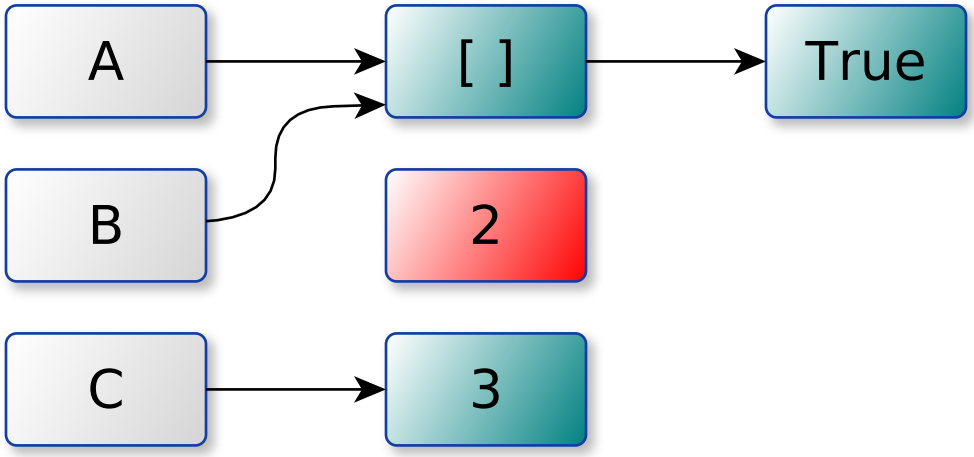
Переменные (идентификаторы)

- 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 языков другой подход. Переменная как ящик, в который кладутся байты и переменная определяет их поведение (тип)
- id(obj) -> int - уникальный идентификатор объекта. Для двух одновременно существующих объектов всегда разный, но может переиспользоваться после уничтожения объекта

Переменные



Переменные от David Goodger

Переменные

Одно и то-же имя(переменная) может в разные моменты указывать на данные разных типов

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

Базовые типы данных

- None
- int (long int), long
- float (double)
- str, unicode
- bool – True, False
- complex - 1 + 2.2j
- Elipsis, NotImplemented

Базовые типы данных

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

```
1 1 + "2" # TypeError
2 1 + 1.0 == 2.0
3 (-1) ** 0.5 # ValueError - no float -> complex convers
4 -1 ** 0.5 == -1 # ???
```

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

```
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
```

Базовые типы данных

Все базовые типы данных - константны. Любая операция создает новый объект.

```
1        x = 1
2        print id(x) #32230504
3        x += 1
4        print id(x) #32230480
```

Операции с int, float, complex

- + - / * % // **(возведение в степень)
- +=, -=, *=, ...
- >, <, != (<> - устарел), >=, <=, ==
- Не сравнивайте разные типы данных
- is vs ==, is not, (a is b) == (id(a) == id(b))
- or and not
- & | ^
- 0 < x == y < 10
- math, cmath
- Нет --, ++ (Вместо этого -= 1, += 1)

Строки

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

Строковые операции

```
"abc" > "def" == False
"abc" + "def" == "abcdef"

# a * N – a + a + a + ... + a, N раз
"ab" * 3 == "abababab"

"abcdef"[3] == "d"

# a[x:y] – substring [x, y)
"0123456789"[2:4] == "23"

a.replace(src, dst)
"x + y".replace("+", "//") == "x // y"

a.find(string, pos)
"abcd".find("cd") => 2

```

```
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") # ValueError
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) == '            xx
    '
```

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 `print "\xd1\x85\xd0\xb0\xd0\xb9".decode("utf`
- Иногда python пытается автоматически перекодировать данные, используя текущую кодировку - sys.getdefaultencoding()
- Pragmatic Unicode, or, How do I stop the pain?

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.

```
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
11      isinstance(1, int) == True
12      isinstance(1, (str, float)) == False
```

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 # TypeError
3 "d" > 2 + 1j # True
```

del

- Уничтожает переменную(имя), но не объект на который она указывает
- Объект будет уничтожен, когда на него никто не будет указывать

```
1       x = "1"
2       y = x
3       print id(x), id(y) # 139796728292736 139796728292736
4       del x
5       print x # NameError
6       print y, id(y) # 1 139796728292736
```