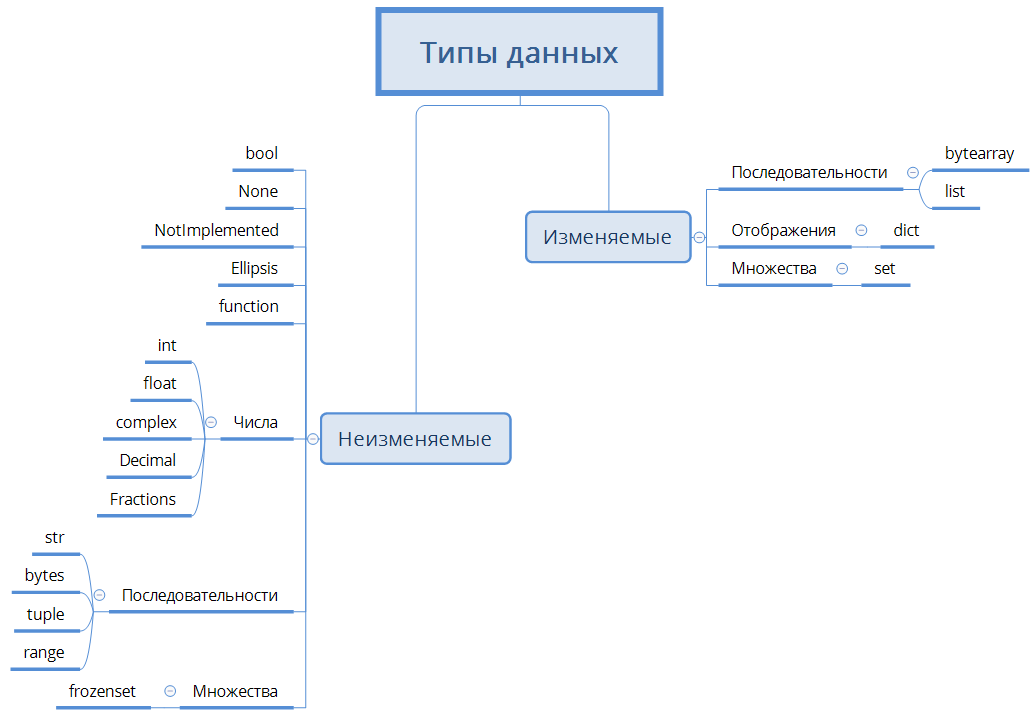
# Введение

Изменяемые и неизменяемые типы данных



Переменные в Python не требуют объявления типа переменной (так как Python – язык с динамической типизацией) и являются ссылками на область памяти. Правила именования переменных:

* имя переменной может состоять только из букв, цифр и знака подчёркивания;
* имя не может начинаться с цифры;
* имя не может содержать специальных символов @, $, %.

# Урок №1.

1. Hello World!

*# Первая программа  
#  
# Если в Python 2 скобки можно не использовать,  
# то в Python3 они обязательны.  
# Если их не указать, то будет вызвана синтаксическая ошибка.  
# Функция print выводит все элементы, разделяя их значением sep,  
# и завершает вывод значением end.  
# end по умолчанию использует перевод строки \n  
  
# Вывод: Hello, World!*print(**'Hello, World!'**)  
*# Вывод: Hello, World!*print(**'Hello,'**, **'World!'**, sep=**' '**) *# Разделитель пробел  
# Вывод: Hello,World!*print(**'Hello,'**, **'World!'**, sep=**''**) *# Разделителя нет  
# Вывод: Hello, World!*print(**'Hello, World'**, end=**'!\n'**) *# Окончание вывода с ! и переходом на новую строчку  
# Вывод: Hello, World!!*print(**'Hello,'**, **'World!'**, sep=**' '**, end=**'!'**) *# Разделитель пробел, окончание вывода с !  
  
# Для печати спец. символов нужно использоать экранирование \*print()  
*# Вывод: \t*print(**'\\t'**)  
*# Вывод: \n*print(**'\\n'**)  
*# Вывод: \*print(**'\\'**)

2. Объявление и разница строк

*# Строки  
# Одинарные и двойные кавычки равны между собой*my\_string\_o = **'Hello, World!'**my\_string\_d = **"Hello, World!"***# Вывод: Hello, World!*print(my\_string\_o)  
*# Вывод: Hello, World!*print(my\_string\_d)  
  
*# Тройные кавычки сохраняют форматирование*my\_string\_to = **''' Hello, World!'''**my\_string\_td = **"""  
 Hello, World!"""***# Вывод: Hello, World!*print(my\_string\_to)  
*# Вывод: ----------*print(**'-'** \* 10) *# Напечатать символ 10 раз  
# Вывод:   
# Hello, World!*print(my\_string\_td)  
  
*# Форматирование без тройных кавычек*my\_string\_of = **"\t\tHello, World!"** *# Повторение форматирования на строчке 9  
# Вывод: Hello, World!*print(my\_string\_of)  
*# Вывод: ----------*print(**'-'** \* 10) *# Напечатать символ 10 раз*my\_string\_df = **"\n\tHello, World!"** *# Повторение форматирования на строчке 10  
# Вывод:   
# Hello, World!*print(my\_string\_df)  
  
*# Использование кавычек внутри строки  
# Можно использовать одинарные кавычки внутри двойных и наоборот  
# Вывод: "Hello, World!"*print(**'"Hello, World!"'**)  
*# Вывод: 'Hello, World!'*print(**"'Hello, World!'"**)  
*# Либо использовать экранирование \  
# Вывод: 'Hello, World!'*print(**'\'Hello, World!\''**)  
  
*# Операции над строками  
# Вывод: HelloWorld*print(**'Hello'** + **'World'**)  
*# Вывод: HelloHelloHelloWorldWorld*print(**'Hello'** \* 3 + **'World'** \* 2)

3. Форматирование строк

*# Форматирование строк  
# Существует несколько способов форматирования строк*print(**'% / Python 2'**)  
*# Старый способ / Python 2  
# Раньше форматирование строк происходило при помощи оператора %*hello = **"Hi, my name is %s"** % **"Jane"***# Вывод: Hi, my name is Jane*print(hello)  
  
hello = **"Hi, my name is %s %s"** % (**"Jane"**, **"Doe"**)  
*# Вывод: Hi, my name is Jane Doe*print(hello)  
  
print(**'Format'**)  
*# Использование метода format  
# Подстановка по порядку*names = **"{}, {} and {}"**.format(**"John"**, **"Bob"**, **"Jane"**)  
*# Вывод: John, Bob and Jane*print(names)  
  
*# Подстановка по позиционному (positional) аргументу*names = **"{2}, {0} and {1}"**.format(**"John"**, **"Bob"**, **"Jane"**)  
*# Вывод: Jane, John and Bob*print(names)  
  
*# Подстановка по аргументу ключевому (keyword) слову*names = **"{a}, {c} and {b}"**.format(a=**"John"**, b=**"Bob"**, c=**"Jane"**)  
*# Вывод: John, Jane and Bob*print(names)  
  
print(**'f-строки'**)  
*# Использование f-строк*names = **f"{'John'}, {'Bob'} and {'Jane'}"***# Вывод: John, Bob and Jane*print(names)  
  
john = **'John'**bob = **'Bob'**jane = **'Jane'**names = **f'{**jane**}, {**bob**} and {**john**}'***# Вывод: Jane, Bob and John*print(names)  
  
print(**'-'** \* 10)  
  
*# Примеры форматирования с format и f-строками  
# Вывод числа в разных форматах  
# Вывод: int: 42; hex: 0x2a; oct: 0o52; bin: 0b101010*print(**"int: {0:d}; hex: {0:#x}; oct: {0:#o}; bin: {0:#b}"**.format(42))  
*# Вывод: int: 42; hex: 0x2a; oct: 0o52; bin: 0b101010*print(**f"int: {**42**:d}; hex: {**42**:#x}; oct: {**42**:#o}; bin: {**42**:#b}"**)  
  
*# Выравнивание строк  
# Вывод: Cat*print(**"{:15}"**.format(**'Cat'**)) *# Лево  
# Вывод: Cat*print(**"{:>15}"**.format(**'Cat'**)) *# Право  
# Вывод: Cat*print(**"{:^15}"**.format(**'Cat'**)) *# Центр  
# Вывод: \*\*\*\*\*\*Cat\*\*\*\*\*\**print(**"{:\*^15}"**.format(**'Cat'**)) *# Центр с заполнением символа \*  
  
# Вывод: Cat*print(**f"{'Cat':15}"**)  
*# Вывод: Cat*print(**f"{'Cat':>15}"**)  
*# Вывод: Cat*print(**f"{'Cat':^15}"**)  
*# Вывод: \*\*\*\*\*\*Cat\*\*\*\*\*\**print(**f"{'Cat':\*^15}"**)  
  
*# Выравнивание чисел  
# Целочисленные  
# Вывод: 12*print(**"{:5d}"**.format(12)) *# Отступ с шириной  
# Вывод: 12345*print(**"{:2d}"**.format(12345)) *# Отступ не работает, если число больше ширины  
# Дробные  
# Вывод: 012.2457*print(**"{:08.4f}"**.format(12.2456845)) *# Отступ заполненный нулями и ограничение тремя знаками после запятой  
# Вывод: 12.246*print(**"{:2.3f}"**.format(12.2456845)) *# Отступ заполненный нулями и ограничение тремя знаками после запятой*

4. Срезы

*# Срезы строк  
# Можно использовать только целочисленные значения 0, 1, 2 ...  
# Нельзя 0,5; 2.3 ...*str = **'Hello, Python!'***#  
# Печать первого символа*print(**f'str[0] = {**str[0]**}'**) *# Вывод: H  
# Печать последнего символа*print(**f'str[-1] = {**str[-1]**}'**) *# Вывод: !  
  
# TypeError: string indices must be integers*print(**f'str[0.5] = {**str[0.5]**}'**)  
  
*# IndexError: string index out of range. Выход за пределы строки*print(**f'str[15] = {**str[15]**}'**)  
*# IndexError: string index out of range. Выход за пределы строки*print(**f'str[-15] = {**str[-15]**}'**)  
  
*# Стрез со второго элемента по четвертый  
# Отсчет начинается с 0*print(**f'str[1:5] = {**str[1:5]**}'**) *# Вывод: str[1:5] = ello  
  
# Срез с шестого по второй элемент*print(**f'str[5:-2] = {**str[5:-2]**}'**) *# Вывод: str[5:-2] = , Pytho  
  
# Отрицательный срез*print(**f'str[-6:-2] = {**str[-6:-2]**}'**) *# Вывод: str[-6:-2] = ytho*

5. Изменение строк

*# Изменение строк  
# Строки являются неизменяемым типом данных  
# Нельзя изменить символы, нельзя удалять символы*str = **'Hello, World!'**str[0] = **'Q'  
del** str[3]

6. Ввод данных с клавиатуры

*# Ввод данных с клавиатуры  
# Данные возвращаются в строковом формате  
# Функция input позволяет вывод сообщения без вызова функции print*name = input(**"Name?\n"**)  
print(**"Hello"**, name)  
print(**"Hello "** + name)  
print(**f"Hello {**name**}"**)  
print(**"Hello {}"**.format(name))  
print(**"Hello %s"** % name)  
  
*# Функция type позволяет узнать тип переменной (имени)  
# <class 'str'>*print(type(name))

7. Приведение типов. Числа.

*# Конвертация str в int*number = input(**'Введите число: '**) *# 9*print(type(number)) *# Вывод: <class 'str'>*int\_n = int(number)  
print(int\_n) *# Вывод: 9*print(type(int\_n)) *# Вывод: <class 'int'>  
  
# Тип данных int  
# Числа в Python сразу могут в длинную математику*print(7 + 8) *# Сложение. Вывод: 15*print(7 - 8) *# Вычитание. Вывод: -1*print(7 \* 8) *# Умножение. Вывод: 56  
# Деление типов int всегда возвращает float, даже если деление нацело*print(7 / 7) *# Деление. Вывод: 1.0*print(8 // 7) *# Деление, остаток отсекается. Вывод: 1*print(8 % 7) *# Остаток от деления. Вывод: 1*print(8 \*\* 8) *# Возведение в степень. Вывод: 16777216  
  
# Тип данных Float*print(7.5 + 3.8) *# Сложение. Вывод: 11.3*print(7.5 - 3.8) *# Вычитание. Вывод: 3.7*print(7.5 \* 3.8) *# Умножение. Вывод: 28.5*print(7.5 / 7.5) *# Деление. Вывод: 1.0*print(3.8 // 7.5) *# Деление, остаток отсекается. Вывод: 0.0*print(3.8 % 7.5) *# Остаток от деления. Вывод: 3.8*print(3.8 \*\* 7.5) *# Возведение в степень. Вывод: 22303.703067314564  
  
# Конвертация int <-> float*print(int(2.6)) *# Вывод: 2. Дробная часть отсекается*print(float(2)) *# Вывод: 2.0  
  
# tbc round*

8. Bool

*# Bool  
# Логический тип представлен двумя постоянными значениями False и True.  
# Значения используются для представления истинности.  
# Можно представить, как False - 0, True - 1  
# Подтип int*print(bool(0)) *# Вывод: False*print(bool(1)) *# Вывод: True  
  
# Сравнение  
# В данном примере возвращается True/False в зависимости от равенства, if использовать необязательно*print(**True** == 0) *# Вывод: False*print(**False** == 1) *# Вывод: False  
  
# Можно сравнивать с другими числами*print(**True** > 0) *# Вывод: True*print(**False** <= 0) *# Вывод: True*print(**True** > 2) *# Вывод: True*print(**False** <= 10) *# Вывод: True  
  
# Bool с другими значениями*print(bool(**''**)) *# Вывод: False*print(bool(**'string'**)) *# Вывод: True*print(bool(10)) *# Вывод: True*print(bool(-10)) *# Вывод: True*print(bool(5.5)) *# Вывод: True*print(bool(-9.8)) *# Вывод: True*

9. Условия

*# Условная инструкция if-elif-else (оператор ветвления)  
# Любое число, не равное 0, или непустой объект - истина.  
# Числа, равные 0, пустые объекты и значение None - ложь  
# Операции сравнения применяются к структурам данных рекурсивно  
# Операции сравнения возвращают True или False  
# Логические операторы and и or возвращают истинный или ложный объект-операнд  
  
# Простая программа на понимание условий. Если введенное число == 5, то выводится надпись "Вы угадали число!"*print(**'Угадай число'**)  
a = int(input(**'Введите число: '**))  
*# 5 - искомое число***if** a == 5:  
 print(**'Вы угадали число!'**)  
**else**:  
 print(**'Введите другое число!'**)  
  
*# Обновленная версия с несколькими числами*print(**'Угадай число'**)  
a = int(input(**'Введите число: '**))  
*# 5, 55 - искомые число***if** a == 5:  
 print(**'Вы угадали число!'**)  
**elif** a == 55:  
 print(**'Вы угадали число!'**)  
**else**:  
 print(**'Введите другое число!'**)  
  
*# Использование или.*print(**'Угадай число'**)  
a = int(input(**'Введите число: '**))  
*# 5, 55 - искомые число***if** a == 5 **or** a == 55:  
 print(**'Вы угадали число!'**)  
**else**:  
 print(**'Введите другое число!'**)  
  
*# Использование и.*print(**'Угадай число'**)  
a = int(input(**'Введите число: '**))  
b = int(input(**'Введите число: '**))  
*# 5, 55 - искомые число***if** a == 5 **and** b == 55:  
 print(**'Вы угадали число!'**)  
**else**:  
 print(**'Введите другое число!'**)  
  
*# Что будет если пользователь введет не число, а букву  
# ValueError: invalid literal for int() with base 10: 'f'*print(**'Угадай число'**)  
a = input(**'Введите число: '**)  
b = input(**'Введите число: '**)  
**if not** a.isdigit() **or** b.isdigit():  
 print(**'Введите число, а не строчку!'**)  
 exit()  
*# 5, 55 - искомые число***if** int(a) == 5 **and** int(b) == 55:  
 print(**'Вы угадали число!'**)  
**else**:  
 print(**'Введите другое число!'**)  
  
*# или*print(**'Угадай число'**)  
a = input(**'Введите число: '**)  
b = input(**'Введите число: '**)  
**if** a.isdigit() **and** b.isdigit():  
 *# 5, 55 - искомые число* **if** int(a) == 5 **and** int(b) == 55:  
 print(**'Вы угадали число!'**)  
 **else**:  
 print(**'Введите другое число!'**)  
**else**:  
 print(**'Введите число, а не строчку!'**)

10. Цикл for