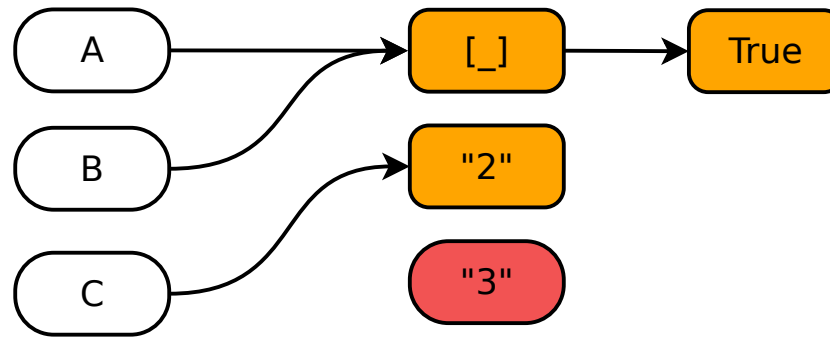


## Переменные (идентификаторы)

- Переменные - имена для объектов (ссылки на объект)
- Имя - [a-zA-Z\_][a-zA-Z\_0-9]\*
- Переменная создается присваиванием  
name = val (a = b = 1 работает, но a = (b = 1) нет)
- У одного объекта может быть много имен
- Часть имен зарезервирована (for, while, in, ..)  
`import` keyword; `print` keyword.kwlist
- Типизация динамическая - тип связывается с объектом. Можно считать что все переменные имеют тип (PyObject \*)
- В компилируемых языках другой подход. Переменная как ящик, в который кладутся данные и переменная определяет их поведение (тип)

## Переменные



Переменные от [David Goodger](#)



## Базовые типы данных

- None
- int (C long int), long
- float (c double)
- str, unicode
- bool – True, False
- complex
- ...

## Базовые типы данных

- Одно и то-же имя(переменная) может в разные моменты указывать на разные типы данных

```
1          x = 12
2          x = "12"
```

- Строгая типизация - не производится автоматических приведений типов, кроме очевидных (int -> float)

```
1          1 + "2" # error
2          1 + 1.0 == 2.0
3          (-1) ** 0.5 # error — no float -> complex conversion
```

- Однако все приводится к bool при использовании операций and/or/not

```
1          Empty objects , 0, 0.0, 0 + 0j, None => False
2          All else => True
3          obj and obj
4          obj or obj
```

- Типы нужно приводить явно

```
1          int("12") == 12
```

```
2      str(12) == "12"  
3      repr("12") == "12"  
4      1 + int("2") == 3
```

## Операции с целыми

- Все базовые типы данных - константны. Любая операция создает новый объект
- `+` `-` `/` `*` `%` `//` `**`
- Нет `--`, `++` (Вместо этого `-= 1`, `+= 1` )
- `+=`, `-=`, `*=`, ....
- `>`, `<`, `!=` (`<>` - устарел), `>=`, `<=`, `==` Не сравнивайте разные типы данных
- `is` vs `==`, `is not`
- `or` and `not`
- `&`    `|`    `^`
- `0 < x == y < 10`
- `math`, `cmath`

## Строки

```
1      "abc "
2      'abc '
3
4      b'abc '
5      B'abc '
6
7      """abcdef
8      gj
9      h""" == "abcdef\n gj\nh"
10
11     r"C:\temp\dir\fname" == "C:\\temp\\dir\\fname"
12
13     U"Unicode_text"
14     RU"Raw_unicode_text"
```



## Строковые операции

В python функции/методы возвращающие значение - делают копию, возвращающие None - модифицируют объект на месте.

```
1      "abc" > "def" == False
2      "abc" + "def" == "abcdef"
3
4      # a * N - a + a + a + ... + a, N раз
5      "ab" * 3 == "ababab"
6
7      "abcdef"[3] == "d"
8
9      # a[x:y] - substring [x, y)
10     "0123456789"[2:4] == "23"
11
12     a.replace(from, to)
13     "x_+_y".replace("+", "//") == "x_//_y"
14
15     a.find(string, pos)
16     "abcd".find("cd") => 2
17
18     # for x[:x.index(substr)]
19     "abcd".find("t") => -1
```

```
20
21     "cdabcd".find("cd", 1) => 4
22
23     a.index(string)
24     "abc".index("fd") # error
25
26     a.split(string)
27     "a,b,c,d".split(",") == ("a", "b", "c", "d")
28
29     "abc".startswith("ab") == True
30     "abc".endswith("dabc") == False
```

## Строки форматирование

- %, format

```
1      "size_=%d%s" % (12, 'cm')
2      # "size = 12 cm"
3
4      "size_=%(sz)d%(units)s" % dict(sz=1, units='m')
5      # "size = 1 m"
6
7      "Some_{ }_{}".format("very_interesting", "text")
8      #"Some very interecting text"
9
10     "Brown_fox_{what}_over".format(what="jump")
11     #"Brown fox jump over"
12
13     #"{name or index or EMPTY!conversion:format_spec}"
14     "{: ^25}".format(xx) == 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
```

## Unicode

- Unicode vs encoding
- Unicode libraries
- Unicode libraries complexity and problems
- encode/decode/encoding module
- `string.decode(encoding[, errors]) => unicode`
- `unicode.encode(encoding[, errors]) => string`
- Иногда python пытается автоматически перекодировать данные, используя текущую кодировку - `sys.getdefaultencoding()`

## type & id & hash & isinstance

- `type(x)` => тип `x`
- `id(x)` – int идентификатор значения (адрес в памяти)
- `hash(x)` - int - хэш значения, широко используется внутри питона
- `isinstance(x, X)` - проверяет, что `x` имеет тип `X` `isinstance(1, int)` `X` может быть списком в скобках

```
1     type(1) == int
2     d = "asad"
3     type(d) == str
4     type(1 is 2) == bool
5
6     id(1) == 34564790 #example
7     a is b # same as id(a) == id(b)
8
9     hash(1) == 1
10    hash("1") == 1977051568
```

## Особенности сравнений

- Сравнения между разными типами данных работает, но смысла не имеет
- Кроме комплексных чисел
- И юникодных строк с бинарными (иногда, зависит от кодировки - `sys.getdefaultencoding()`)

```
1      1 > "d" # False
2      1 > 1 + 1j # error
3      "d" > 2 + 1j # True
```