Переменные в Python

что такое, зачем нужны, как использовать

Переменные

По «научному»:

- Переменная – это простейшая именованная структура данных, в которой может быть сохранен промежуточный или конечный результат работы программы

По «простому»:

- Переменная – имя, которое ссылается на некоторые данные в памяти компьютера

Как именовать переменные

Имена переменных могут содержать **только** следующие символы:

- Буквы в нижнем регистре (от «a» до «z»)
- Буквы в верхнем регистре (от «A» до «Z»)
- Цифры (от 0 до 9)
- Нижнее подчеркивание («_»)

Имена переменных не могут начинаться с цифры!

```
1  username = 'BestUserEver'
2
3  project_X = 'Super secret project'
4
5  month_9 = 'September'
6
7  2_oh_no = 'No way'
```

Зарезервированные имена Python

Следующие имена **не следует** использовать в качестве имен переменных, так как они являются зарезервированным словами Python:

import	global	for	assert	in
from	local	while	del	is
as	nonlocal	continue	pass	not
class	try	break	with	and
def	except	if	True	or
lambda	finally	elif	False	async
return	raise	else	None	await

Имена переменных: best practices

- Переменные следует называть в стиле snake_case_with_underscores (Нижний регистр, слова через подчеркивания, PEP8)
- Константы следует называть в стиле UPPERCASE_WITH_UNDERSCORES (Верхний регистр, слова через подчеркивания, PEP8)
- Не используйте символ «l» (маленькая эль), «O» (заглавная о) или «I» (заглавная ай) как однобуквенные идентификаторы, так как в некоторых шрифтах они могут быть неразличимы (PEP8)
- Имена переменны должны отражать суть того, что они в себе хранят.
  - Никаких a1, a2, value и т.д.
  - Хорошие примеры: dog_color, phone_number и т.д.
- Если есть возможность не сокращать слова, то лучше не сокращать, ибо мы знаем из дзена «Читаемость имеет значение»

Присваивание переменных

Переменную в Python создать очень просто – нужно присвоить некоторому идентификатору значение при помощи оператора «=».

- Символ «=» во многих языках программирования обозначает «присваивание»
- Присваивание – операция связывания некоторых данных с некоторым именем переменной
- Присваивание **не копирует** значение, а только **прикрепляет имя** к объекту, который содержит данные
- Имя переменной – *ссылка* на какой-то объект, а не сам объект (имя вроде стикера, который можно наклеить на коробку, чтобы знать, что внутри)

```
1  user = 'SomeUser'
2  age = 25
3  skills = ['python', 'sql']
4
```

Обновление переменных

Операция	Что делает
var = value	Присваивает значение value переменной var
var += value	Присваивает сумму (var + value) переменной var
var -= value	Присваивает разность (var - value) переменной var
var *= value	Присваивает произведение (var * value) переменной var
var /= value	Присваивает частное (var / value) переменной var
var %= value	Присваивает остаток от деления (деление по модулю) (var % value) переменной var
var **= value	Присваивает результат возведения в степень (var ** value) переменной var
var //= value	Присваивает целую часть деления (var // value) переменной var

Вопрос-ответ

? Могу ли я в одной строке присвоить сразу несколько переменных?

✔ В Python – да. Но не во всех языках есть множественное присвоение

```
1  a, b, c = 1, 2, 3
```

Вопрос-ответ

? Как поменять местами значения 2-х переменных?

✔ В Python для этого не надо вводить 3-ю переменную!

```
>>> a = 1
>>> b = 2
>>> a, b = b, a
>>> a
2
>>> b
1
```

Выводим значение переменной на экран

Для того, чтобы вывести значение переменной на экран, нужно воспользоваться встроенной функцией:

- print(переменная)

```
1  some_var = 'Super variable'
2
3  print(some_var)
4  # Super variable
```

Вводим данные с клавиатуры

Для записи в переменную значения с клавиатуры, используется функция:

- input([строка-приглашение])

```
>>> age = input('Сколько тебе лет: ')
Сколько тебе лет: 56
>>> print(f"У нас разница в {abs(int(age) - 25)} год/лет")
У нас разница в 31 год/лет
```

Ваши вопросы

что необходимо прояснить в рамках данного раздела

Простейшие типы данных в Python

какие бывают, как с ними работать, динамическая типизация в Python

Тип данных

Тип данных характеризует одновременно:

- Множество допустимых значений, которые могут принимать данные, принадлежащие к этому типу
- Набор операций, которые можно осуществлять над данными, принадлежащими к этому типу

Какие бывают типы данных

Неизменяемыми (immutable)

закрытый ящик с окном (можно посмотреть, что внутри, но нельзя изменить), например:

- int
- float
- complex
- str
- tuple
- frozenset (immutable версия set)

Изменяемыми (mutable)

открытый ящик (можно посмотреть, что внутри, можно изменить содержимое, но нельзя изменить его тип), например:

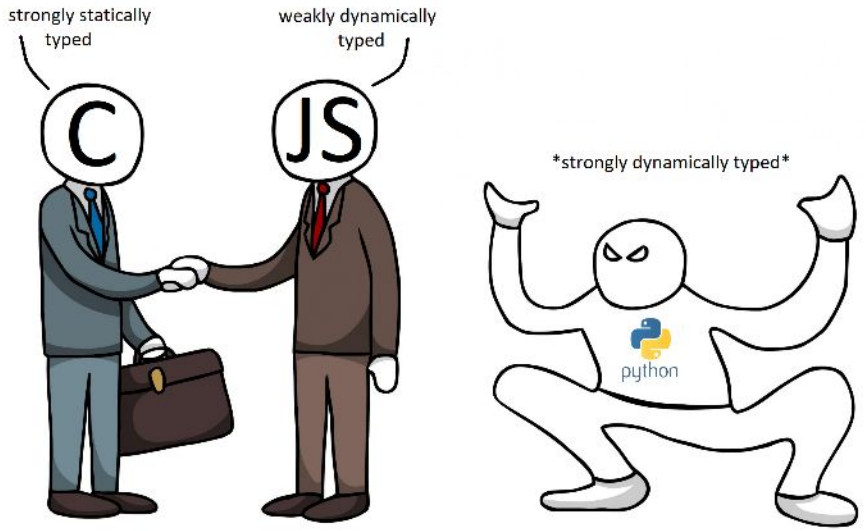
- list
- dict
- set
- bytearray

Динамическая типизация в Python

Динамическая типизация — прием, при котором переменная связывается с типом в момент присваивания значения, а не в момент объявления переменной.

В Python **строгая динамическая типизация** (Python не делает неявных преобразований типов). Чтобы проводить операции с разными типами их надо привести к одному типу данных

```
>>> 123 + 123
246
>>> 'Hello' + 'World'
'HelloWorld'
>>> 123 + '123'
Traceback (most recent call last):
  File "<input>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```



Простейшие типы данных

- Булевые значения (bool)
  - True
  - False
- Целые числа (int)
  - 123
  - -105264
  - и т.д.
- Числа с плавающей точкой (float)
  - числа с десятичной точкой (3.14159)
  - экспоненты (1.0e8, что означает «один умножить на 10 в 8 степени»)
- Комплексные числа (complex)
  - (1+2j)

```
>>> bool_var = True
>>> bool_var
True
>>> bool_var = False
>>> bool_var
False
>>> int_var = 123
>>> int_var
123
>>> int_var = -123
>>> int_var
-123
>>> float_var = 3.14159
>>> float_var
3.14159
>>> float_var = 1.0e8
>>> float_var
100000000.0
>>> complex_var = 1+2j
>>> complex_var
(1+2j)
>>> complex_var = complex(2, 5)
>>> complex_var
(2+5j)
```

Вопрос-ответ

? Как узнать тип переменной?

✔ Для того, чтобы узнать тип переменной в Python, необходимо выполнить функцию type(объект)

```
>>> type(1)
<class 'int'>
>>> type(1.3)
<class 'float'>
>>> type(1+2j)
<class 'complex'>
>>> type("hi")
<class 'str'>
>>> type((1, 2, 3))
<class 'tuple'>
>>> type({1, 2, 3})
<class 'set'>
>>> type({})
<class 'dict'>
>>> type({'color': 'black'})
<class 'dict'>
>>> type([1, 2, 3])
<class 'list'>
>>> type(b'\xd0\xd1\xd0')
<class 'bytes'>
>>> type(bytearray(b'hello world!'))
<class 'bytearray'>
```

Вопрос-ответ

? Как узнать, принадлежит ли объект к типу(типам)?

✔ Для того, чтобы узнать, принадлежит ли объект к какому-то определенному типу или к одному из нескольких типов, необходимо выполнить функцию isinstance(объект, [тип или кортеж типов]), а не прямым сравнением типов (PEP8)

```
>>> isinstance(123, int)
True
>>> isinstance(123, (int, str))
True
>>> isinstance('string', (int, str))
True
>>> isinstance('string', int)
False
```

Вопрос-ответ

? Как узнать, является ли тип подтипом другого типа(типов)?

✔ Для того, чтобы узнать, является ли тип подтипом другого, необходимо использовать функцию issubclass(тип, [тип или кортеж типов])

```
>>> issubclass(str, object)
True
>>> issubclass(str, (object, type))
True
>>> issubclass(str, int)
False
```

Вопрос-ответ

? Почему при выводе перед типом написано class?

✔ Потому что все, что есть в Python – это объект (экземпляр) некоторого класса (object)

```
1  >>> some_var = 'string'
2  >>> type(some_var)
3  <class 'str'>
4  >>> some_var = 123
5  >>> type(some_var)
6  <class 'int'>
```

Пустой тип (NoneType)

None – это ничего (нет значения или функционала):

- None можно присвоить переменной
- **Для сравнения с None используем оператор «is»**, а не «==» (во-первых PEP8, во-вторых можно словить баг, когда в своих классах переопределяется метод __bool__)

```
>>> var = None
>>> num = 123
>>> num is None
False
>>> var is None
True
>>> num is not None
True
>>> type(var)
<class 'NoneType'>
```

Логический тип (bool)

- Могут быть только 2 значения: True или False. Если рассматривать эти значения в числовом контексте (например, когда они используются как аргументы в арифметической операции), то ведут они себя как целые 1 и 0 соответственно
- Является подклассом целочисленного типа (int)
- От bool нельзя наследоваться
- Значения используются для обозначения истинности

```
>>> some_thing = True
>>> if some_thing:
...     print('YES')
...
YES
>>> isinstance(True, int)
True
>>> isinstance(False, int)
True
>>> issubclass(bool, int)
True
```

Приводим другие типы к bool

Для того, чтобы привести другой тип к логическому, необходимо выполнить функцию:

- bool(переменная_или_значение)

```
>>> bool({'a': 'b'})
True
>>> bool({})
False
>>> bool(1)
True
>>> bool()
False
>>> bool('string')
True
>>> bool('')
False
>>> bool(None)
False
```

int to bool

- 0 - это False
- Все остальное - это True

```
>>> bool(0)
False
>>> bool(1)
True
>>> bool(-123)
True
>>> bool(123)
True
```

float to bool

- 0.0 - это False
- Все остальное - это True

```
>>> bool(0.0)
False
>>> bool(1.12)
True
>>> bool(-1.12)
True
```

str to bool

- Пустая строка - это False
- Все остальное - это True

```
>>> bool('')
False
>>> bool('HI')
True
```

Коллекции к bool

- Если в dict, set, list, tuple и т.д. нет элементов, то False
- Если элементы есть, то True

```
>>> bool({'a': 1})
True
>>> bool({})
False
>>> bool(())
False
>>> bool((1))
True
>>> bool([])
False
>>> bool([1, 2])
True
>>> bool(set())
False
>>> bool(set('hi'))
True
```

Целочисленный тип (int)

Предполагается, что целые числа указываются в **десятичной** системе счисления, если не указана другая.

Система счисления указывает, сколько цифр можно использовать до того, как перенести единицу.

В Python числа можно выразить в 3-х системах счисления:

- 0b или 0B для двоичной системы (основание 2)
- 0o или 0O для восьмеричной системы (основание 8)
- 0x Или 0X для шестнадцатеричной системы (основание 16)

Насколько большой тип int

Python 2

- максимальный размер int – 32 бита, что достаточно чтобы сохранять числа в диапазоне от -2147483648 до 2147483648
- Для переменных типа long – 64 бита, что позволяло хранить значения от -9223372036854775808 до 9223372036854775808

Python 3

- int может быть любого размера, что позволяет нам даже умножить гугол (googol, 10**100) на гугол
- нет типа long

Приводим другие типы к int

Для того, чтобы привести другой тип к целочисленному, необходимо выполнить функцию:

- int(переменная_или_значение)

```
>>> int(True)
1
>>> int(False)
0
>>> int(3.1)
3
>>> int('123')
123
```

bool to int

- True – это 1
- False – это 0

```
1   >>> int(True)
2   1
3   >>> int(False)
4   0
```

float to int

При приведении чисел с плавающей точкой к целочисленным, «отсекается» часть, которая после запятой

```
1  >>> int(98.7)
2  98
3  >>> int(-12.3)
4  -12
```

str to int

- Можно привести строки, содержащие цифры с их знаком (13, +17, -12)
- Нельзя привести строки содержащие числа с плавающей точкой 13,8 1.0e4

```
>>> int('13')
13
>>> int('-13')
-13
>>> int('+13')
13
>>> int('13.8')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '13.8'
>>> int('1.0e4')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '1.0e4'
```

Вопрос-ответ

? Что мне делать, если я хочу работать с округлениями чисел?

✔ Есть несколько способов:

- встроенная функция round(число, кол-во знаков)
- math.ceil(число) из пакета math
- math.floor(число) из пакета math
- Класс Decimal из пакета decimal

Каверзный момент

Необходимо помнить, что числа с плавающей точкой «не совсем» точны, так как в памяти компьютера они представлены в двоичной системе счисления: https://docs.python.org/3.9/tutorial/floatingpoint.html#tut-fp-issues

```
>>> print(0.1 + 0.1 + 0.1)
0.30000000000000004
```

Округляем с round

Python 2

- Математическое округление

```
>>> round(1.4), round(1.5), round(1.6)
(1, 2, 2)
>>> round(2.4), round(2.5), round(2.6)
(2, 2, 3)
>>> round(2.85, 1)
2.9
>>> round(2.75, 1)
2.8
>>> round(2.65, 1)
2.6
```

Python 3

- «Банковское округление» или округление к ближайшему четному:
- https://nagornyy.me/courses/intro2python/banking-rounding-in-python/

Округляем с math

math.ceil

- Округляет «вверх»

```
>>> import math
>>> math.ceil(1.2)
2
>>> math.ceil(2)
2
>>> math.ceil(-0.5)
0
```

math.floor

- Округляет «вниз»

```
>>> import math
>>> math.floor(1.7)
1
>>> math.floor(2)
2
>>> math.floor(-0.5)
-1
```

Округляем с Decimal

- На вход подается строка
- Нельзя мешать с float
- Можно мешать с int

```
>>> from decimal import Decimal
>>> number = Decimal('0.444')
>>> number = number.quantize(Decimal('1.00'))
>>> number
Decimal('0.44')
>>> number = Decimal('0.555678')
>>> number = number.quantize(Decimal('1.00'))
>>> number
Decimal('0.56')
>>> number = Decimal('0.999')
>>> number = number.quantize(Decimal('1.00'))
>>> number
Decimal('1.00')
```

Числа с плавающей точкой (float)

Числа с плавающей точкой обрабатываются также, как и целые: вы можете использовать операторы +, -, *, /, //, **, % и функцию divmod

```
>>> 1.1 + 2.3
3.4
>>> 1.1 - 2.3
-1.1999999999999997
>>> 1.1 * 2.3
2.53
>>> 1.1 / 2.3
0.47826086956521746
>>> 1.1 // 2.3
0.0
>>> 1.1 ** 2.3
1.2450969688995253
>>> 1.1 % 2.3
1.1
>>> divmod(1.1, 2.3)
(0.0, 1.1)
```

Приводим другие типы к float

Для того, чтобы привести другой тип к числу с плавающей точкой, необходимо выполнить функцию:

- float(переменная_или_значение)

```
>>> float(True)
1.0
>>> float(12)
12.0
>>> float('12.3')
12.3
>>> float('1.0e4')
10000.0
```

bool to float

- True – это 1.0
- False – это 0.0

```
>>> float(True)
1.0
>>> float(False)
0.0
```

str to float

Можно привести строки, содержащие целые положительные и отрицательные числа, числа с плавающей точкой положительные и отрицательные, числа записанные через экспоненту

```
>>> float('123')
123.0
>>> float('-123')
-123.0
>>> float('-123.23')
-123.23
>>> float('123.23')
123.23
>>> float('2.0e3')
2000.0
```

Комплексные числа - complex

- Создаем с помощью complex(целая-часть, мнимая-часть)
- Либо с помощью добавления символа «j»

```
>>> from_function = complex(2, -3)
>>> from_function
(2-3j)
>>> type(from_function)
<class 'complex'>
>>> from_numbers = 2+3j
>>> from_numbers
(2+3j)
>>> type(from_numbers)
<class 'complex'>
```

Операторы в Python

какие бывают операторы, приведение типов

Что такое оператор

Оператор в Python – это символ, который выполняет операцию над одним или несколькими операндами

Операнд – это переменная или значение, над которым проводится операция

Операторы Python бывают 7 типов:

- Арифметические
- Сравнения (реляционные)
- Присваивания
- Побитовые
- Логические
- Операторы членства (Membership operators)
- Операторы тождественности (Identity operators)

Арифметические операторы

Оператор	Как называется	Что делает
+	Сложение	Суммирует значения слева и справа от оператора
-	Вычитание	Вычитает правый операнд из левого
*	Умножение	Перемножает операнды
/	Деление	Делит левый операнд на правый
%	Деление по модулю	Делит левый операнд на правый и возвращает остаток
**	Возведение в степень	Возводит левый операнд в степень правого
//	Целочисленное деление	деление, при котором возвращается только целая часть результата. Часть после запятой отбрасывается

Операторы сравнения

Оператор	Как называется	Что делает
==	Равно	Проверяет равны ли оба операнда. Если да, то условие становится истинным
!=	Не равно	Проверяет равны ли оба операнда. Если нет, то условие становится истинным
>	Больше	Проверяет больше ли значение левого операнда, чем значение правого. Если да, то условие становится истинным
<	Меньше	Проверяет меньше ли значение левого операнда, чем значение правого. Если да, то условие становится истинным
>=	Больше или равно	Проверяет больше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным
<=	Меньше или равно	Проверяет меньше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным

Операторы присваивания

Оператор	Как называется	Что делает
=	Присвоение	присваивает значение правого операнда левому
+=	Присвоение со сложением	прибавит значение правого операнда к левому и присвоит эту сумму левому операнду
-=	Присвоение с вычитанием	отнимает значение правого операнда от левого и присваивает результат левому операнду
*=	Присвоение с умножением	умножает правый операнд с левым и присваивает результат левому операнду
/=	Присвоение с делением	делит левый операнд на правый и присваивает результат левому операнду
%=	Присвоение с делением по модулю	делит по модулю операнды и присваивает результат левому
**=	Присвоение с возведением в степень	возводит в левый операнд в степень правого и присваивает результат левому операнду
//=	Присвоение с целочисленным делением	производит целочисленное деление левого операнда на правый и присваивает результат левому операнду

Побитовые операторы

Побитовые операторы предназначены для работы с данными в битовом (двоичном) формате. Предположим, что у нас есть два числа:

- a = 60
- b = 13

В двоичном формате они будут иметь следующий вид:

- a = 0011 1100
- b = 0000 1101

Побитовые операторы

Оператор	Как называется	Что делает
&	Бинарный «И»	Копирует бит в результат, если бит есть в обоих операндах: (a & b) даст нам 12, которое в двоичном формате выглядит так 0000 1100
|	Бинарный «Или»	Копирует бит, если тот есть хотя бы в одном операнде: (a | b) даст нам 61, в двоичном формате 0011 1101
^	Бинарный «Исключительное или»	Копирует бит, если бит присутствует в одном из операндов, но не в обоих сразу: (a ^ b) даст нам 49, в двоичном формате 0011 0001
~	Бинарный комплиментарный оператор	Является унарным (то есть ему нужен только один операнд) меняет биты на обратные, там где была единица становиться ноль и наоборот: (~a) даст в результате -61, в двоичном формате выглядит 1100 0011
<<	Побитовый сдвиг влево	Значение левого операнда "сдвигается" влево на количество бит указанных в правом операнде: a << 2 в результате даст 240, в двоичном формате 1111 0000
>>	Побитовый сдвиг	Значение левого операнда "сдвигается" вправо на количество бит

Логические операторы

Оператор	Как называется	Что делает
and	Логическое «И»	Условие будет истинным если оба операнда истина
or	Логическое «Или»	Если хотя бы один из операндов истинный, то и все выражение будет истинным
not	Логическое «Не»	Изменяет логическое значение операнда на противоположное

Операторы членства

Оператор	Как называется	Что делает
in	«Содержится в …», «Входит в …»	возвращает истину, если элемент присутствует в последовательности, иначе возвращает ложь
not in	«Не содержится в …», «Не входит в …»	возвращает истину если элемента нет в последовательности

Операторы тождественности

Оператор	Как называется	Что делает
is	«Является …»	возвращает истину, если оба операнда указывают на один объект: x is y вернет истину, если id(x) будет равно id(y)
is not	«Не является …»	возвращает ложь если оба операнда указывают на один объект

Приоритет операторов

	Операторы	Группа
1	(**a)	Возведение в степень
2	(~a) (+a) (-a)	Комплиментарный оператор
3	(a * b) (a / b) (a % b) (a // b)	Умножение, деление, деление по модулю, целочисленное деление
4	(a + b) (a – b)	Сложение и вычитание
5	(a << b) (a >> b)	Побитовый сдвиг вправо и побитовый сдвиг влево
6	(a & b)	Бинарный "И"
7	(a ^ b) (a | b)	Бинарный "Исключительное ИЛИ" и бинарный "ИЛИ"
8	(a <= b) (a < b) (a > b) (a >= b)	Операторы сравнения
9	(a <> b) (a == b) (a != b)	Операторы равенства
10	(a = b) (a %= b) (a /= b) (a //= b) (a -= b) (a += b) (a *= b) (a **= b)	Операторы присваивания
11	(a is b) (a is not b)	Тождественные операторы
12	(a in b) (a not in b)	Операторы членства
13	(a not b) (a or b) (a and b)	Логические операторы