# Введение в программирование

# Что такое программа?



**Компьютерная программа –** это список инструкций для компьютера.

Инструкции могут быть абсолютно произвольными:

- считать информацию с клавиатуры;
- произвести арифметические вычисления `(+, -, *, /);`
- вывести информацию на экран.

Windows

Word Office

Skype

Chrome

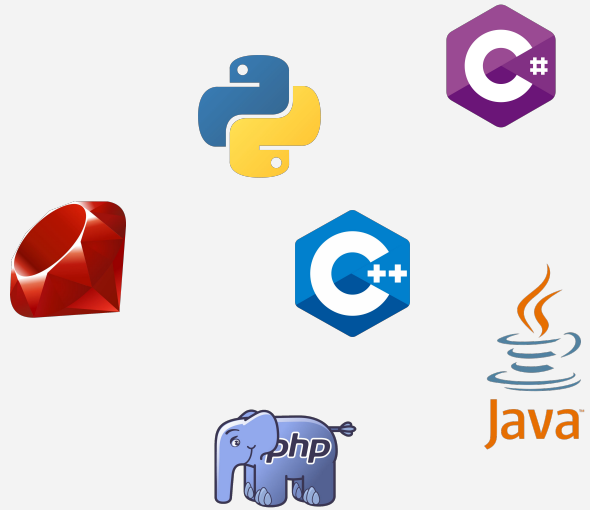
# Что такое язык программирования?



**Язык программирования –** набор определенных правил, согласно которым компьютер может понимать инструкции и выполнять их.

Текст программы называется **программным кодом.**

- Python
- C/C++
- C#
- Java
- PHP
- Ruby



# Типы языков программирования



**Языки программирования**

**Компилируемые**

Если программа написана на компилируемом языке, то перед выполнением ее нужно проверить на наличие синтаксических ошибок и уже после этого перевести в понятную для компьютера форму – машинный код.


**Интерпретируемые**

Если программа написана на интерпретируемом языке, она не переводится целиком в машинный код, а специальная программа, которая называется интерпретатором – идет по коду, анализирует его и выполняет каждую отдельную команду.



# Преимущества и недостатки Python



**Язык Python:**

- интерпретируемый
- платформо-независимый
- простой язык
- встраиваемый скриптовый язык
- динамическая типизация
- имеет огромную библиотеку классов на любой вкус

Основным недостатком языка Python является его низкая скорость выполнения.

# Задачи решаемые с помощью Python



**Python подходит для:**

- системного программирования
- графических приложений
- веб приложений
- веб-сценариев
- интеграции компонентов
- приложений баз данных
- приложений анализа данных

**Python не подходит для:**

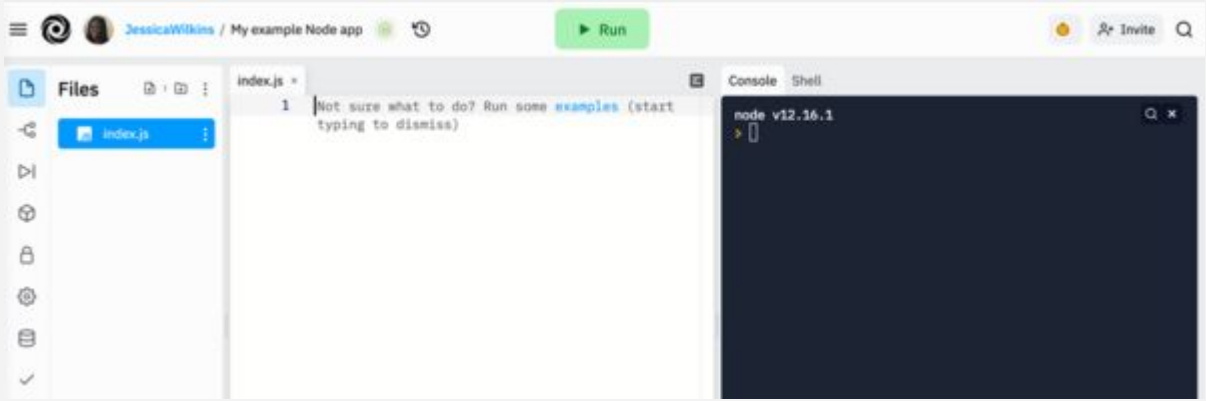
- низкоуровневых приложений
- высокопроизводительных приложений
- создания серьезных игр

# Среда разработки Replit (онлайн интерпретатор)

# Replit



**Среда разработки** – интерактивная онлайн-среда программирования с дополнительными возможностями.



# Команда print()

# Команда print()



Для вывода данных на экран используется команда **`print()`:**

```
print('Мы изучаем язык Python')
```

Кавычки могут быть как одинарными, так и двойными:

```
print('Python')
print("Python")
```

аргументы

# Аргументы команды print()



Команда **`print()`** позволяет указывать несколько аргументов

Аргументы отделяются запятыми:

```
print('Скоро я', 'буду программировать', 'на языке', 'Python!')
```

```
Скоро я буду программировать на
языке Python!
```

Команда **`print()`** добавляет ровно 1 пробел между всеми своими аргументами

# Примечания



Команда **`print()`** записывается только маленькими буквами

Команда **`print()`** выводит текст начиная с новой строки:

```
print('Какой хороший день!')
print('Работать мне не лень!')
```

```
Какой хороший день!
Работать мне не лень!
```

Команда **`print()`** без аргументов, вставляет пустую строку:

```
print('Какой хороший день!')
print()
print('Работать мне не лень!')
```

```
Какой хороший день!

Работать мне не лень!
```

# Необязательные параметры sep, end



## Параметр sep

```
print('a', 'b', 'c')
print('d', 'e', 'f')
```

```
a b c
d e f
```

```
print('a', 'b', 'c', sep='*')
print('d', 'e', 'f', sep='**')
```

```
a*b*c
d**e**f
```

**sep** = separator, разделитель

# Необязательные параметры sep, end



## Параметр end

```
print('a', 'b', 'c')
print('d', 'e', 'f')
```

```
a b c
d e f
```

```
print('a', 'b', 'c', end='@')
print('d', 'e', 'f', end='@@')
```

```
a b c@d e f@@
```

**end** = end, конец

# Переменные и команда input()

# Команда input()



Для считывания данных используется команда **`input():`**

```
print('Как тебя зовут?')
name = input()
print('Привет,', name)
```

1. Сначала программа выведет текст на экран «Как тебя зовут?»
2. Далее программа будет ждать от пользователя ввода данных
3. Введенные данные запишутся в переменную **`name`**

Каждая команда **`input()`** завершается нажатием **`Enter`** на клавиатуре

# Переменные



**Переменная** – это именованный участок памяти, в котором хранятся данные

```
name = 'Анна'
surname = input()
print('Привет,', name, surname)
```

- в переменной **name** хранится строка `'Анна'`
- в переменной **surname** хранится вводимый пользователем текст

Любая переменная имеет имя и значение

# Имя переменной



1. может содержать только латинские буквы `a-z, A-Z`, цифры и символ нижнего подчеркивания `_`
2. не может начинаться с цифры
3. по возможности должно отражать её назначение

**Верное имя**

```
name1
my_variable
_sum
a
TotalAmount
qwerty12my
```

**Имя с ошибкой**

```
5name
переменная
surname$%
```

# Имя переменной



Python – регистрозависимый язык программирования

`name` и `Name` – две совершенно разные переменные

Для именования переменных принято использовать стиль `lower_case_with_underscores` (маленькие буквы с подчеркиваниями)

# Значение переменной



**Значение переменной** – информация, хранящаяся в переменной.

В переменной может храниться текст, число и т. д.

Если вы хотите, чтобы у вас была переменная, нужно написать:

```
<имя переменной> = <значение переменной>
```

Оператор присваивания

Имя переменной всегда должно быть **слева** от знака равенства

# Значение переменной



Значение переменной можно переприсваивать:

```
subject = 'Химия'
subject = 'Биология'
print(subject)
```

Оператор присваивания сообщает переменной значение независимо от того, была ли эта переменная введена раньше или нет

```
name1 = 'Анна'
name2 = name1
```

Если у нас имеется переменная, мы можем делать с её значением все что угодно, например присвоить другой переменной

# Комментарии



**Комментарий** – примечание в коде программы, которое Python игнорирует

Комментарии могут помочь вам или кому-то другому, глядя на программу спустя некоторое время, понять принцип её работы

Любую строку можно превратить в комментарий, поместив перед ней символ #

# Комментарии



Однострочные комментарии:

```
# Это комментарий
print('Всем привет!')
```

Комментарий в конце строки:

```
print('Всем привет!')  # Это комментарий
```

# Целые числа

# Целые числа

Все предыдущие программы, работали с **текстовыми данными**
Во многих случаях нам нужно работать именно с **числами**

Для того, чтобы в языке Python создать переменную целого типа, нужно опустить кавычки при объявлении переменной:

```
num1 = 7                # num1 - это число
num2 = 10               # num2 - это число
num3 = num1 + num2      # num3 - это число
print(num3)
```

Будет выведено число 17

Числа обозначаются без кавычек, а строки с кавычками

# Арифметические операции

В Python над числами можно совершать **4 основные операции:**

| Операция | Описание |
|---|---|
| + | сложение |
| - | вычитание |
| * | умножение |
| / | деление |

```
a = 3
b = 2
print(a + b)
print(a - b)
print(a * b)
print(a / b)
```

```
5
1
6
1.5
```

# Приоритет арифметических операций

Порядок выполнения арифметических операций в Python аналогичен порядку выполнения операций в математике:

| Операция | Приоритет |
|---|---|
| * | 1 |
| / | 1 |
| + | 2 |
| - | 2 |

```
num1 = 2 + 3 * 4
num2 = (2 + 3) * 4
print(num1)
print(num2)
```

```
14
20
```

для изменения приоритета понадобятся скобки

# Преобразование строки к целому числу

Чтобы преобразовать строку к целому числу, мы используем команду **int():**

```
age = '1992'
year = int(age)
grade = int(input())
```

Переменная **age** имеет строковый тип
Переменная **year** имеет целочисленный тип
Переменная **grade** имеет целочисленный тип

**int():** возьми то, что указано в скобках и преврати это в целое число

# Преобразование строки к целому числу

Программа, которая считывает два целых числа и выводит на экран их сумму:

```
num1 = int(input())
num2 = int(input())
print(num1 + num2)
```

чтобы считать одно целое число, мы пишем код:

```
num = int(input())
```

# Дополнительные операции

# Дополнительные операции



В Python над числами можно совершать **3 дополнительные операции:**

1. возведение в степень **
2. целочисленное деление //
3. нахождение остатка %

# Возведение в степень



Операция возведения в степень **a**n** возводит число a в степень n:

```
print(2 ** 0)
print(2 ** 1)
print(2 ** 2)
print(2 ** 3)
print(2 ** (-1))
```

```
1
2
4
8
0.5
```

В Python для возведения в степень используется символ **, а не ^

# Целочисленное деление



Операция целочисленного деления **//** отбрасывает десятичную часть результата:

```
print(10 // 3)
print(10 // 4)
print(10 // 5)
print(10 // 6)
print(10 // 12)
print(-10 // 12)
```

```
3
2
2
1
0
-1
```

При целочисленном делении округление берётся в меньшую сторону

# Нахождение остатка от деления



Операция нахождения остатка **%** возвращает остаток от деления двух целых чисел:

```
print(10 % 3)
print(10 % 4)
print(10 % 5)
print(10 % 6)
print(10 % 12)
print(3 % 7)
```

```
1
2
0
4
10
3
```

При нахождении остатка от деления на целое число **n**, получаются числа **0, 1, 2,…, n-1**

# Примечания



Операция **нахождения остатка** очень полезна при решении задач:

число делится на **n** нацело, если
остаток от деления на **n** равен **0**

Приоритет дополнительных операций:

| Операция | Приоритет |
|---|---|
| `**` | 0 |
| `//` | 1 |
| `%` | 1 |

операции **//** и **%** имеют такой же приоритет как и операции умножения и обычного деления

# Обработка цифр числа

# Цифры двузначного числа



При помощи операций `//` и `%` можно вычислять цифры числа:

```
num = 17
a = num % 10
b = num // 10
print(a)
print(b)
```

```
7
1
```

Получить последнюю цифра числа `n: n % 10`

Отделить последнюю цифру от числа `n: n // 10`

# Цифры трехзначного числа



При помощи операций `//` и `%` можно вычислять цифры числа:

```
num = 754
a = num % 10
b = (num % 100) // 10
c = num // 100
print(a)
print(b)
print(c)
```

```
4
5
7
```

# Алгоритм нахождения цифр



Алгоритм нахождения цифр `n`-значного числа `num`:

- последняя цифра: `(num % 10**1) // 10**0`;
- предпоследняя цифра: `(num % 10**2) // 10**1`;
- предпредпоследняя цифра: `(num % 10**3) // 10**2`;
- .....
- вторая цифра: `(num % 10**n-1) // 10**n-2`;
- первая цифра: `(num % 10**n) // 10**n-1`.

# Числа с плавающей точкой и встроенные функции

# Числа с плавающей точкой

Для представления чисел с плавающей точкой в Python используется тип данных `float`

```
e = 2.71828  # литерал с плавающей точкой
pi = 3.1415  # литерал с плавающей точкой
```

# Преобразование строки к числу с плавающей точкой

Программа, которая считывает два вещественных числа и выводит на экран их сумму:

```
num1 = float(input())
num2 = float(input())
print(num1 + num2)
```

чтобы считать одно вещественное число, мы пишем код:

```
num = float(input())
```

Операции с вещественными числами такие же как с целыми числами

# Встроенные функции

Функции **`min()`** и **`max()`**, используются для определения соответственно минимального или максимального значения

```
a = max(3, 8, -3, 12, 9)
b = min(3, 8, -3, 12, 9)
c = max(3.14, 2.17, 9.8)
print(a)
print(b)
print(c)
```

```
12
-3
9.8
```

# Встроенные функции

Функция **`abs()`** используются для нахождения модуля (абсолютной величины) числа

```
print(abs(10))

print(abs(-7))

print(abs(0))

print(abs(-17.67))
```

```
10
7
0
17.67
```

# Встроенные функции

Функция `len()` используются хтобы посчитать длину строки

```
s1 = 'abcdef'

length1 = len(s1)

length2 = len('Python rocks!')

print(length1)

print(length2)
```

```
6
13
```

# Встроенные функции

Для преобразования строки к числу мы использовали функции **`int()`** и **`float()`**.
Для обратного преобразования, то есть из числа в строку мы используем функцию **`str()`**

```
num1 = 1777

num2 = 17.77

s1 = str(num1)

s2 = str(num2)
```

```
s1 = '1777'

s2 = '17.77'
```

# Конкатенация строк

Строки, как и числа, можно складывать.

```
s1 = 'ab' + 'bc'

s2 = 'bc' + 'ab'

s3 = s1 + s2 + '!!'

print(s1)

print(s2)

print(s3)
```

```
abbc

bcab

abbcbcab!!
```

# Умножение строки на число

В Python так же можно умножать строку на число.

```
s = 'Hi' * 4

print(s)
```

```
HiHiHiHi
```