# 📘 День 2

## Встроенные функции

В Python есть множество встроенных функций. Встроенные функции глобально доступны для вашего использования, что означает, что вы можете использовать встроенные функции без импорта или настройки. Некоторые из наиболее часто используемых встроенных функций Python следующие: \_print()\_, \_len()\_, \_type()\_, \_int()\_, \_float()\_, \_str()\_, \_input()\_, \_list()\_, \_dict()\_, \_min()\_, \_max()\_, \_sum()\_, \_sorted()\_, \_open()\_, \_file()\_, \_help()\_ и \_dir()\_. В следующей таблице вы увидите исчерпывающий список встроенных функций Python, взятый из [документации python](https://docs.python.org/3.9/library/functions.html).

![Встроенные функции](../images/builtin-functions.png)

Давайте откроем оболочку Python и начнем использовать некоторые из наиболее распространенных встроенных функций.

![Встроенные функции](../images/builtin-functions\_practice.png)

Давайте потренируемся, используя различные встроенные функции

![Help and Dir Built in Functions](../images/help\_and\_dir\_builtin.png)

Как видно из терминала выше, в Python есть зарезервированные слова. Мы не используем зарезервированные слова для объявления переменных или функций. Мы рассмотрим переменные в следующем разделе.

Думаю, к настоящему моменту вы уже знакомы со встроенными функциями. Давайте сделаем еще одну тренировку встроенных функций и перейдем к следующему разделу.

![Min Max Sum](../images/builtin-functional-final.png)

## Переменные

Переменные хранят данные в памяти компьютера. Мнемонические переменные рекомендуется использовать во многих языках программирования. Мнемоническая переменная - это имя переменной, которое легко запоминается и ассоциируется. Переменная относится к адресу памяти, в котором хранятся данные.

Число в начале, специальный символ, дефис не допускаются при именовании переменной. Переменная может иметь короткое имя (например, x, y, z), но настоятельно рекомендуется использовать более описательное имя (имя, фамилия, возраст, страна).

Правила именования переменных в Python

- Имя переменной должно начинаться с буквы или символа подчеркивания

- Имя переменной не может начинаться с цифры

- Имя переменной может содержать только буквенно-цифровые символы и символы подчеркивания (A-z, 0-9 и \\_ ).

- Имена переменных чувствительны к регистру (firstname, Firstname, FirstName и FIRSTNAME) - это разные переменные)

Приведем имена допустимых переменных

``Шелл

фамилия

фамилия

возраст

страна

город

имя\_фамилия

фамилия

столица\_города

\_if # если мы хотим использовать зарезервированное слово в качестве переменной

год\_2021

год2021

текущий\_год\_2021

год рождения

нум1

нум2

```

Недопустимые имена переменных

``Шелл

имя и фамилия

имя@фамилия

фамилия

num-1

1num

```

Мы будем использовать стандартный стиль именования переменных Python, который был принят многими разработчиками Python. Разработчики Python используют соглашение об именовании переменных в змеином регистре (snake\_case). Мы используем символ подчеркивания после каждого слова для переменной, содержащей более одного слова (например, first\_name, last\_name, engine\_rotation\_speed). Пример ниже - это пример стандартного именования переменных, подчеркивание требуется, когда имя переменной состоит более чем из одного слова.

Когда мы присваиваем переменной определенный тип данных, это называется объявлением переменной. Например, в приведенном ниже примере мое имя присваивается переменной first\_name. Знак равенства - это оператор присваивания. Присвоение означает сохранение данных в переменной. Знак равенства в Python не является равенством, как в математике.

\_Пример:\_

``py

# Переменные в Python

first\_name = 'Asabeneh'

last\_name = 'Yetayeh'

страна = 'Финляндия'

город = 'Хельсинки'

возраст = 250

is\_married = True

навыки = ['HTML', 'CSS', 'JS', 'React', 'Python'].

информация о человеке = {

'firstname':'Asabeneh',

'lastname':'Yetayeh',

'страна': 'Финляндия',

'город': 'Хельсинки'

}

```

Давайте воспользуемся встроенными функциями \_print()\_ и \_len()\_. Функция print принимает неограниченное количество аргументов. Аргумент - это значение, которое можно передать или поместить внутрь скобки функции, см. пример ниже.

\*\*Пример:\*\*

``py

print('Hello, World!') # Текст Hello, World! является аргументом

print('Hello',',', 'World','!') # он может принимать несколько аргументов, было передано четыре аргумента

print(len('Hello, World!')) # принимает только один аргумент

```

Выведем на печать, а также найдем длину переменных, объявленных вверху:

\*\*Пример:\*\*

``py

# Вывод значений, хранящихся в переменных

print('First name:', first\_name)

print('Длина первого имени:', len(first\_name))

print('Фамилия: ', фамилия)

print('Длина фамилии: ', len(last\_name))

print('Страна: ', страна)

print('Город: ', город)

print('Возраст: ', возраст)

print('Женат: ', is\_married)

print('Навыки: ', навыки)

print('Информация о человеке: ', person\_info)

```

### Объявление нескольких переменных в одной строке

Несколько переменных также могут быть объявлены в одной строке:

\*\*Пример:\*\*

``py

имя\_фамилия, фамилия\_фамилия, страна, возраст, is\_married = 'Asabeneh', 'Yetayeh', 'Helsink', 250, True

print(имя\_имя, фамилия\_имя, страна, возраст, замужем\_ли)

print('First name:', first\_name)

print('Фамилия: ', фамилия)

print('Страна: ', страна)

print('Возраст: ', возраст)

print('Женат: ', is\_married)

```

Получение пользовательского ввода с помощью встроенной функции \_input()\_. Давайте присвоим данные, полученные от пользователя, переменным first\_name и age.

\*\*Пример:\*\*

``py

first\_name = input('Как вас зовут: ')

age = input('Сколько вам лет? ')

print(first\_name)

print(age)

```

## Типы данных

В Python существует несколько типов данных. Для определения типа данных мы используем встроенную функцию \_type\_. Я хотел бы попросить вас сосредоточиться на понимании различных типов данных. Когда речь идет о программировании, все дело в типах данных. Я ввел типы данных в самом начале, и я снова ввожу их, потому что каждая тема связана с типами данных. Мы рассмотрим типы данных более подробно в соответствующих разделах.

## Проверка типов данных и приведения

- Проверка типов данных: Чтобы проверить тип данных определенных данных/переменной, мы используем \_тип\_.

\*\*Пример:\*\*

``py

# Различные типы данных python

# Давайте объявим переменные с различными типами данных

first\_name = 'Asabeneh' # str

last\_name = 'Yetayeh' # str

страна = 'Финляндия' # str

город= 'Хельсинки' # str

возраст = 250 # int, это не мой реальный возраст, не беспокойтесь об этом

# Вывод типов

print(type('Asabeneh')) # str

print(type(first\_name)) # str

print(type(10)) # int

print(type(3.14)) # поплавок

print(type(1 + 1j)) # комплекс

print(type(True)) # bool

print(type([1, 2, 3, 4])) # список

print(type({'name':'Asabeneh','age':250,'is\_married':250})) # dict

print(type((1,2))) # кортеж

print(type(zip([1,2],[3,4]))) # установить

```

- Кастинг: Преобразование одного типа данных в другой тип данных. Мы используем \_int()\_, \_float()\_, \_str()\_, \_list\_, \_set\_.

При выполнении арифметических операций строковые числа должны быть сначала преобразованы в int или float, иначе будет выдана ошибка. Если мы конкатенируем число со строкой, число должно быть сначала преобразовано в строку. О конкатенации мы поговорим в разделе String.

\*\*Пример:\*\*

``py

# int в float

num\_int = 10

print('num\_int',num\_int) # 10

num\_float = float(num\_int)

print('num\_float:', num\_float) # 10.0

# float to int

гравитация = 9,81

print(int(gravity)) # 9

# int в str

num\_int = 10

print(num\_int) # 10

num\_str = str(num\_int)

print(num\_str) # '10'

# преобразование строки в int или float

num\_str = '10.6'

print('num\_int', int(num\_str)) # 10

print('num\_float', float(num\_str)) # 10.6

# строка в список

first\_name = 'Asabeneh'

print(first\_name) # 'Asabeneh'

first\_name\_to\_list = list(first\_name)

print(first\_name\_to\_list) # ['A', 's', 'a', 'b', 'e', 'n', 'e', 'h']

```

## Числа

Числовые типы данных в Python:

1. Целые числа: Целые (отрицательные, нулевые и положительные) числа

Пример:

... -3, -2, -1, 0, 1, 2, 3 ...

2. Числа с плавающей запятой (десятичные числа)

Пример:

... -3.5, -2.25, -1.0, 0.0, 1.1, 2.2, 3.5 ...

3. Комплексные числа

Пример:

1 + j, 2 + 4j, 1 - 1j

🌕 Вы великолепны. Вы только что завершили второй день испытаний, и вы на два шага впереди на пути к величию. Теперь сделайте несколько упражнений для мозга и мышц.

## 💻 Упражнения - день 2

### Упражнения: Уровень 1

1. Внутри 30DaysOfPython создайте папку под названием day\_2. Внутри этой папки создайте файл с именем variables.py

2. Напишите python комментарий к статье 'День 2: 30 дней программирования на python'

3. Объявите переменную первого имени и присвойте ей значение

4. Объявите переменную "Фамилия" и присвойте ей значение

5. Объявите переменную с полным именем и присвойте ей значение

6. Объявите переменную country и присвойте ей значение

7. Объявите переменную city и присвойте ей значение

8. Объявите переменную возраст и присвойте ей значение

9. Объявите переменную year и присвойте ей значение

10. Объявите переменную is\_married и присвойте ей значение

11. Объявите переменную is\_true и присвойте ей значение

12. Объявите переменную is\_light\_on и присвойте ей значение

13. Объявление нескольких переменных в одной строке