

## **Тема 1.**

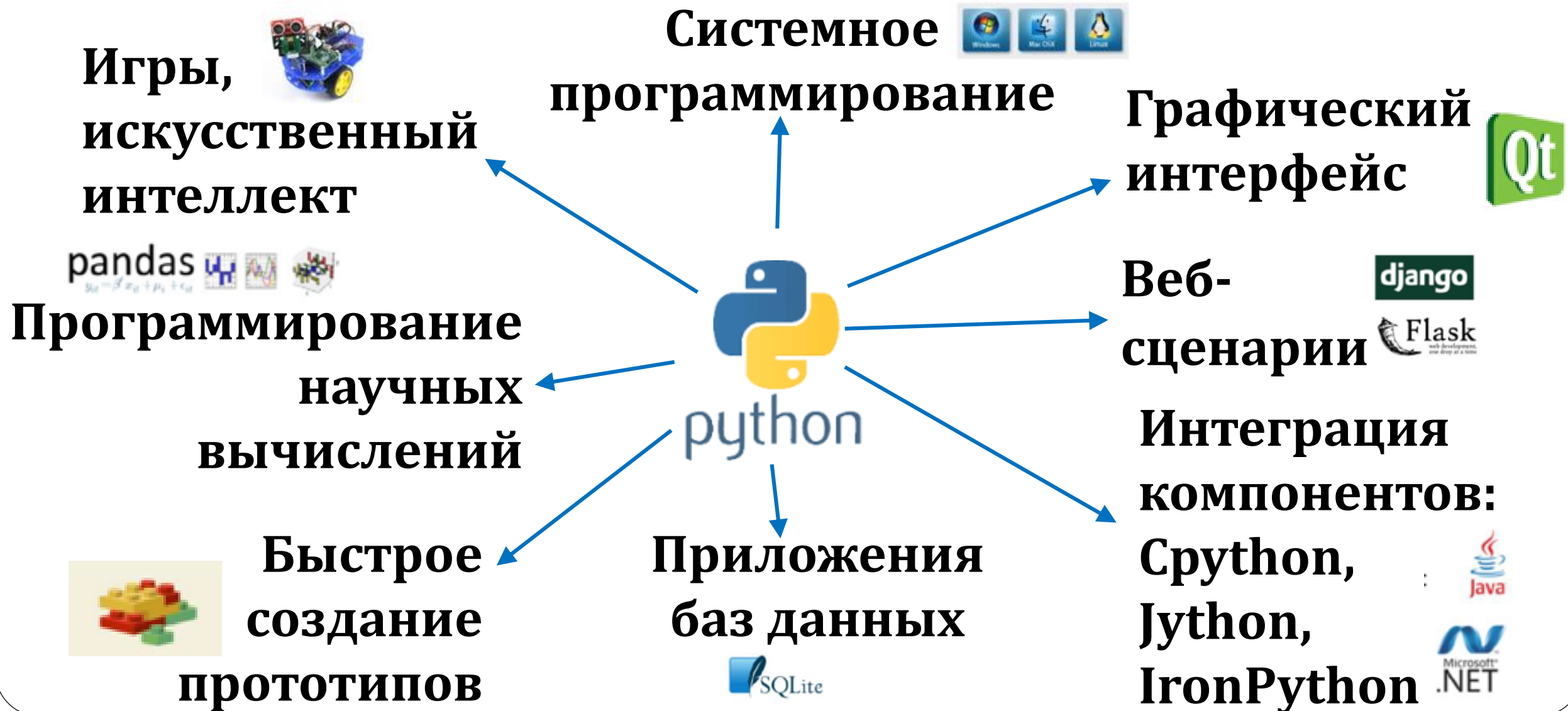
# **Введение в язык программирования Python. Алгоритмические средства языка Python**

# Введение в язык программирования Python

---

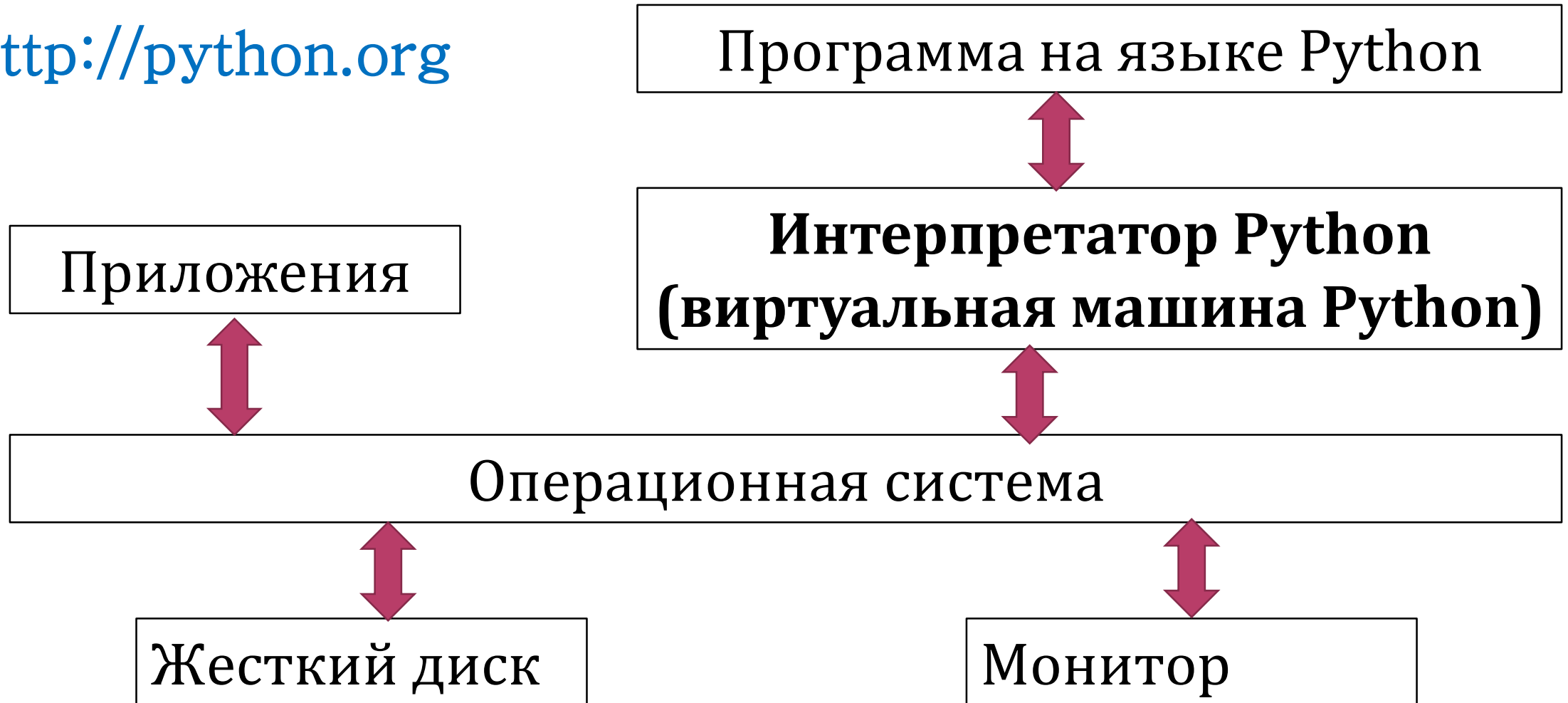
- Python – высокоуровневый язык программирования общего назначения (полностью объектно-ориентированный)
- Возможности:
  - динамическая типизация,
  - автоматическое управление памятью
  - поддержка многопоточных вычислений
  - удобные структуры данных
  - и т.д.

# Область применения языка программирования Python



# Схема запуска программ на Python

<http://python.org>



# Введение в язык программирования Python

---

- **Поддерживается всеми операционными системами**
- **Программы могут разрабатываться в консольном режиме (расширение .py) и с графическим интерфейсом (расширение .pyw).**
- **Программа на языке Python – это обычный текстовый файл (инструкции выполняются интерпретатором)**

# The Zen of Python (PEP20 от Тима Петерсона)

---

- Свод правил по улучшению языка Python (1999)
- **import this**
  - Красивое лучше уродливого.
  - Явное лучше неявного.
  - Простое лучше сложного.
  - Сложное лучше запутанного.
  - Развернутое лучше вложенного.
  - Разреженное лучше плотного.

# The Zen of Python (PEP20 от Тима Петерсона)

---

- Свод правил по улучшению языка Python (1999)
- **import this**
  - Читаемость имеет значение.
  - Особые случаи не настолько особые, чтобы нарушать правила.
  - При этом практичность важнее безупречности.
  - Ошибки не должны замалчиваться.
  - Если не замалчиваются явно.
  - Встретив двусмысленность, отбрось искушение угадать.

# The Zen of Python (PEP20 от Тима Петерсона)

---

- Свод правил по улучшению языка Python (1999)
- **import this**
  - Должен существовать один - и, желательно, только один – очевидный способ сделать что-то.
  - Хотя этот способ поначалу может быть и не очевиден, если вы не голландец.
  - Сейчас лучше, чем никогда.
  - Хотя никогда часто лучше, чем \*прямо\* сейчас.



# The Zen of Python (PEP20 от Тима Петерсона)

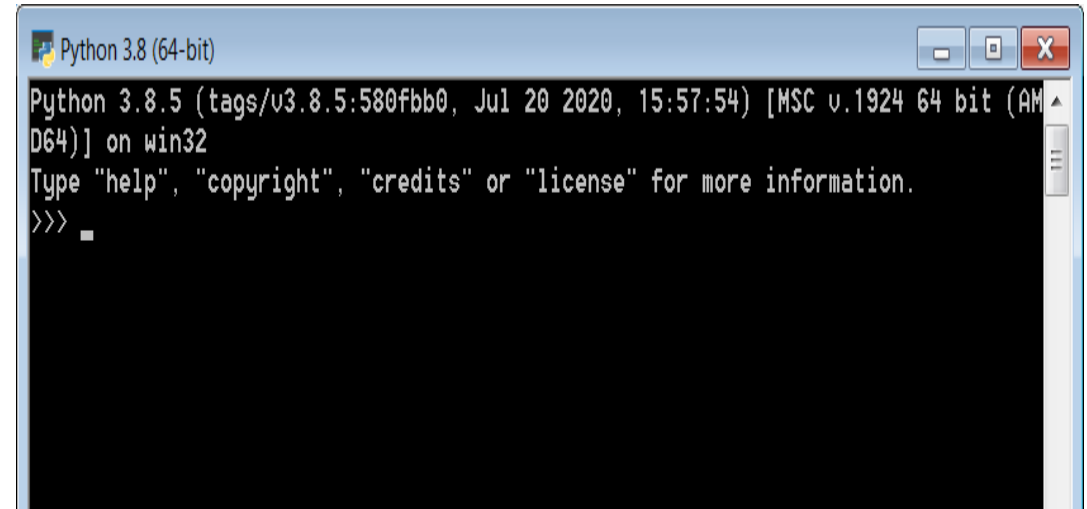
---

- Свод правил по улучшению языка Python (1999)
- **import this**
  - Если реализацию сложно объяснить – идея точно плоха.
  - Если реализацию легко объяснить – возможно, идея хороша.
  - Пространства имен – отличная штука! Будем использовать их чаще!

# Язык программирования Python

---

- **Среда бесплатная**
- **Версия 3.6 и выше**
- **Для работы доступна консольная версия и активная графическая среда IDLE**
- **IDE PyCharm Community Edition от JetBrains**



```
Python 3.8 (64-bit)
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

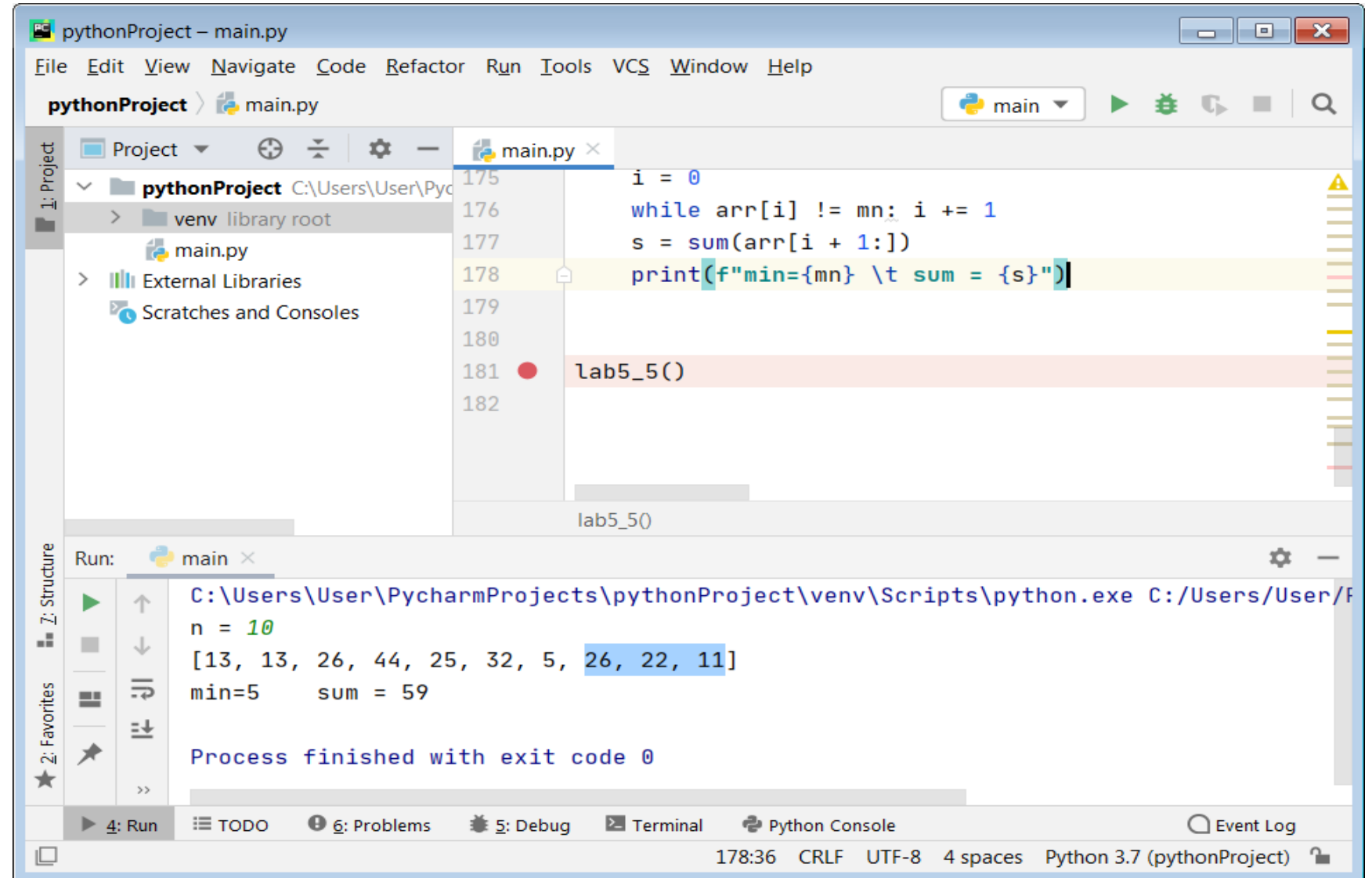


```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> |
```

**IDLE** – Integrated Development Environment

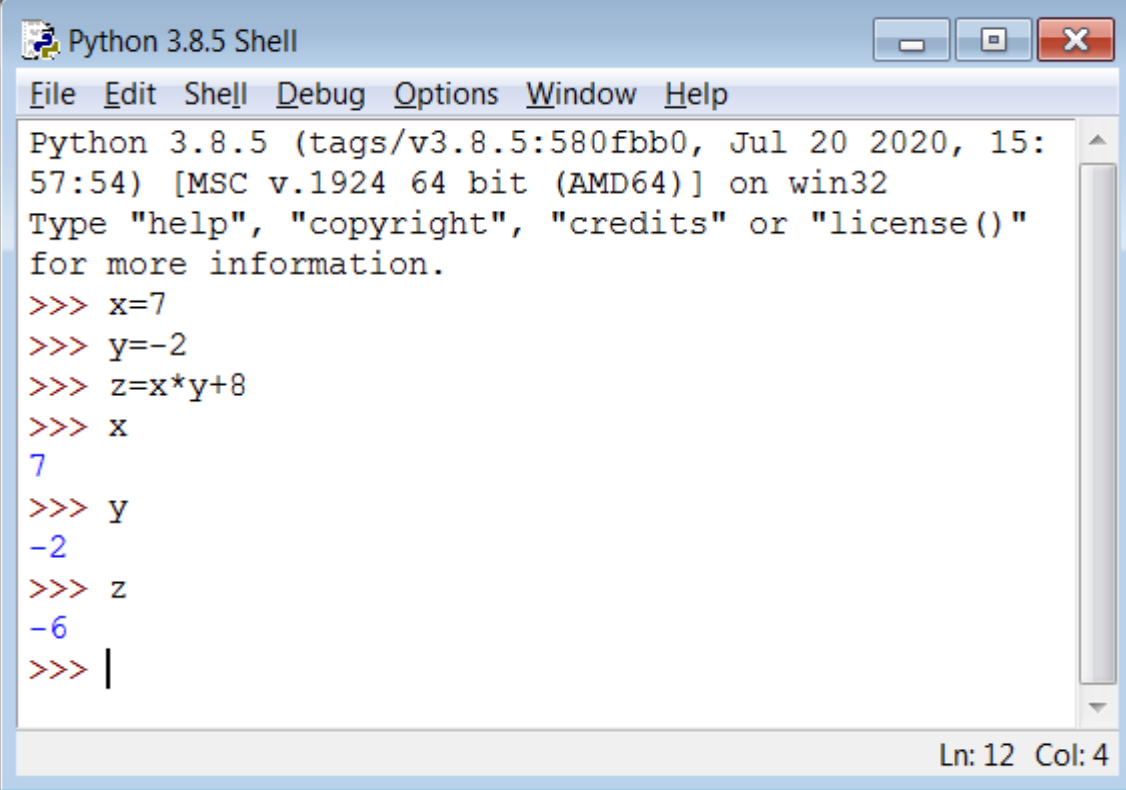
# Язык программирования Python

IDE  
PyCharm  
Community  
Edition от  
JetBrains



# Язык программирования Python

- Возможны два режима работы:
  - *Интерактивный* – команды выполняются сразу после их вызова, результат выполнения сразу выводится на экран Сохранить инструкции в файле нельзя!

A screenshot of the Python 3.8.5 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following content:

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>> x=7
>>> y=-2
>>> z=x*y+8
>>> x
7
>>> y
-2
>>> z
-6
>>> |
```

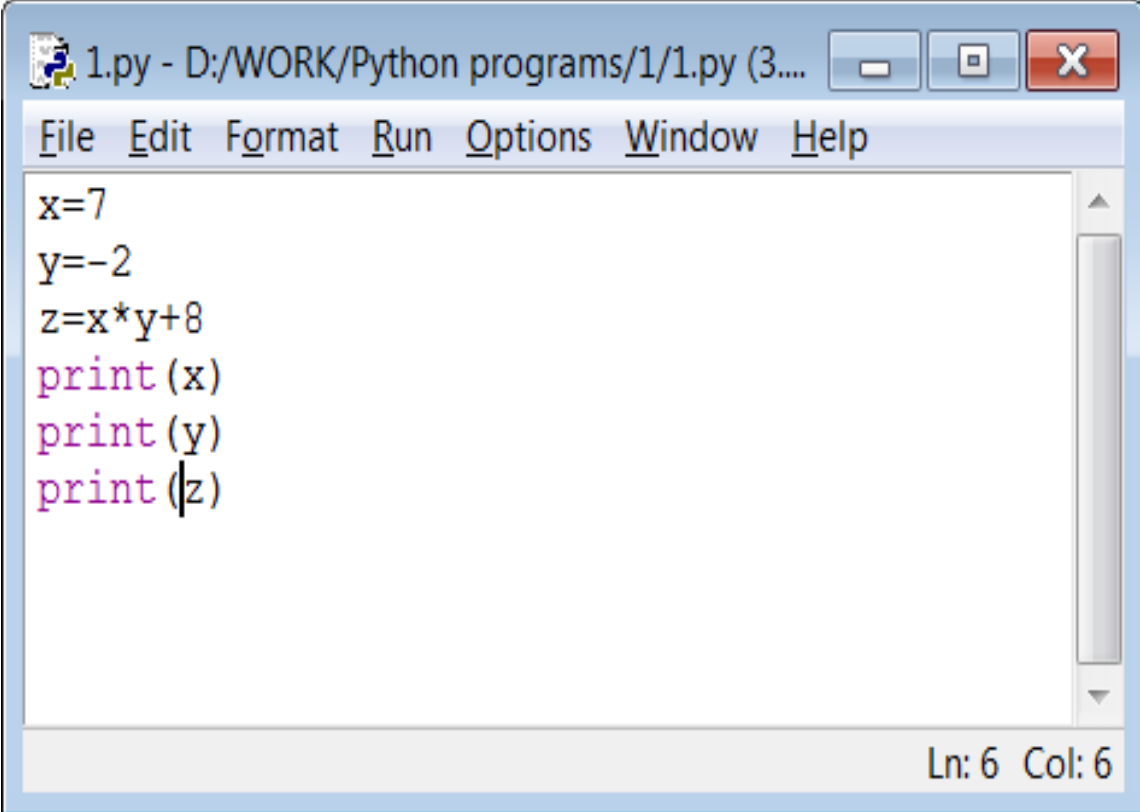
The status bar at the bottom right indicates 'Ln: 12 Col: 4'.

*(подходит для простых программ и тестирования фрагментов кода)*

# Язык программирования Python

---

- Возможны два режима работы:
  - *Программный* – записывается вся программа (также называется «сценарий», «скрипт»), а после выполняется.



The screenshot shows a window titled "1.py - D:/WORK/Python programs/1/1.py (3...." with a menu bar (File, Edit, Format, Run, Options, Window, Help) and a text area containing the following Python code:

```
x=7
y=-2
z=x*y+8
print(x)
print(y)
print(z)
```

The status bar at the bottom right indicates "Ln: 6 Col: 6".

# Программный режим

---

- Для работы в программном режиме требуется создать в IDLE новый файл.
- Файлы с текстами программ на Python имеют расширение **\*.py** и называются **модулями**.
- Запуск программ из среды – F5 (или пункт меню Run / Run Module)
- Результат выполнения программы выводится в интерактивном режиме.

# Интерактивный режим

---

# Интерактивный режим

---

- В интерактивном режиме интерпретатор выполняет инструкции и сразу выводит результат.
- Можно использовать для получения мгновенных расчетов, проведения экспериментов, тестирования программ «на лету»
- Удобно выполнять эксперименты над небольшими фрагментами кода, которые затем могут быть перенесены в скрипты



# Интерактивный режим

---

- *Принцип работы:*

- Однострочные команды:

>>> Команда + Enter

- Многострочные команды:

>>> Команда # строка 1

Команда # строка 2

...

Команда # строка N

Enter

Enter

>>>

# Интерактивный режим

---

- *Пример ввода многострочной инструкции:*

```
>>> x=3
>>> y=8
>>> if x>0: # многострочная инструкция
           y=5 #здесь нужно 2 раза нажать Enter
```

```
>>>
```

# Интерактивный режим

---

- ***Недостатки:***
  - Программный код не сохраняется, соответственно для повторного запуска необходимо его снова ввести

# Интеллектуальный калькулятор

---

- Среду python можно использовать в качестве интеллектуального калькулятора.
- *Принцип работы:* вводится математическое выражение и нажимается Enter

```
>>> 2+2
4
>>> (50 - 5*6) / 4
5.0
>>> 7/3
2.3333333333333335
>>> 7// -3
-3
>>> 7/-3
-2.3333333333333335
```

# Интеллектуальный калькулятор

---

- Операнды в математических выражениях – целые или вещественные числа.
- Операторы, доступные над числами:
  - $+$ ,  $-$ ,  $*$ ,  $/$  - сложение, вычитание, умножение, деление (обычное, результат вещественный)
  - $//$  - деление с округлением вниз (целочисленное)
  - $**$  - возведение в степень
  - $\%$  - остаток от деления

# Интеллектуальный калькулятор

---

- Приоритеты операций соответствуют математическим соглашениям.
- Для изменения приоритета – скобки

>>>  $-2^{*}4$

-16

>>>  $-(2^{*}4)$

-16

>>>  $(-2)^{*}4$

16

# Интеллектуальный калькулятор

---

## Переменные

- **Именованное:** нельзя начинать с цифры и использовать ключевые слова языка Python

<code>and</code>	<code>as</code>	<code>assert</code>	<code>break</code>	<code>class</code>	<code>continue</code>
<code>def</code>	<code>del</code>	<code>elif</code>	<code>else</code>	<code>except</code>	<code>exec</code>
<code>finally</code>	<code>for</code>	<code>from</code>	<code>global</code>	<code>if</code>	<code>import</code>
<code>in</code>	<code>is</code>	<code>lambda</code>	<code>nonlocal</code>	<code>not</code>	<code>or</code>
<code>pass</code>	<code>raise</code>	<code>return</code>	<code>try</code>	<code>while</code>	<code>with</code>
<code>yield</code>	<code>True</code>	<code>False</code>	<code>None</code>	<code>—</code>	<code>—</code>

# Интеллектуальный калькулятор

---

## Переменные

- Для задания переменной в калькуляторе необходимо присвоить ей значение (знак = )
- Вывод на экран:
  - по имени переменной(ых)
  - метод `print()`

```
>>> width=20
>>> height=5*9
>>> width*height
900
```

```
>>> width
20
>>> width, height
(20, 45)
>>> print(height)
45
```



# Интеллектуальный калькулятор

---

## Переменные

- Для задания переменной в калькуляторе необходимо присвоить ей значение (знак = )

```
>>> width=20  
>>> height=5*9  
>>> width*height  
900
```

- Значение можно присвоить одновременно нескольким переменным:

```
>>> x=y=z=0 # Переменным x, y, z присваивается 0  
>>> a,b,c=20,14.6,19 #Присваивание a=20, b=14.6, c=19
```

# Интеллектуальный калькулятор

---

## Переменная \_

- В интерактивном режиме в переменной `_` сохраняется последнее выведенное значение.
- К этому значению можно обращаться в следующих инструкциях.

```
>>> tax=17.5/100
>>> price=3.50
>>> price*tax
0.6124999999999999
>>> price + _
4.1125
>>> print(round(_, 2))
4.11
>>>
```

# Интеллектуальный калькулятор

---

## Работа с комплексными числами

- Мнимая часть записывается с суффиксом 'j' или 'J'.
- Комплексные числа записываются как '(real+imagj)' или могут быть созданы функцией 'complex(real, imag)'

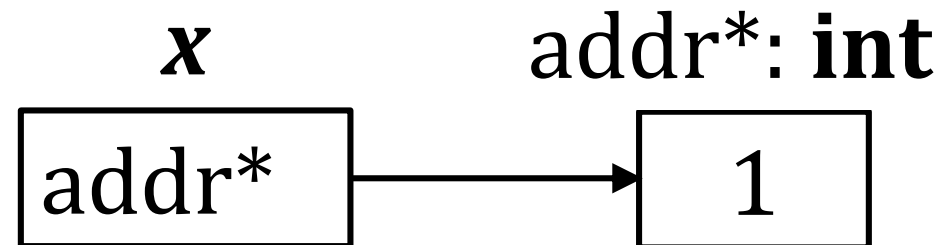
```
>>> 1j*1J
(-1+0j)
>>> 1j*complex(0,1)
(-1+0j)
>>> 3+1j*3
(3+3j)
>>> (3+1j)*3
(9+3j)
>>> (1+2j)/(1+1j)
(1.5+0.5j)
>>> a=1.5+0.5j
>>> a.real
1.5
>>> a.imag
0.5
```

# Интеллектуальный калькулятор

---

- Язык полностью объектно-ориентированный.
  - Все данные языка, в том числе простые типы данных (числа, строки) являются *объектами*.
  - В переменной хранится не сам объект, а ссылка на него.

```
>>> x = 1  
>>>
```



# Интеллектуальный калькулятор

---

## Функции

- В интерактивном калькуляторе можно использовать и различные математические функции, например:
  - `abs (x)`
  - `pow (x, y)`
  - `round (number [, ndigits] )`
  - `int ()` - **возвращает целочисленный объект, построенный из числа или строки**
  - и др.

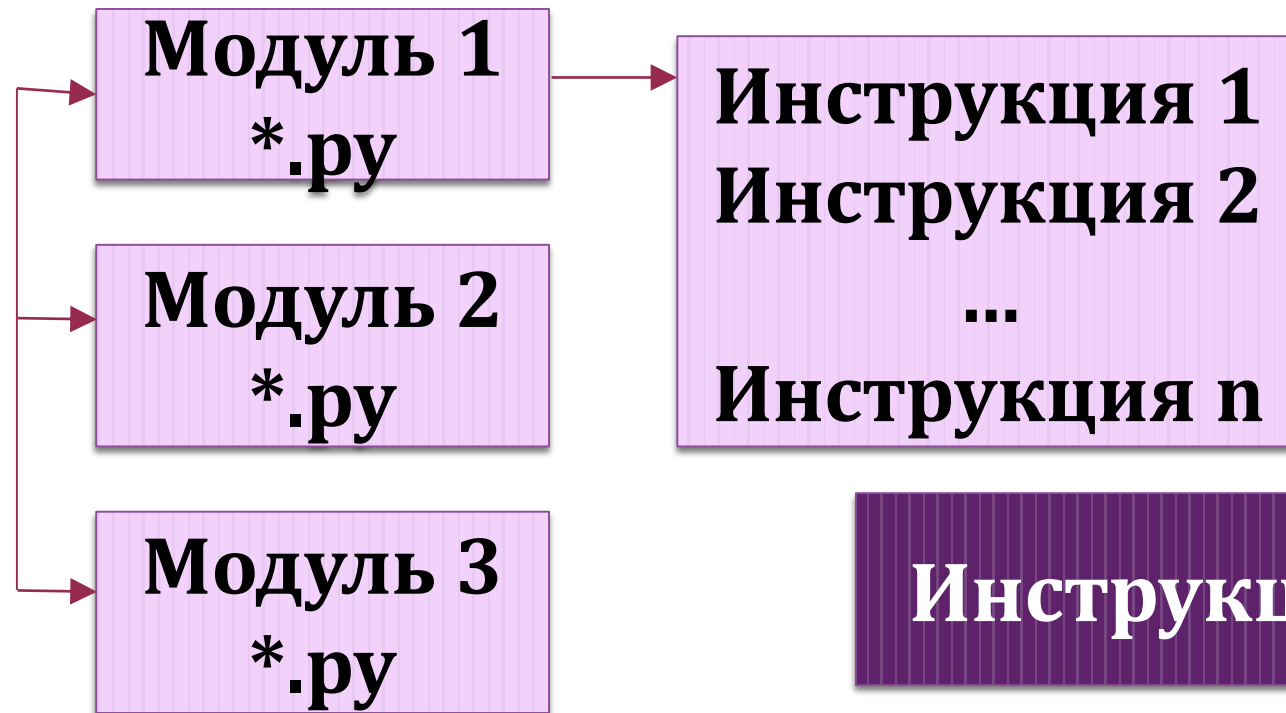
# Основные элементы языка Python

# Структура программы на языке Python

Структура проекта на языке Python состоит из отдельных модулей.

**Программа**

**Модуль** –  
это ряд связанных  
между собой  
операций



**Инструкции**

**Простые**

**Составные**

# Введение в язык программирования Python

---

- Программа на Python, с точки зрения интерпретатора, состоит из логических строк.
- Конец строки – конец инструкции (; не требуется)
- Логические строки можно разбить на несколько физических строк с помощью:
  - Обратной косой чертой «\»
  - Скобок (круглые, квадратные или фигурные)



# Пример

---

`print (a, " - очень длинная строка, которая не ", \`  
`" помещается в ", 80, " знакоместах")`

*или*

`if (a == 1 and b == 2 and`  
`c == 3 and d == 4):`  
 `print('spam' * 3)`

# Введение в язык программирования Python

---

- В Python **отступ** очень важен.
- Вложенные инструкции объединяются в блоки по величине отступов.
- Когда основная инструкция завершается двоеточием, вслед за которым располагается вложенный блок кода, Python использует отступ для указания блока кода.

# Введение в язык программирования Python

---

- **Отступы:** 4 пробела или символ табуляции

*Основная инструкция:*

*Вложенный блок инструкций*

# Введение в язык программирования Python

---

- **Комментарии**

- *Однострочные* – символ #

**# Это комментарий**

- *Многострочный* – тройные кавычки (двойные или одинарные) в начале и в конце строки

**""" Многострочный комментарий  
Многострочный комментарий """**

# Основные элементы языка Python

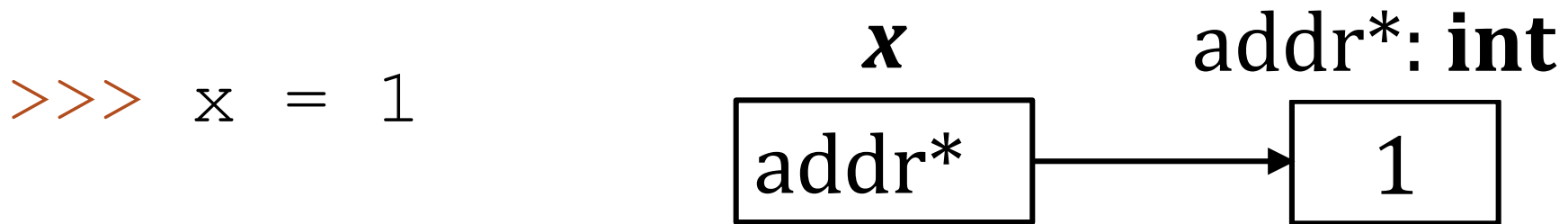
---

- **Лексемы языка:**
  - **Ключевые слова языка**
  - **Идентификаторы**
  - **Литералы (константы)**
  - **Операторы (операции)**
  - **Знаки препинания**

# Основные элементы языка Python

---

- Данные в языке представлены в форме объектов или связей между ними.
- Переменные хранят не сам объект, а ссылку на него (т.е. адрес объекта в памяти компьютера).



- Тип переменной определяется типом объекта, на который она ссылается.

# Литералы

---

- В языке Python литерал – это выражение (или константа), которое создает объект.
- Для каждого литерала в тексте программы создается отдельный объект некоторого типа.
- Примеры литералов:
  - `1234`
  - `2.71`
  - `1+2j`
  - `'Hello world!'`
  - `[1, 3, [2, 4, 6]]`
  - `{ 'Sun', 'Mon', 'Tue' }`
  - `set('jklm')`

# Литералы

---

- Язык Python содержит набор базовых объектных типов (встроенных в язык) :

Тип объекта	Литерал
Число	12, 2.855, 1=2j
Строка	'Sunday', "Monday"
Список	[1,2,[3,4[5]]],[13, 'Text']
Словарь	{'abc':1,'abcd':2,'abd':3}



# Литералы

---

- Язык Python содержит набор базовых объектных типов (встроенных в язык) :

Тип объекта	Литерал
Кортеж	(8, 'bestprog' )
Файл	filename=open('myfile.txt', 'r')
Множество	set('jklmn'), {'j','k','l','m','n'}
Другие	<b>None, True</b> , объекты шаблонов и прочее

# Операторы

---

- В языке Python существуют следующие типы операторов:
  - Арифметические операторы
  - Операторы сравнения
  - Операторы присваивания
  - Побитовые операторы
  - Логические операторы
  - Операторы членства
  - Операторы тождественности

# Операторы

---

- **Арифметические операторы:**

**+, −, \*, /** - сложение, вычитание, умножение, деление

**//** - целочисленное деление

**%** - остаток от деления

**\*\*** - возведение в степень (выполняется справа налево!)

*Пример,  $3 ** 2 ** 4$  тождественно  $3 ** (2 ** 4)$ ,  
т.е. выполняется по правилам математики:  $3^{2^4} = 3^{16}$*

# Операторы

---

- **Операторы сравнения:**  
`==, !=, >, <, >=, <=`
- **Операторы присваивания:**  
`=, +=, -=, *=, /=, %=, **=, //=`
- **Побитовые операторы:**  
`&, |, ^, ~, <<, >>`
- **Логические операторы:**  
`and, or, not`

# Операторы

---

- Операторы членства:

**in** (**not in**) – возвращает True, если элемент присутствует в последовательности, иначе – False

Примеры:

“cad” **in** “cadillac” - True

1 **in** [2, 3, 1, 6] - True

“hi” **in** {“hi”:2, “bye”:1} - True

2 **in** {“hi”:2, “bye”:1} - False

# Операторы

---

- Операторы тождественности:

**is (is not)** – возвращает True, если оба операнда указывают на один объект (т.е. если значения функции **id()** у этих объектов равны).

Примеры:

```
>>> x=3
```

```
>>> y=3
```

```
>>> x is y
```

```
True
```

# Операторы

---

- **Приоритеты операторов (по убыванию):**

Оператор
<b>**</b>
<b>~, +, - (унарные)</b>
<b>*, /, %, //</b>
<b>+, - (бинарные)</b>
<b>&gt;&gt;, &lt;&lt;</b>

Оператор
<b>&amp;</b>
<b>^,  </b>
<b>&lt;=, &lt;, &gt;, &gt;=</b>
<b>&lt;&gt;, !=, ==</b>
<b>=, +=, -=, /=, ...</b>

Оператор
<b>is, is not</b>
<b>in, not in</b>
<b>not</b>
<b>and</b>
<b>or</b>

# Операторы

---

## Трехместное выражение if/else

**x** **if** **y** **else** **z**

- Используется в выражениях
- Значение **x** вычисляется, только если значение **y** истинно

```
x = int(input("x = "))  
y = int(input("y = "))  
z = x if x>y else y  
print("z = ", z)
```



# Основные элементы языка Python

---

- **Каждый объект в Python имеет следующие характеристики:**
  - **идентичность;**
  - **тип;**
  - **значение.**

# Основные элементы языка Python

---

- **Идентичность объекта** – это целое уникальное и постоянное число, которое устанавливается для данного конкретного объекта.
- Идентичность объекта устанавливается однократно при его создании.
- Идентичность объекта ассоциируется с адресом объекта в памяти.
- Для получения значения идентичности объекта используется функция **id()**.

# Основные элементы языка Python

---

## • Идентичность объекта

```
>>> a=5
```

# Создаем объект – целое 5 и  
присваиваем его адрес в *a*

```
>>> id(a)
```

# Возвращаем его идентификатор (для 5)

```
1752381760
```

```
>>> ra=id(a)
```

# Значение *ra* совпадает с  
идентификатором (5)

```
>>> ra
```

```
1752381760
```

```
>>> id(5)
```

# Значение *a* совпадает с  
идентификатором 5

```
1752381760
```

# Основные элементы языка Python

---

- *Идентичность объекта*
- Т.о. в Python все является объектом, в том числе и числа.
- Значения идентичности для разных имен могут совпадать.
- **Важно:** объекты числового типа являются неизменяемыми! (*т.е. для экономии ресурсов с небольшими целыми значениями Python ссылается на уже существующие в памяти объекты*)

# Интеллектуальный калькулятор

---

- *Идентичность объекта*

```
>>> i=3
>>> j=3
>>> k=4-1
>>> id(i)
8790474430176
>>> id(j)
8790474430176
>>> id(k)
8790474430176
```

**Небольшие  
значения, id  
совпадают**

```
>>> i=30000000000
>>> j=30000000000
>>> id(i)
48910064
>>> id(j)
48910032
>>>
```

**Большие  
значения,  
id не совпадают**

# Основные элементы языка Python

---

- *Идентичность объекта*
- С помощью операторов **is** и **is not** можно сравнивать значения объектов на идентичность.

```
>>> a=3
>>> b=5
>>> a is b
False
>>> a=5
>>> b=5
>>> a is b
True
```

```
>>> a=5
>>> b=7
>>> a is b
False
>>> a is not b
True
```

# Типы данных в Python

---

- В Python каждый объект имеет определенный тип.
- Тип объекта определяет:
  - множество операций, которые поддерживаются данным объектом;
  - возможные значения для объектов данного типа.
- В Python нет необходимости в объявлении типа объекта!

# Типы данных в Python

---

- В языке Python применяется *динамическая типизация данных*: тип переменной определяется автоматически в процессе присваивания ей значения.
- Примеры:

```
>>> x=4      # x - целая, тип int, значение 44
>>> y=z=1.10  # y и z - float, значение 1.10
>>> x,y=y,x   #меняем значения местами
```



# Типы данных в Python

---

- Для проверки типа любого значения и любой переменной можно использовать функцию **type()**
- Пример:

```
>>> a=4
>>> b=1.1
>>> c=4+2j
>>> ta=type(a)
>>> tb=type(b)
>>> z="abcd"
>>> ta;tb;type(c); type(z)
```

## Результат

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

# Типы данных в Python

---

- **Типы данных в Python делятся на простые и составные.**
- **К простым типам относятся:**
  - **Числа: целые, вещественные и комплексные**
  - **Логический тип**
- **Составные типы:**
  - **Неизменяемые – строки, кортежи**
  - **Изменяемые – списки, словари и множества**

# Типы данных в Python

---

- **Основные типы:**
  - **int** – целые значения
  - **float** – вещественные
  - **complex** – комплексные
  - **bool** – логические (True, False)
  - **str** – строка символов или отдельный символ, заключенные в двойные или одинарные кавычки.

# Числовые типы в Python

---

- К числовым типам относятся:
  - Целые, вещественные и комплексные числа
  - Числа, фиксированной точности
  - Рациональные числа
  - Множества
  - Логические значения
- Целые числа можно представить в 10-чном, 16-чном, 8-чном и двоичном форматах

# Числовые типы в Python

---

- Целые числа можно представить в 10-чном, 16-чном, 8-чном и двоичном форматах
  - `30` # десятичный формат
  - `0x1e`, `0X1E` # шестнадцатеричный формат
  - `0o36` # восьмеричный (версия Python 3.0 и выше)
  - `036` # восьмеричный (в версиях Python до 3.0)
  - `0b11110`, `0B11100` # двоичный формат

# Функции преобразования

---

- **int**([x=0, [base=10]]) – преобразует x к целому числу в десятичной системе счисления. Параметр *base* от 2 до 36.
- **float**([x]) – преобразование к вещественному числу. Если аргумент не указан, возвращает 0.0.
- **bool**([x]) – преобразование к типу bool. Если x является ложным или опущен, возвращает значение False, иначе True.

# Функции преобразования

---

- **str(object=“”)** – строковое представление объекта

# Ввод и вывод данных

---



# Ввод и вывод данных

---

- **input([*prompt*])** – функция считывает и возвращает строку входных данных. Параметр *prompt* – строка-приглашение для ввода.
- **Пример:**  

```
my_str = input('Input your string: ')
```
- Для ввода значений других типов, требуется применить функции преобразования  

```
x = int(input('Input x: '))
```

# Ввод и вывод данных

---

- Для ввода нескольких значений можно воспользоваться методом `split()`, который позволяет разбить строку на подстроки.
- Пример:  

```
a, b = input('Input a, b: ').split();  
a = float(a)  
b = float(b)
```

# ВВОД И ВЫВОД ДАННЫХ

---

- `print(*objs, sep=' ', end='\n', file=sys.stdout, flush=False)`

**выводит заданные объекты на экран или отправляет их текстовым потоком в файл**

- **`objs`** – список вывода
- **`sep=""`** – разделитель для выводимых объектов (значение по умолчанию – `None`)
- **`end = '\n'`** – строка, выводимая после всех объектов (значение по умолчанию – `None`)

# ВВОД И ВЫВОД ДАННЫХ

---

- **print(\*objs, sep=",", end='\n', file=sys.stdout, flush=False)**  
выводит заданные объекты на экран или отправляет их текстовым потоком в файл
  - **file=sys.stdout** – ожидается объект, реализующий метод `write(string)`. Если значение не указано, либо `None` – будет использован `sys.stdout`
  - **flush = False** – Если `True` поток будет сброшен в файл принудительно.

# Ввод и вывод данных

---

- Примеры

```
print('При')  
print('вет!')  
print('При', end='')  
print('вет!')  
print('Раз', 'два', 'три')  
print('Раз', 'два', 'три', sep='--')
```

# Ввод и вывод данных

---

- Примеры

```
print('При')
print('вет!')
print('При', end='')
print('вет!')
print('Раз', 'два', 'три')
print('Раз', 'два', 'три', sep=
```

```
При
вет!
Привет!
Раз два три
Раз--два--три
>>> |
```

# Форматирование строк

---

- Часто возникают ситуации, когда нужно сделать строку, подставив в неё некоторые данные, полученные в процессе выполнения программы.
- Способы форматирования строк:
  - Оператор %
  - Метод `format`
  - f-строки (начиная с версии 3.6)

# Форматирование строк – оператор %

---

- В форматированной строке используются спецификаторы формата (схоже с printf (Java, C))
  - *Например, %d – целое число, %f – вещественное число, %s – строка и др.*
- Пример:

```
name=input("name = ")
age=int(input("age="))
print("Name =%s age=%d " % (name, age) )
```



# Форматирование строк – оператор %

---

- **Спецификаторы преобразования записываются в следующем порядке:**
  - **%.**
  - **Ключ (опционально), определяет, какой аргумент из значения будет подставляться.**
  - **Флаги преобразования.**
  - **Минимальная ширина поля.**
  - **Точность, начинается с '.'**
  - **Модификатор длины (опционально).**
  - **Тип**

# Форматирование строк – метод `format`

---

- Строках шаблонах используется специальный символ `{}`, который указывает на место подстановки значения из списка аргументов метода `format`.
- Типы подставляемых значений различны

```
name=input("name = ")
age=int(input("age="))
print("Name ={} age={} ".format(name,age))
```

# Форматирование строк – метод `format`

---

- По аналогии с `%`, метод `format` также предоставляет расширенные возможности форматирования
- Спецификация формата включает поля:
  - заполнитель – символы, кроме `{` и `}`
  - выравнивание – `<`, `>`, `^`; `=` (для чисел)
  - знак – `+`, `-`,  (только для чисел)
  - ширина и точность – целые значения
  - тип выводимых данных (`d`, `f`, `s` и др.)

# Форматирование строк – метод format

---

- Пример форматирования

```
st=input("st = ")
x=float(input("x="))
y=int(input("y="))
print("st ={:<15s} x={: .2f} y={: +5d} ".\
      format(st, x, y))
```

```
st =ewwe                x=-3.12 y= +436|
```

# Форматирование строк – f-строка

---

- С версии 3.6 и выше
- f-строка – это строка, перед которой стоит символ “f”
- f-строка содержит заменяющие поля (выражения в {})

```
name=input("name = ")  
age=int(input("age="))  
print(f"Name ={name} age={age} ")
```

# Форматирование строк – f-строка

---

- При форматировании f-строки использует тот же «миниязык» формирующих спецификаторов, что и метод `format`.
- В f-строках можно использовать преобразование “**!r**”, которое обеспечивает вывод строковых значений в кавычках.

# Форматирование строк – метод format

---

- Пример форматирования

```
st=input("st = ")
```

```
x=float(input("x="))
```

```
y=int(input("y="))
```

```
print("st ={:<15s} x={: .2f} y={: +5d}".format(st, x, y))
```

```
print(f"st ={{st!r:<15s}} x={{x: .2f}} y={{y: +5d}}")
```

```
st =wrq
```

```
x=-43.12 y= +34
```

```
st ='wrq'
```

```
x=-43.12 y= +34
```