Этот текст\tсодержит \\ слишком много \\ обратных слешей\n
bar:  Этот текст\tсодержит \\ слишком много \\ обратных слешей\n
```

```python
# Вставка символов Unicode по коду

foo = "Это символ Unicode - \u03b4"
bar = "Это символ Unicode - \U0001f923"

print("foo:", foo)
print("bar:", bar)
```

```
foo: Это символ Unicode - δ
bar: Это символ Unicode - 🤣
```

```python
# Вставка символов Unicode по названию

foo = "Это символ Unicode - \N{Greek Small Letter Delta}"
bar = "Это символ Unicode - \N{rolling on the floor laughing}"

print("foo:", foo)
print("bar:", bar)
```

```
foo: Это символ Unicode - δ
bar: Это символ Unicode - 🤣
```

```python
# Вставка символов Unicode непосредственно в текст

foo = "Это символ Unicode - б"
bar = "Это символ Unicode - 🤣"

print("foo:", foo)
print("bar:", bar)
```

```
foo: Это символ Unicode - б
bar: Это символ Unicode - 🤣
```

```python
# Создание строкового представления объектов с помощью функции str()

foo = 10
foo_str = str(foo)

bar = 20.5
bar_str = str(bar)

spam = 15 - 5j
spam_str = str(spam)

eggs = {"key_1": 10, "key_2": 20, "key_3": 30}
eggs_str = str(eggs)

baz = [10, 20, 42]
baz_str = str(baz)

print("foo_str: ", foo_str)
print("bar_str: ", bar_str)
print("spam_str:", spam_str)
print("eggs_str:", eggs_str)
print("baz_str: ", baz_str)
```

```
foo_str:  10
bar_str:  20.5
spam_str: (15-5j)
eggs_str: {'key_1': 10, 'key_2': 20, 'key_3': 30}
baz_str:  [10, 20, 42]
```

```python
# Создание строкового представления объектов с помощью функции repr()

foo = 10
foo_str = str(foo)
foo_repr = repr(foo)

bar = 20.5
bar_str = str(bar)
bar_repr = repr(bar)

baz = [10, 20, 42]
baz_str = str(baz)
baz_repr = repr(baz)

print("foo_str: ", foo_str)
print("foo_repr: ", foo_repr)

print("bar_str: ", bar_str)
print("bar_repr: ", bar_repr)

print("baz_str: ", baz_str)
print("baz_repr: ", baz_repr)
```

```
foo_str:  10
foo_repr:  10
bar_str:  20.5
bar_repr:  20.5
baz_str:  [10, 20, 42]
baz_repr:  [10, 20, 42]
```

```python
# Сравнение результатов работы функций
# str() и repr() для более сложных объектов

import datetime

date = datetime.datetime(2024, 3, 10, 18, 40, 2)

print("str(date): ", str(date))
print("repr(date):", repr(date))
print("date:      ", date)
```

```
str(date):  2024-03-10 18:40:02
repr(date): datetime.datetime(2024, 3, 10, 18, 40, 2)
date:       2024-03-10 18:40:02
```

Действия со строками

```
# Получение длины строки

foo = "Hello"
bar = "Привет"
spam = "你好"
baz = "\U0001f923"

print("len(foo): ", len(foo))
print("len(bar): ", len(bar))
print("len(spam):", len(spam))
print("len(baz): ", len(baz))
```

```
len(foo):  5
len(bar):  6
len(spam): 2
len(baz):  1
```

```python
# Получение символов по индексу
foo = "Привет \U0001f923"

print("foo[0]: ", foo[0])
print("foo[3]: ", foo[3])
print("foo[-1]:", foo[-1])
```

```
foo[0]:  П
foo[3]:  в
foo[-1]: 🤣
```

```
# Строка - неизменяемый объект
foo = "Привет"

# Ошибка!
foo[0] = "X"
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

```
# Получение подстроки
spam = "Lorem ipsum dolor sit amet, consectetur adipiscing elit"

print("spam[2:10]:   ", spam[2:10])
print("spam[2:14:2]: ", spam[2:14:2])
print("spam[9:1:-1]: ", spam[9:1:-1])
print("spam[:5]:     ", spam[:5])
print("spam[-4:]:    ", spam[-4:])
print("spam[:]:      ", spam[:])
print("spam[::-1]:   ", spam[::-1])
```

```
spam[2:10]:    rem ipsu
spam[2:14:2]:  rmismd
spam[9:1:-1]:  uspi mer
spam[:5]:      Lorem
spam[-4:]:     elit
spam[:]:       Lorem ipsum dolor sit amet, consectetur adipiscing elit
spam[::-1]:    tile gnicsipida rutetcesnoc ,tema tis rolod muspi meroL
```

```python
# Использование операторов in / not in

spam = "Lorem ipsum dolor sit amet, consectetur adipiscing elit"
foo = "ipsum"
bar = "hello"

print("foo in spam:    ", foo in spam)
print("bar not in spam:", bar not in spam)
```

```
foo in spam:     True
bar not in spam: True
```

```
print(dir(str))
```

['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

```
# Использование методов startswith() / endswith()

foo = "images/picture.png"
bar = "data/application.exe"

print("foo.startswith('images'):", foo.startswith('images'))
print("bar.startswith('images'):", bar.startswith('images'))
print("foo.endswith('.png'):    ", foo.endswith('.png'))
print("bar.endswith('.png'):    ", bar.endswith('.png'))
```

```
foo.startswith('images'): True
bar.startswith('images'): False
foo.endswith('.png'):     True
bar.endswith('.png'):     False
```

`str.`**`find`**`(`*`sub`*`[, `*`start`*`[, `*`end`*`]])`

Return the lowest index in the string where substring *sub* is found within the slice `s[start:end]`.

Optional arguments *start* and *end* are interpreted as in slice notation. Return `-1` if *sub* is not found.

```
# Использование метода find()

foo = "Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet"

print("foo.find('ipsum'):        ", foo.find('ipsum'))
print("foo.find('Ipsum'):        ", foo.find('Ipsum'))
print("foo.find('ipsum', 7):     ", foo.find('ipsum', 7))
print("foo.find('ipsum', 35, 60):", foo.find('ipsum', 35, 60))
```

```
foo.find('ipsum'):         6
foo.find('Ipsum'):         -1
foo.find('ipsum', 7):      34
foo.find('ipsum', 35, 60): -1
```

```
# Нахождение всех позиций заданной подстроки

foo = "Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet"
sub = "ipsum"

result = []
pos = 0

while pos != -1:
    pos = foo.find(sub, pos)
    if pos != -1:
        result.append(pos)
        pos += 1

print("result:", result)
```

```
result: [6, 34, 62]
```

```python
# Нахождение всех позиций заданной подстроки
# Использование оператора :=

foo = "Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet"
sub = "ipsum"

result = []
pos = 0

while (pos := foo.find(sub, pos)) != -1:
    result.append(pos)
    pos += 1

print("result:", result)
```

```
result: [6, 34, 62]
```

```python
# Использование методов lower() / upper() / capitalize()

foo = "lorem IPSUM dolor SIT amet"

foo_lower = foo.lower()
foo_upper = foo.upper()
foo_cap = foo.capitalize()

print("foo:       ", foo)
print("foo_lower: ", foo_lower)
print("foo_upper: ", foo_upper)
print("foo_cap:   ", foo_cap)
```

```
foo:        lorem IPSUM dolor SIT amet
foo_lower:  lorem ipsum dolor sit amet
foo_upper:  LOREM IPSUM DOLOR SIT AMET
foo_cap:    Lorem ipsum dolor sit amet
```

```python
# Использование методов strip() / lstrip() / rstrip()

foo = "    Lorem ipsum    "

foo_strip = foo.strip()
foo_lstrip = foo.lstrip()
foo_rstrip = foo.rstrip()

print("foo:        |", foo, "|", sep="")
print("foo_strip:  |", foo_strip, "|", sep="")
print("foo_lstrip: |", foo_lstrip, "|", sep="")
print("foo_rstrip: |", foo_rstrip, "|", sep="")
```

```
foo:        |    Lorem ipsum    |
foo_strip:  |Lorem ipsum|
foo_lstrip: |Lorem ipsum    |
foo_rstrip: |    Lorem ipsum|
```

```python
# Использование методов removeprefix() / removesuffix()

foo = "images/picture.png"
bar = "data/application.exe"

print("foo:                          ", foo)
print("bar:                          ", bar)
print()
print("foo.removeprefix('images/'):",foo.removeprefix('images/'))
print("foo.removesuffix('.png'):   ", foo.removesuffix('.png'))
print()
print("bar.removeprefix('images/'):", bar.removeprefix('images/'))
print("bar.removesuffix('.png'):   ", bar.removesuffix('.png'))
```

```
foo:                          images/picture.png
bar:                          data/application.exe

foo.removeprefix('images/'): picture.png
foo.removesuffix('.png'):    images/picture

bar.removeprefix('images/'): data/application.exe
bar.removesuffix('.png'):    data/application.exe
```

`str.replace(old, new[, count])`

Return a copy of the string with all occurrences of substring *old* replaced by *new*. If the optional argument *count* is given, only the first *count* occurrences are replaced.

```
# Использование метода replace()

foo = "Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet"
old = "ipsum"
new = "*****"

bar = foo.replace(old, new)
baz = foo.replace(old, new, 1)

print("foo: ", foo)
print("bar: ", bar)
print("baz: ", baz)
```

```
foo:  Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet
bar:  Lorem ***** dolor sit amet, Lorem ***** dolor sit amet, Lorem ***** dolor sit amet
baz:  Lorem ***** dolor sit amet, Lorem ipsum dolor sit amet, Lorem ipsum dolor sit amet
```

```
# Использование метода split()

foo = "a=10, b=20, c=30"
bar = foo.split(", ")

print("bar:", bar)
```

```
bar: ['a=10', 'b=20', 'c=30']
```

```python
# Использование метода join()

bar = ["a=10", "b=20", "c=30"]
foo = ", ".join(bar)

print("foo:", foo)
```

```
foo: a=10, b=20, c=30
```

Форматирование строк с использованием оператора "%"


```
# Плохой способ форматирования строк
foo = 42
bar = 12.5
eggs = "hello"

spam = "f = " + str(foo) + "; b = " + str(bar) + "; e = " + str(eggs) + "."
print("spam:", spam)
```

```
spam: f = 42; b = 12.5; e = hello.
```

```
# Использование оператора "%" для форматирования строк

foo = 42
bar = 12.5
eggs = "hello"

spam = "f = %d; b = %f; e = %s." % (foo, bar, eggs)
print("spam:", spam)
```

```
spam: f = 42; b = 12.500000; e = hello.
```

Символ форматирования	Формат
`d` или `i`	Целое число в десятичной форме
`o`	Целое число в восьмеричной форме
`x` или `X`	Целое число в шестнадцатеричной форме
`e` или `E`	Число с плавающей точкой в экспоненциальной форме
`f` или `F`	Число с плавающей точкой
`g` или `G`	Число с плавающей точкой. Формат подбирается автоматически
`c`	Одиночный символ
`r`	Строка. Для преобразования используется repr()
`s`	Строка. Для преобразования используется str()
`a`	Строка. Для преобразования используется ascii()
`%`	%% заменяется на знак процента

```
# Различные представления целых чисел
foo = 42

spam = """%%d: %d
%%o: %o
%%x: %x
%%f: %f
%%e: %e
%%g: %g
""" % (foo, foo, foo, foo, foo, foo)

print(spam)
```

```
%d: 42
%o: 52
%x: 2a
%f: 42.000000
%e: 4.200000e+01
%g: 42
```

```
# Использование дополнительных параметров форматирования
foo = 42.5

spam = """|%%14f| :     |%14f|
|%%-14f|:     |%-14f|
|%%14.3f|:    |%14.3f|
|%%-14.3f|:   |%-14.3f|
|%%+-14.3f|:  |%+-14.3f|
|%% -14.3f|:  |% -14.3f|
|%%014.3f|:   |%014.3f|
""" % ((foo,) * 7)

print(spam)
```

```
|%14f| :     |     42.500000|
|%-14f|:     |42.500000     |
|%14.3f|:    |        42.500|
|%-14.3f|:   |42.500        |
|%+-14.3f|:  |+42.500       |
|% -14.3f|:  | 42.500       |
|%014.3f|:   |0000000042.500|
```

```
# Использование именованных параметров

foo = 42
bar = 12.5
eggs = "hello"

spam = ("f = %(f)d; b = %(b)f; e = %(e)s."
        % {"e": eggs, "f": foo, "b": bar})
print("spam:", spam)
```

```
spam: f = 42; b = 12.500000; e = hello.
```

```
from numpy import linspace
from numpy.random import rand

freq = linspace(10e9, 12e9, 11)
col1 = rand(len(freq)) * 2 - 1
col2 = rand(len(freq)) * 2 - 1

for f, c1, c2 in zip(freq, col1, col2):
    print("%-15.3e %- 12.3f %- 12.3f" % (f, c1, c2))
```

```
1.000e+10       -0.942       -0.425
1.020e+10        0.430       -0.547
1.040e+10        0.050        0.563
1.060e+10        0.025       -0.402
1.080e+10        0.452        0.173
1.100e+10        0.939        0.413
1.120e+10       -0.850       -0.859
1.140e+10        0.053       -0.635
1.160e+10       -0.043        0.249
1.180e+10        0.954        0.713
1.200e+10        0.347        0.999
```

Форматирование строк с использованием метода `str.format()`

```
# Использование метода str.format() для форматирования строк

foo = 42
bar = 12.5
eggs = "hello"

spam = "f = {}; b = {}; e = {}.".format(foo, bar, eggs)
print("spam:", spam)
```

```
spam: f = 42; b = 12.5; e = hello.
```

```
# Использование нумерованных параметров

foo = 42
bar = 12.5
eggs = "hello"

spam = ("f = {0}; b = {1}; e = {2}; foo = {0}."
        .format(foo, bar, eggs))

print("spam:", spam)
```

```
spam: f = 42; b = 12.5; e = hello; foo = 42.
```

```
# Использование именованных параметров

foo = 42
bar = 12.5
eggs = "hello"

spam = ("f = {foo}; b = {bar}; e = {spam}."
        .format(foo=foo, spam=eggs, bar=bar))

print("spam:", spam)
```

```
spam: f = 42; b = 12.5; e = hello.
```

```
# Различные представления целых чисел
foo = 42

spam = """{{foo}}: {foo}
{{foo:d}}: {foo:d}
{{foo:b}}: {foo:b}
{{foo:o}}: {foo:o}
{{foo:x}}: {foo:x}
{{foo:X}}: {foo:X}
""".format(foo=foo)

print(spam)
```

```
{foo}: 42
{foo:d}: 42
{foo:b}: 101010
{foo:o}: 52
{foo:x}: 2a
{foo:X}: 2A
```

```
# Различные представления чисел с плавающей точкой
foo = 0.425

spam = """{{foo}}: {foo}
{{foo:g}}: {foo:g}
{{foo:f}}: {foo:f}
{{foo:e}}: {foo:e}
{{foo:E}}: {foo:E}
{{foo:%}}: {foo:%}
""".format(foo=foo)

print(spam)
```

```
{foo}: 0.425
{foo:g}: 0.425
{foo:f}: 0.425000
{foo:e}: 4.250000e-01
{foo:E}: 4.250000E-01
{foo:%}: 42.500000%
```

```
# Использование дополнительных параметров форматирования
foo = 42.5

spam = """|{{foo:14f}}| :     |{foo:14f}|
|{{foo:<14f}}|:      |{foo:<14f}|
|{{foo:14.3f}}|:     |{foo:14.3f}|
|{{foo:<14.3f}}|:    |{foo:<14.3f}|
|{{foo:^14.3f}}|:    |{foo:^14.3f}|
|{{foo:<+14.3f}}|:   |{foo:<+14.3f}|
|{{foo:< 14.3f}}|:   |{foo:< 14.3f}|
|{{foo:<014.3f}}|:   |{foo:<014.3f}|
|{{foo:014.3f}}|:    |{foo:014.3f}|
|{{foo:_<14.3f}}|:   |{foo:_<14.3f}|
|{{foo:_>14.3f}}|:   |{foo:_>14.3f}|
|{{foo:_^14.3f}}|:   |{foo:_^14.3f}|
""".format(foo=foo)

print(spam)
```

```
|{foo:14f}| :     |     42.500000|
|{foo:<14f}|:     |42.500000     |
|{foo:14.3f}|:    |        42.500|
|{foo:<14.3f}|:   |42.500        |
|{foo:^14.3f}|:   |    42.500    |
|{foo:<+14.3f}|:  |+42.500       |
|{foo:< 14.3f}|:  | 42.500       |
|{foo:<014.3f}|:  |42.50000000000|
|{foo:014.3f}|:   |0000000042.500|
|{foo:_<14.3f}|:  |42.500________|
|{foo:_>14.3f}|:  |________42.500|
|{foo:_^14.3f}|:  |____42.500____|
```

```
# Использование индексации
foo = [10.1, 20.2, 40.3, 60.4]

bar = "foo[0] = {0[0]:.3f}; foo[1] = {0[1]:.3f}".format(foo)
baz = "foo[0] = {foo[0]:.3f}; foo[1] = {foo[1]:.3f}".format(foo=foo)

print("bar:", bar)
print("baz:", baz)
```

```
bar: foo[0] = 10.100; foo[1] = 20.200
baz: foo[0] = 10.100; foo[1] = 20.200
```

```
# Использование индексации
foo = [10.1, 20.2, 40.3, 60.4]

# Ошибка! Отрицательные индексы не работают
bar = "foo[-1] = {foo[-1]}".format(foo=foo)
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: list indices must be integers or slices, not str
```

```
from numpy import linspace
from numpy.random import rand

freq = linspace(10e9, 12e9, 11)
col1 = rand(len(freq)) * 2 - 1
col2 = rand(len(freq)) * 2 - 1

for f, c1, c2 in zip(freq, col1, col2):
    print("{freq:<15.3e}{col1:< 12.3f}{col2:< 12.3f}"
          .format(freq=f, col1=c1, col2=c2))
```

```
1.000e+10      -0.996      -0.922
1.020e+10      -0.736      -0.269
1.040e+10       0.197      -0.591
1.060e+10      -0.359       0.396
1.080e+10       0.223       0.648
1.100e+10      -0.816       0.834
1.120e+10       0.823      -0.553
1.140e+10       0.637       0.948
1.160e+10      -0.925       0.814
1.180e+10      -0.252      -0.216
1.200e+10      -0.513       0.437
```

Форматирование с использованием f-строк

```python
# Использование f-строк
foo = f"2 x 2 = {2 * 2}"

print("foo:", foo)
```

```
foo: 2 x 2 = 4
```

```
# Использование f-строк

foo = 42
bar = 12.5
eggs = "hello"

spam = f"foo = {foo}; bar = {bar}; eggs = {eggs}."
print("spam:", spam)
```

```
spam: foo = 42; bar = 12.5; eggs = hello.
```

```
# Различные представления целых чисел
foo = 42

spam = f"""{{foo}}: {foo}
{{foo:d}}: {foo:d}
{{foo:b}}: {foo:b}
{{foo:o}}: {foo:o}
{{foo:x}}: {foo:x}
{{foo:X}}: {foo:X}
"""

print(spam)
```

```
{foo}: 42
{foo:d}: 42
{foo:b}: 101010
{foo:o}: 52
{foo:x}: 2a
{foo:X}: 2A
```

```
# Различные представления чисел с плавающей точкой
foo = 0.425

spam = f"""{{foo}}: {foo}
{{foo:g}}: {foo:g}
{{foo:f}}: {foo:f}
{{foo:e}}: {foo:e}
{{foo:E}}: {foo:E}
{{foo:%}}: {foo:%}
"""

print(spam)
```

```
{foo}: 0.425
{foo:g}: 0.425
{foo:f}: 0.425000
{foo:e}: 4.250000e-01
{foo:E}: 4.250000E-01
{foo:%}: 42.500000%
```

```
# Использование дополнительных параметров форматирования
foo = 42.5

spam = f"""|{{foo:14f}}| :     |{foo:14f}|
|{{foo:<14f}}|:      |{foo:<14f}|
|{{foo:14.3f}}|:     |{foo:14.3f}|
|{{foo:<14.3f}}|:    |{foo:<14.3f}|
|{{foo:^14.3f}}|:    |{foo:^14.3f}|
|{{foo:<+14.3f}}|:   |{foo:<+14.3f}|
|{{foo:< 14.3f}}|:   |{foo:< 14.3f}|
|{{foo:<014.3f}}|:   |{foo:<014.3f}|
|{{foo:014.3f}}|:    |{foo:014.3f}|
|{{foo:_<14.3f}}|:   |{foo:_<14.3f}|
|{{foo:_>14.3f}}|:   |{foo:_>14.3f}|
|{{foo:_^14.3f}}|:   |{foo:_^14.3f}|
"""

print(spam)
```

```
|{foo:14f}| :     |     42.500000|
|{foo:<14f}|:     |42.500000     |
|{foo:14.3f}|:    |        42.500|
|{foo:<14.3f}|:   |42.500        |
|{foo:^14.3f}|:   |    42.500    |
|{foo:<+14.3f}|:  |+42.500       |
|{foo:< 14.3f}|:  | 42.500       |
|{foo:<014.3f}|:  |42.50000000000|
|{foo:014.3f}|:   |0000000042.500|
|{foo:_<14.3f}|:  |42.500________|
|{foo:_>14.3f}|:  |________42.500|
|{foo:_^14.3f}|:  |____42.500____|
```

```python
# Использование индексации
# можно использовать отрицательные индексы
foo = [10.1, 20.2, 40.3, 60.4]

bar = f"foo[0] = {foo[0]:.3f}; foo[-1] = {foo[-1]:.3f}"

print("bar:", bar)
```

```
bar: foo[0] = 10.100; foo[-1] = 60.400
```

```python
# Доступ к элементам словаря
foo = {"key1": 20.5, "key2": 42, "key3": None}

bar = f"""foo['key1'] = {foo['key1']}
foo['key2'] = {foo['key2']}
foo['key3'] = {foo['key3']}"""

print(f"bar: {bar}")
```

```
bar: foo['key1'] = 20.5
foo['key2'] = 42
foo['key3'] = None
```

```python
# Использование выражений в f-строках

freq = 10.5e9
c = 3e8

foo = f"Частота: {freq / 1e9:.3f} ГГц. Длина волны: {c / freq * 1e3:.3f} мм"
print(foo)
```

```
Частота: 10.500 ГГц. Длина волны: 28.571 мм
```

```python
# Вызов функций в f-строках

from math import log10
gain = 10e3

foo = f"Коэффициент усиления: {gain:.2e} или {10 * log10(gain):.3f} дБ"
print(foo)
```

```
Коэффициент усиления: 1.00e+04 или 40.000 дБ
```

```python
# Использование методов класса
hello = "Hello"

foo = f"hello.lower(): {hello.lower()}\nhello.upper(): {hello.upper()}"
print(foo)
```

```
hello.lower(): hello
hello.upper(): HELLO
```

```python
# Использование методов класса
foo = ["hello", "world", "spam"]

bar = f"foo: {', '.join(foo)}"
print(bar)
```

```
foo: hello, world, spam
```

```python
# Использование методов класса
foo = [10, 20, 30]
foo_str = [str(item) for item in foo]

bar = f"foo: {', '.join(foo_str)}"
print(bar)
```

```
foo: 10, 20, 30
```

```
foo = [10, 20, 30]

bar = f"foo: {', '.join([str(item) for item in foo])}"
print(bar)
```

```
foo: 10, 20, 30
```

```python
# Использование режима отладки
foo = 10
bar = "hello"
baz = 20.5

eggs = f"{foo}\n{bar}\n{baz:10.3f}"
spam = f"{foo=}\n{bar=}\n{baz=:10.3f}"

print(eggs)
print()
print(spam)
```

```
10
hello
    20.500

foo=10
bar='hello'
baz=    20.500
```

```python
from numpy import linspace
from numpy.random import rand

freq = linspace(10e9, 12e9, 11)
col1 = rand(len(freq)) * 2 - 1
col2 = rand(len(freq)) * 2 - 1

for f, c1, c2 in zip(freq, col1, col2):
    print(f"{f:<15.3e}{c1:< 12.3f}{c2:< 12.3f}")
```

```
1.000e+10      -0.445      -0.671
1.020e+10      -0.774      -0.909
1.040e+10       0.160       0.971
1.060e+10      -0.646       0.558
1.080e+10       0.131      -0.070
1.100e+10       0.176       0.415
1.120e+10       0.168      -0.433
1.140e+10       0.648      -0.940
1.160e+10       0.032      -0.819
1.180e+10       0.331      -0.635
1.200e+10      -0.116      -0.617
```

```
from numpy import linspace
from numpy.random import rand

width = 17
prec = 4

freq = linspace(10e9, 12e9, 11)
col1 = rand(len(freq)) * 2 - 1
col2 = rand(len(freq)) * 2 - 1

for f, c1, c2 in zip(freq, col1, col2):
    print(f"{f:<{width + 3}.{prec}e}{c1:< {width}.{prec}f}{c2: .{prec}f}")
```

```
1.0000e+10          0.3853          0.1419
1.0200e+10         -0.1974         -0.9864
1.0400e+10         -0.3081          0.9513
1.0600e+10          0.3721         -0.7087
1.0800e+10         -0.4048         -0.0484
1.1000e+10          0.8782         -0.4207
1.1200e+10         -0.4168          0.5628
1.1400e+10         -0.5999         -0.5726
1.1600e+10         -0.3414          0.6707
1.1800e+10          0.8510          0.3094
1.2000e+10         -0.3867         -0.6827
```