



Московский авиационный институт
(национальный исследовательский университет)

1

Основы языка программирования Python

Что понадобится

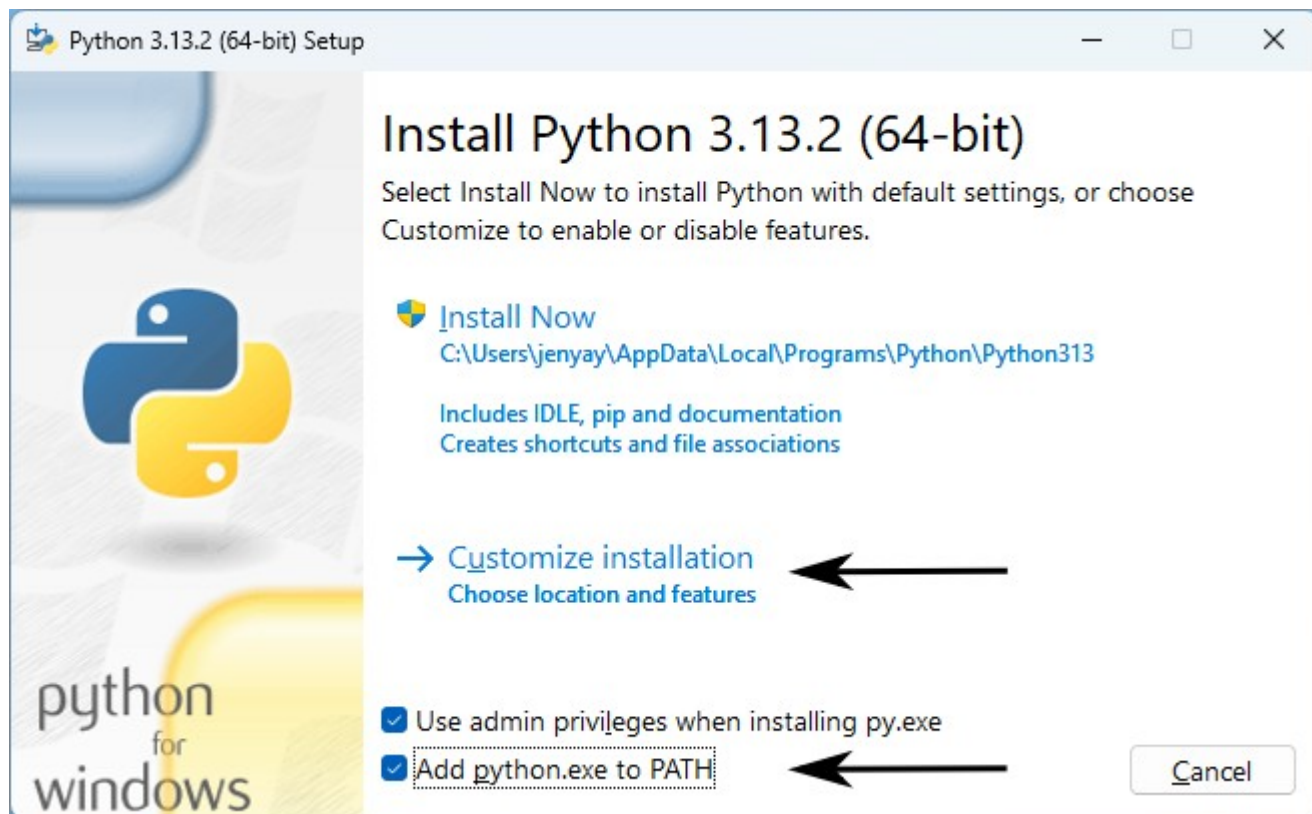
- Текстовый редактор.
- Умение работать в консоли.

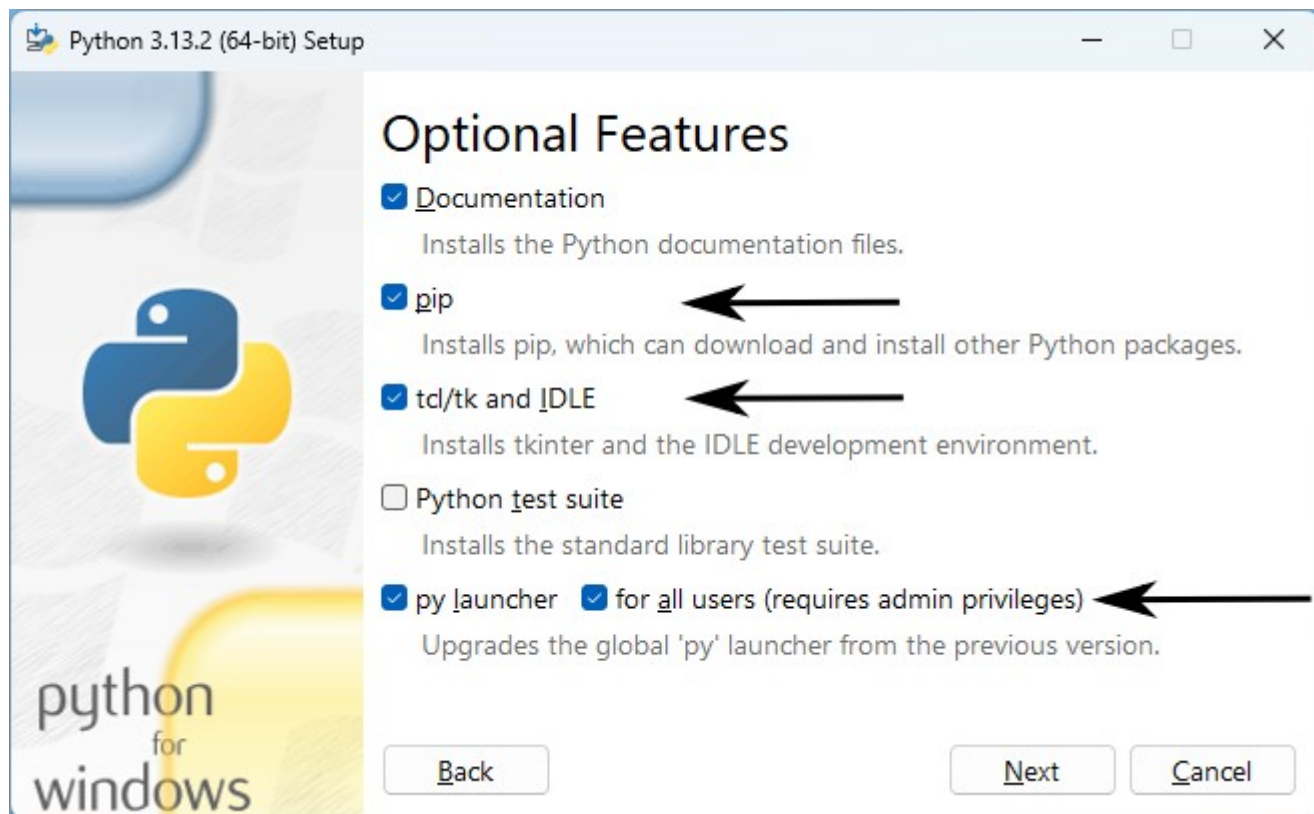
Примеры текстовых редакторов

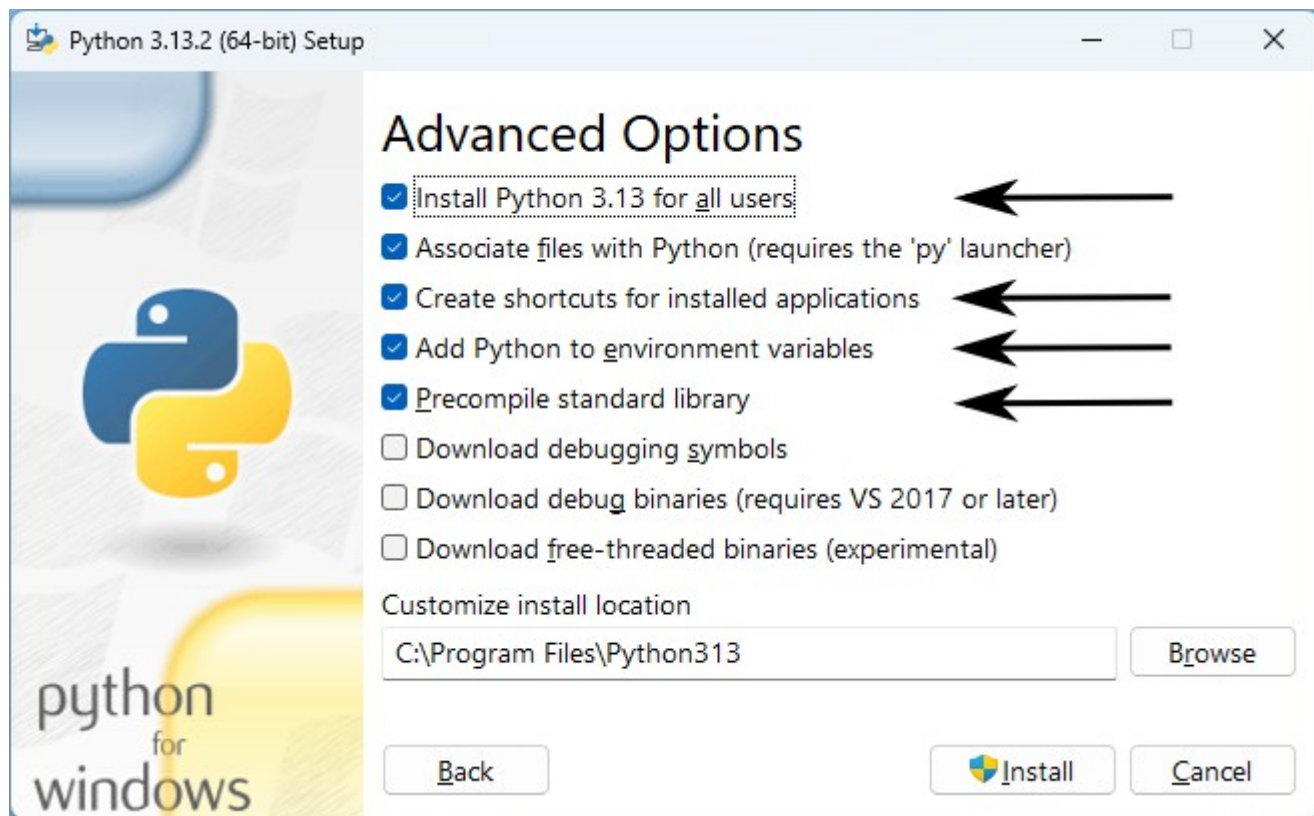
- IDLE.
- SciTE.
- Notepad++.
- Geany.
- Sublime Text
- VSCode.
- Atom.
- Spyder IDE.
- Ninja-IDE
- Vim.
- Emacs.

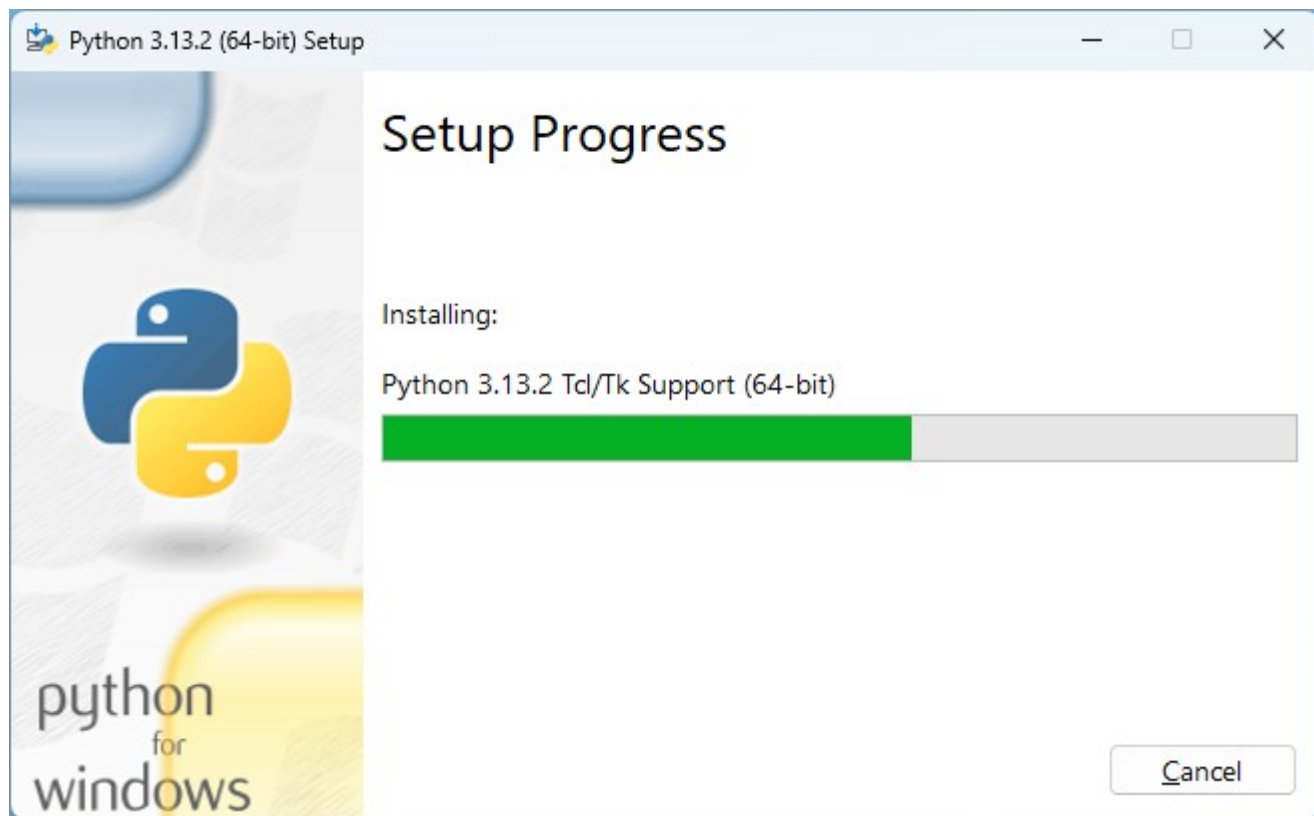
и множество других...

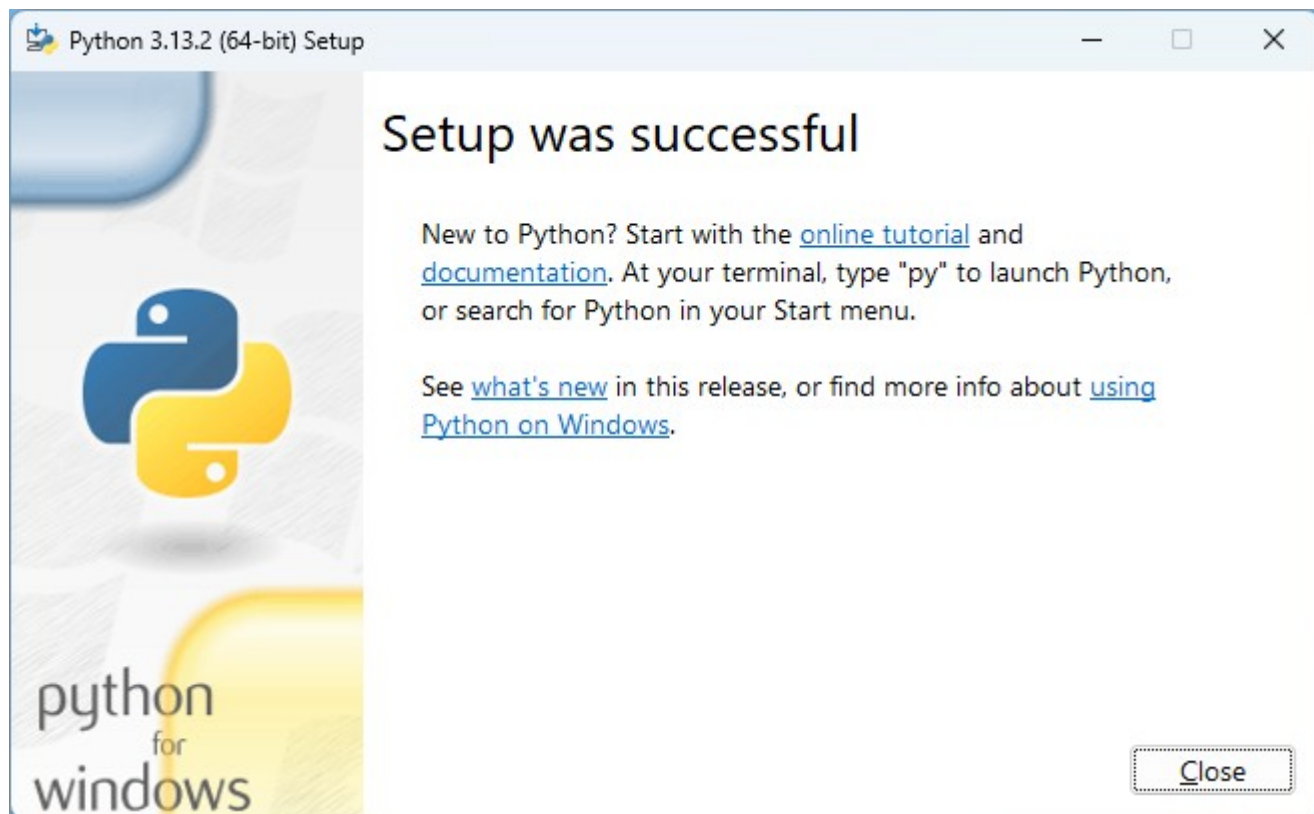
Установка Python



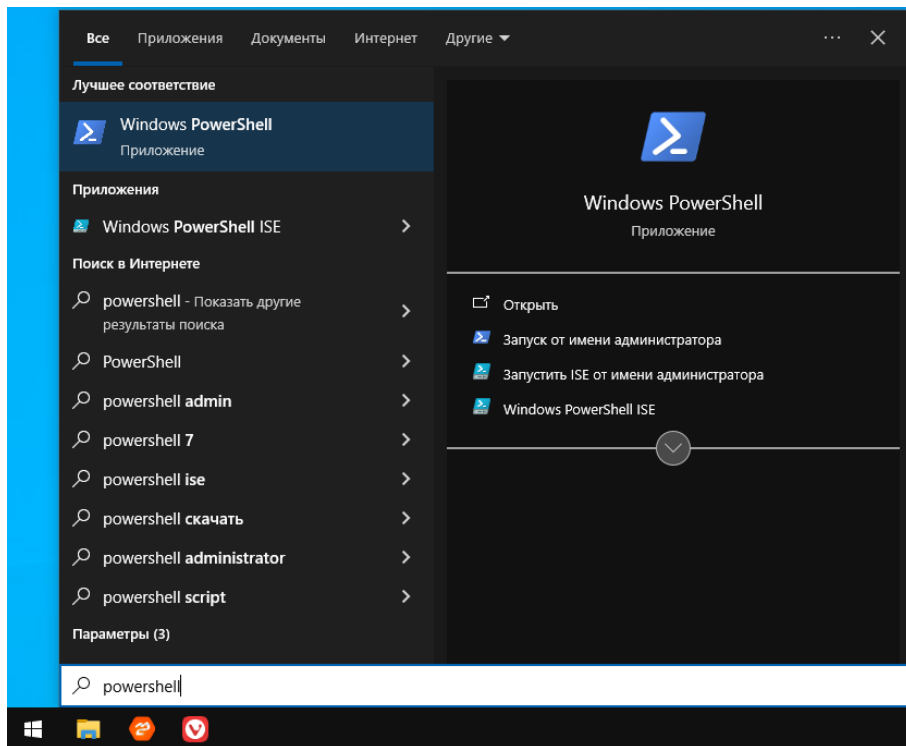
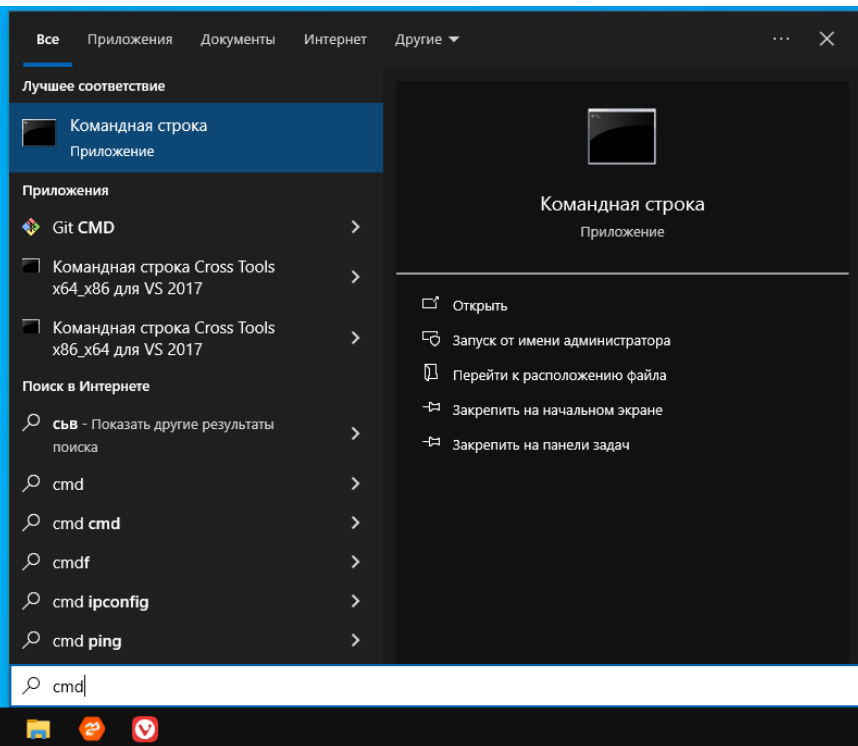






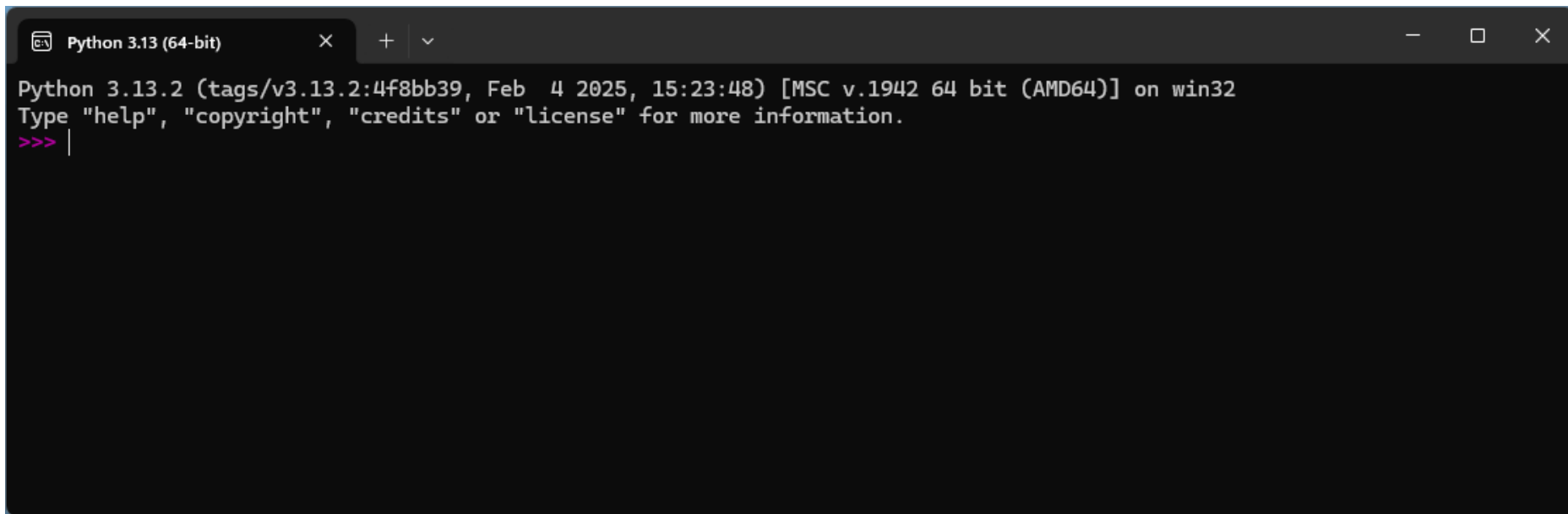


Запуск командной строки



Запуск Python в командном режиме

> python



```
Python 3.13 (64-bit) x + v
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

Выполнение скрипта, написанного на Python

12



The image shows a screenshot of a code editor window titled "hello.py - SciTE". The window has a menu bar with options: File, Edit, Search, View, Tools, Options, Language, Buffers, and Help. Below the menu bar, there is a tab labeled "1 hello.py". The main editing area contains a single line of Python code: `print("Hello, world!")`. The code is written in a monospaced font, with the string "Hello, world!" in purple and the rest in black. The editor has a light gray background and a vertical scrollbar on the right side.

> python hello.py

Первое использование переменных

```
text = "Hello, world!"  
print(text)
```

Создание переменной	Тип	Комментарий
foo = 10	int	Целочисленный тип
foo = 20.5 bar = .7 spam = 3e8	float	Число с плавающей точкой
foo = 3j bar = 2 + 5j	complex	Комплексное число
foo = "Текст" bar = 'Текст' spam = """Многострочный текст"""	str	Строка
foo = []	list	Пустой список
foo = {}	dict	Пустой словарь
foo = True bar = False	bool	Булево значение
foo = None	NoneType	Специальное значение для обозначения не инициализированной переменной

Функция `type()`

Функция `type()` предназначена для получения типа переменной

```
text = "Hello, world!"  
print(type(text))
```

Результат выполнения:

```
<class 'str'>
```

Встроенные (built-in) функции Python

<https://docs.python.org/3/library/functions.html>

Built-in Functions

A abs() aiter() all() anext() any() ascii()	E enumerate() eval() exec()	L len() list() locals()	R range() repr() reversed() round()
B bin() bool() breakpoint() bytearray() bytes()	F filter() float() format() frozenset()	M map() max() memoryview() min()	S set() setattr() slice() sorted() staticmethod() str() sum() super()
C callable() chr() classmethod() compile() complex()	G getattr() globals()	N next()	T tuple() type()
D delattr() dict() dir() divmod()	H hasattr() hash() help() hex()	O object() oct() open() ord()	V vars()
	I id() input() int() isinstance() issubclass() iter()	P pow() print() property()	Z zip()
			_ __import__()

Арифметические выражения

```
a = 2
```

```
b = 3.3
```

```
c = 10
```

```
d = a + (b ** 2)
```

```
e = c / a
```

```
print(d, e)
```

Результат выполнения:

12.889999999999999 5.0

Основные арифметические операторы

18

+	Сложение
-	Вычитание
*	Умножение
/	Деление
//	Целочисленное деление
%	Остаток от деления
**	Возведение в степень

Сокращенное арифметическое выражение	Полное выражение
foo += bar	foo = foo + bar
foo -= bar	foo = foo - bar
foo *= bar	foo = foo * bar
foo /= bar	foo = foo / bar
foo // bar	foo = foo // bar
foo %= bar	foo = foo % bar
foo **= bar	foo = foo ** bar

Сокращенные арифметические выражения

20

```
e = 10  
print(e)
```

```
e += 2  
print(e)
```

```
e *= 3  
print(e)
```

```
e /= 2  
print(e)
```

Результат выполнения:

10

12

36

18.0

Динамическая типизация

```
x = 10  
print(type(x))
```

```
x = 1.5  
print(type(x))
```

```
x = "Строка"  
print(type(x))
```

Результат выполнения:

```
<class 'int'>  
<class 'float'>  
<class 'str'>
```

Комментарии

Комментарий

text = "Hello, world!"

print(text)

Еще один комментарий

Скрипты с указанием кодировки

```
# coding: utf-8  
text = "Hello, world!"  
print(text)
```

Способы указания кодировки

```
# coding: utf-8  
# -*- coding: utf-8 -*-  
# coding=utf-8  
...
```


Кодировки

Таблица ASCII (American Standard Code for Information Interchange)

26

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	

Таблица Unicode (фрагмент)

Graphic character symbol			Hexadecimal character value		
0020	0	@ 0040	P 0050	` 0060	p 0070
00A0	° 00B0	À 00C0	Đ 00D0	à 00E0	đ 00F0
! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
i 00A1	± 00B1	Á 00C1	Ñ 00D1	á 00E1	ñ 00F1
" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
ç 00A2	² 00B2	Â 00C2	Ò 00D2	â 00E2	ò 00F2
# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
£ 00A3	³ 00B3	Ã 00C3	Ó 00D3	ã 00E3	ó 00F3
\$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
¤ 00A4	´ 00B4	Ä 00C4	Ô 00D4	ä 00E4	ô 00F4
% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
¥ 00A5	µ 00B5	Å 00C5	Õ 00D5	å 00E5	ö 00F5
& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
ı 00A6	¶ 00B6	Æ 00C6	Ö 00D6	æ 00E6	ö 00F6
' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
§ 00A7	· 00B7	Ç 00C7	× 00D7	ç 00E7	÷ 00F7
(0028	8 0038	H 0048	X 0058	h 0068	x 0078
¨ 00A8	¸ 00B8	È 00C8	Ø 00D8	è 00E8	ø 00F8
) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
© 00A9	¹ 00B9	É 00C9	Ù 00D9	é 00E9	ù 00F9
* 002A	:	J 004A	Z 005A	j 006A	z 007A
ª 00AA	º 00BA	Ê 00CA	Ú 00DA	ê 00EA	ú 00FA
+ 002B	;	K 004B	[005B	k 006B	{ 007B
« 00AB	» 00BB	Ë 00CB	Û 00DB	ë 00EB	û 00FB
, 002C	< 003C	L 004C	\ 005C	l 006C	007C
¬ 00AC	¼ 00BC	Ì 00CC	Ü 00DC	ì 00EC	ü 00FC
- 002D	= 003D	M 004D] 005D	m 006D	} 007D
- 00AD	½ 00BD	Í 00CD	Ý 00DD	í 00ED	ý 00FD
. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
® 00AE	¾ 00BE	Î 00CE	Þ 00DE	î 00EE	þ 00FE
/ 002F	? 003F	O 004F	_ 005F	o 006F	007F
00AF	ı 00BF	İ 00CF	ß 00DF	ï 00EF	ÿ 00FF

Таблица Unicode (фрагмент)

U+1F600 : 😄	U+263A : 😊	U+1F61F : 😏	U+1F61E : 😞
U+1F603 : 😃	U+1F61A : 😜	U+1F641 : 😞	U+1F648 : 🙈
U+1F604 : 😄	U+1F619 : 😜	U+2639 : 😞	U+1F649 : 🙈
U+1F601 : 😄	U+1F60B : 😊	U+1F62E : 😲	U+1F64A : 🙈
U+1F606 : 😏	U+1F61B : 😜	U+1F62F : 😲	U+1F44B : 🙌
U+1F605 : 😄	U+1F61C : 😜	U+1F632 : 😲	U+1F91A : 🙌
U+1F923 : 😄	U+1F92A : 😜	U+1F633 : 😲	U+1F590 : 🙌
U+1F602 : 😄	U+1F61D : 😜	U+1F626 : 😲	U+270B : 🙌
U+1F642 : 😊	U+1F911 : 🙌	U+1F627 : 😲	U+1F596 : 🙌
U+1F643 : 😊	U+1F917 : 🙌	U+1F628 : 🙌	U+1F44C : 🙌
U+1F609 : 😊	U+1F92D : 😊	U+1F630 : 🙌	U+270C : 🙌
U+1F60A : 😊	U+1F92B : 😊	U+1F625 : 😲	U+1F91E : 🙌
U+1F607 : 😊	U+1F914 : 🙌	U+1F622 : 🙌	U+1F91F : 🙌
U+1F60D : 😊	U+1F60E : 😊	U+1F62D : 🙌	U+1F918 : 🙌
U+1F929 : 🙌	U+1F913 : 🙌	U+1F631 : 🙌	U+1F919 : 🙌
U+1F618 : 🙌	U+1F9D0 : 🙌	U+1F616 : 🙌	U+1F44D : 🙌
U+1F617 : 🙌	U+1F615 : 🙌	U+1F623 : 🙌	U+1F44E : 🙌

Кодировка	Размер одного символа, байт
UTF-8	1 - 4
UTF-16	2 - 4
UTF-32	4

Примеры кодировки UTF-8

Hello, world!

```
00000000: 48 65 6C 6C 6F 2C 20 77|6F 72 6C 64 21 0A
```

Привет, мир!

```
00000000: D0 9F D1 80 D0 B8 D0 B2|D0 B5 D1 82 2C 20 D0 BC  
00000010: D0 B8 D1 80 21 0A
```

Оператор `if` и булевы операторы

Синтаксис оператора if

if условие 1:

 блок кода

elif условие 2:

 блок кода

...

elif условие N:

 блок кода

else:

 блок кода

Пример использования оператора if

```
x = 10
if x < 0:
    print('x < 0')
    pass
elif x > 0:
    print('x > 0')
    pass
else:
    print('x == 0')
```

Оператор `if ... else` для присваивания значений

34

```
if условие:  
    foo = значение_1  
else:  
    foo = значение_2
```

||

```
foo = значение_1 if условие else значение_2
```

x = 15

```
if x % 3 == 0:  
    y = x // 3  
else:  
    y = x * 3
```

Эквивалентно предыдущему коду

```
y = (x // 3) if (x % 3 == 0) else (x * 3)
```

Или аналогично

```
y = x // 3 if x % 3 == 0 else x * 3
```

Логические операторы

==	Равно
!=	Не равно
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
not	Инверсия
or	Дизъюнкция
and	Конъюнкция

```
foo = 10  
bar = foo > 5  
  
print(bar)  
print(type(bar))
```

```
True  
<class 'bool'>
```

```
foo = 10  
bar = foo > 5  
spam = bar and foo < 8  
  
print(spam)
```

Функция `print()`

<https://docs.python.org/3/library/functions.html#print>

```
print(*objects, sep=' ', end='\n', file=None, flush=False)
```

Print *objects* to the text stream *file*, separated by *sep* and followed by *end*. *sep*, *end*, *file*, and *flush*, if present, must be given as keyword arguments.

All non-keyword arguments are converted to strings like `str()` does and written to the stream, separated by *sep* and followed by *end*. Both *sep* and *end* must be strings; they can also be `None`, which means to use the default values. If no *objects* are given, `print()` will just write *end*.

The *file* argument must be an object with a `write(string)` method; if it is not present or `None`, `sys.stdout` will be used. Since printed arguments are converted to text strings, `print()` cannot be used with binary mode file objects. For these, use `file.write(...)` instead.

Output buffering is usually determined by *file*. However, if *flush* is true, the stream is forcibly flushed.

Changed in version 3.3: Added the *flush* keyword argument.