IT CAREER HUB

# Урок 1.

# Основы работы с Python

IT CAREER HUB

# Особенности языка Python

**Python — это высокоуровневый язык программирования, известный своей простотой и читабельностью.**

Рассмотрим его ключевые особенности:

- **Простота**: Синтаксис Python близок к естественному языку (английскому), что делает код легко читаемым и понятным.
- **Динамическая типизация**: Python автоматически определяет тип данных, в отличие от большинства других языков.
- **Интерпретируемый**: Python выполняет код строчка за строчкой, что облегчает отладку и разработку.
- **Кроссплатформенность**: Python работает на разных операционных системах (Windows, Linux, macOS).
- **Богатая стандартная библиотека**: В Python уже есть множество встроенных инструментов и модулей, которые помогают решать задачи.
- **Поддержка ООП и функционального программирования**: Python поддерживает оба подхода, что делает его гибким для разных типов задач.
- **Сообщество и экосистема**: Python активно развивается и имеет большое сообщество, что обеспечивает доступ к множеству ресурсов и библиотек.

Это делает Python универсальным и популярным выбором для веб-разработки, анализа данных, машинного обучения и автоматизации.

IT CAREER HUB

# Интерпретатор и компилятор

**Компилятор — это программа, которая сначала переводит весь исходный код в машинный код, а затем запускает его.**

Скомпилированные программы работают быстрее, но требуют больше времени на подготовку, так как код сначала компилируется.

**Интерпретатор — это программа, которая выполняет код построчно, сразу переводя его в машинный код.**

**Python — интерпретируемый язык, что позволяет запускать программы без предварительной компиляции.**

Это ускоряет разработку и упрощает отладку, но может быть медленнее по сравнению с компилируемыми языками.

Главное отличие: компилятор преобразует весь код сразу, а интерпретатор исполняет его по мере чтения.

**Машинный код — это низкоуровневые инструкции, которые процессор компьютера может напрямую выполнять.**

Он состоит из бинарных данных (0 и 1) и специфичен для каждого типа процессора. Программы сначала переводятся из исходного кода в машинный код, чтобы их можно было запустить на компьютере.

IT CAREER HUB

# Jupyter Notebook

**Jupyter Notebook — это интерактивная среда разработки и обмена документами, которая позволяет создавать и выполнять код Python в виде ячеек.**

Он предоставляет удобное окружение для разработки, тестирования и демонстрации кода Python.

Jupyter Notebook состоит из веб-интерфейса, который позволяет создавать и редактировать ноутбуки, а также встроенного интерпретатора Python, который выполняет код в каждой ячейке ноутбука. Каждая ячейка может содержать код, текстовые описания, формулы, изображения, графики и другие элементы.

Одним из ключевых преимуществ Jupyter Notebook является его интерактивность. Разработчик может выполнять код в ячейках по одной или несколько сразу, что позволяет получать немедленные результаты и видеть вывод прямо в ноутбуке. Это удобно для тестирования и экспериментирования с кодом.

IT CAREER HUB

# Переменная

**Переменная — это именованная область памяти, в которой хранятся данные.**

После создания переменной можно получить доступ к хранящихся в ней данным по заданному имени.

**Данные — это информация, которая может быть обработана и сохранена компьютером.**

В программировании данные представляют собой любые значения, которые программа использует для выполнения операций с ними. Эти значения могут быть числами, строками, списками, логическими значениями и другими типами.

Пример:

- **Целые числа:** 5, -3, 42.
- **Числа с плавающей запятой или вещественные:** 3.14, -0.001, 2.0.
- **Строки:** "Привет", 'Мир'.

Данные могут поступать из разных источников, таких как файлы, базы данных, пользовательский ввод или внешние устройства. Программы обрабатывают эти данные, изменяют их, анализируют или сохраняют для дальнейшего использования.

IT CAREER HUB

# Оператор присваивания

**Оператор присваивания — это символ, который используется для присвоения значения переменной.**

В Python оператор присваивания обозначается знаком =. Он связывает переменную с заданным значением.

Пример использования:

Python

```
x = 10

name = "Jonh"

x

10

name

'John'
```

После выполнения операции присваивания переменные будут хранить указанные значения, и вы сможете использовать их в дальнейшем в программе.

IT CAREER HUB

# Задание для закрепления

1. Оператор присваивания в Python обозначается символом:
    a. ==
    b. =
    c. #

2. Соотнесите термин с его определением:

| | |
|---|---|
| 1. Компилятор | a. Программа, которая выполняет код построчно. |
| 2. Интерпретатор | b. Низкоуровневый код, который исполняется компьютером напрямую. |
| 3. Оператор присваивания | c. Программа, которая переводит весь код в машинный до его выполнения. |
| 4. Машинный код | d. Символ, связывающий переменную с определённым значением. |

IT CAREER HUB

# Правила именования переменных

- Имя переменной может содержать только буквы, цифры и знак подчеркивания _.
- Начинается только с буквы или знака подчеркивания. (Например **1number** не является допустимым именем)
- Переменные регистрозависимы ("age" и "Age" - разные переменные).
- Не используйте зарезервированные слова (как **if**, **else**, **while** и т.д.).
- Не используйте названия функций (как **print**, **int**, **max** и т.д.).
- В Python переменные принято записывать в формате snake_case, т.е. если название переменной состоит из нескольких слов, то принято писать их с маленькой буквы и через знак нижнего подчёркивания. (Например orders_number). Но встречаются и имена в CamelCase, что не будет явной ошибкой.
- По имени переменной должно быть понятно что в ней хранится. (Например, student_age лучше, чем x.)
- Имя переменной должно быть достаточно длинным, чтобы передавать смысл, но не слишком длинным. Это улучшает читаемость. (Например client_number лучше, чем number_of_the_client)

Примеры правильных имен переменных:

| | |
|---|---|
| 2nd_coordinate = 33 | Начинается с цифры |
| my-variable = 10 | Содержит дефис |
| my variable = 10 | Содержит пробел |
| total%items = 5 | Содержит недопустимый символ "%" |
| if = 42 | Использует зарезервированное слово "if" |

IT CAREER HUB

# Задание для закрепления

Определите, какие из имен переменных являются корректными именами переменных в Python.

a. student_name
b. age_25
c. 2nd_place
d. is_student
e. my-variable
f. if
g. user_@ge
h. total_score
i. num 1
j. name!

IT CAREER HUB

# Синтаксис

**Синтаксис — это набор правил, определяющих, как должны быть организованы конструкции в языке программирования.**

Он описывает, как правильно формулировать команды, выражения и структуры кода, чтобы интерпретатор мог их корректно распознать и выполнить.

В контексте Python синтаксис включает:

1. **Структуру кода**: Правила для написания инструкций и блоков кода, таких как отступы и использование двоеточий.
2. **Идентификаторы**: Правила для именования переменных, функций и классов.
3. **Строки и комментарии**: Как объявлять строки и использовать комментарии.
4. **Операторы**: Правила для использования арифметических, логических и других операторов.

Корректный синтаксис необходим для того, чтобы код выполнялся без ошибок и был понятен другим программистам.

**Некоторые правила синтаксиса:**

- Конец строки является концом инструкции.

```python
num1 = 5
num2 = 7
```

- Иногда возможно записать несколько инструкций в одной строке, разделяя их точкой с запятой. Однако это не рекомендуется

```python
num1 = 5; num2 = 7
```

Позже мы познакомимся и с другими правилами синтаксиса.

IT CAREER HUB

# Ошибки синтаксиса

**Ошибки синтаксиса (SyntaxError) — это ошибки, возникающие когда код не соответствует правилам языка. Эти ошибки препятствуют интерпретатору понять и выполнить код.**

При запуске программы с ошибкой синтаксиса интерпретатор Python остановит выполнение кода и выведет сообщение об ошибке. Это сообщение обычно включает в себя следующие элементы:

1. **Тип ошибки**: Указывается, что это ошибка синтаксиса SyntaxError.
2. **Описание ошибки**: Поясняет, что именно пошло не так.
3. **Номер строки**: Указывает строку кода, где возникла ошибка, что помогает разработчику быстро найти проблему.
4. **Строка кода**: Обычно отображается сам код с указанием места ошибки с помощью стрелки (^) под проблемным участком.

Примеры ошибок синтаксиса включают:

- Использование недопустимых символов в именах переменных.

```
1number = 5
```

```
  Cell In[9], line 1
    1number = 5
    ^
SyntaxError: invalid decimal literal
```

- Неправильное использование кавычек для строк.

```
name = "John'
```

```
  Cell In[10], line 1
    name = "John'
           ^
SyntaxError: unterminated string literal (detected at line 1)
```

- Неправильные управляющие конструкции.

```
1+*2
```

```
  Cell In[2], line 1
    1+*2
      ^
SyntaxError: invalid syntax
```

В результате выполнение программы прерывается, и никакие инструкции после строки с ошибкой не выполняются. Для исправления ошибки необходимо внести изменения в код и повторно запустить программу.

IT CAREER HUB

# Комментарии

Комментарии в Python используются для пояснения кода и делают его более понятным. Они не влияют на выполнение программы, так как интерпретатор игнорирует их. Вот основные правила и виды комментариев в Python:

**Комментарии**

- Начинаются с символа # и продолжаются до конца строки.
- Используются для кратких пояснений.

```python
x = 5  # Присваиваем значение 5 переменной x
```

IT CAREER HUB

# PEP 8

**PEP 8 (Python Enhancement Proposal 8) — это документ, который содержит рекомендации по стилю написания кода на Python.**

Его цель — улучшить читаемость кода. Соблюдение PEP 8 помогает разработчикам писать чистый и понятный код, который легче поддерживать и развивать.

Хотя PEP 8 не является строгим правилом, его рекомендации часто используются в проектах на Python и поддерживаются большинством современных инструментов для форматирования кода.

Первые рекомендации PEP 8:

1. **Длина строки**: Ограничьте длину строк 79 символами.
2. **Пробелы**: Добавляйте пробелы вокруг знаков арифметических операций, но не перед запятыми или открывающими скобками.
3. **Именование:** Используйте snake_case для переменных и функций и UPPER_CASE для констант.
4. **Комментарии**: Пишите комментарии для пояснения кода.

По мере изучения Python мы будем знакомиться и с другими рекомендациями. Соблюдение PEP 8 помогает делать код более понятным и легким для поддержки.

Ссылка на полный документ: PEP 8 — Style Guide for Python Code.

IT CAREER HUB

# Задание для закрепления

1. Какой результат выведет следующий код?

```python
a = 3

b = 2a + 1
```

   a. 3 7
   b. Ошибка
   c. 7
   d. 2a + 1

2. Для создания комментария в Python можно использовать:
   a. /*
   b. #
   c. '''
   d. //

IT CAREER HUB

# Функция

**Функция — это заранее написанный блок кода, который выполняет определённую задачу.**

В Python есть встроенные функции, которые можно использовать, просто вызвав их по имени.

**Вызов функции — это процесс выполнения функции в коде. Чтобы вызвать функцию, нужно:**

- указать её имя
- указать круглые скобки сразу после имени (именно они инициируют выполнение функции)
- и передать необходимые данные (**аргументы**) в круглых скобках, если это необходимо.

Пример вызова функции:

```python
print("Привет, мир!")
```

Здесь:

- print — это имя функции.
- () — вызов функции.
- "Привет, мир!" — это аргумент, который передаётся функции.

Когда функция вызывается, она выполняет записанные заранее действия с переданными данными.

IT CAREER HUB

# Задание для закрепления

Что из перечисленного делает вызов функции?

a. Создает новую функцию
b. Выполняет ранее написанную функцию
c. Определяет переменную
d. Завершает программу

IT CAREER HUB

# Функция print

**Функция print — это встроенная функция в Python, которая выводит информацию на экран (в консоль). Она может выводить текст, числа, переменные и другие типы данных.**

Пример использования:

```python
print("Привет, мир!")
```

Этот код выведет строку "Привет, мир!" на экран.

```
print("Привет, мир!")
```

```
Привет, мир!
```

Функция print также может выводить несколько значений, разделяя их пробелами:

```python
print("Число:", 10)
```

Здесь выводится текст "Число:", значение 10 и пробел между ними.

```
print("Число:", 10)
```

```
Число: 10
```

Пример вызова функции без аргументов:

```python
print()
```

В этом случае выводит на экран пустую строку.

IT CAREER HUB

# Функция input

**Функция input — это встроенная функция в Python, которая позволяет получать данные от пользователя через ввод с клавиатуры.**

Когда вызывается input(), программа приостанавливается и ждёт, пока пользователь введёт текст и нажмёт Enter. На месте вызова функции появляется возвращаемое значение, которое далее можно присвоить переменной или передать в другую функцию.

Пример использования:

```python
name = input()
print("Привет,", name)
```

Вы также можете передать аргумент в виде строки, который будет выведен на экран, чтобы пользователь понимал что именно он должен ввести.

Пример использования:

```python
name = input("Введите ваше имя: ")
print("Привет,", name)
```

Здесь:

- Функция input выводит строку "Введите ваше имя: " и ждёт ввода от пользователя.
- Введённый текст сохраняется в переменной name.
- Затем с помощью print выводится приветствие с именем пользователя.

Функция input() всегда возвращает данные в виде строки.

IT CAREER HUB

# Задание для закрепления

Соотнесите функцию с её описанием:

| | |
|---|---|
| 1. print() | a. Получает данные от пользователя через ввод с клавиатуры. |
| 2. input() | b. Выводит информацию на экран. |

IT CAREER HUB

# Практическая работа

1. Напишите программу, которая выводит на экран строку "Hello, Python!".

   **Пример вывода:**

   Hello, Python!

2. Напишите программу, которая попросит пользователя ввести его возраст, а затем выведет его на экран.

   **Пример вывода:**

   Введите ваш возраст:

   Возраст