[Software Development]

Python 3

Davide Balzarotti

Eurecom – Sophia Antipolis, France

When Python is not Enough

- Python 2.7 was released in July 2010 and the 2.x branch will have no further major releases

- Python 3
  - Designed to break backwards compatibility with the 2.x series to fix a number of "*language flaws*"
  - Goal: reduce feature duplication by removing old ways of doing things

Some Differences

- All classes are now new-style
- `print` function
- String `%` operator is deprecated and replaced by `str.format()`
- All strings are unicode by default
- Exceptions
- No default comparisons
- Function annotations
- Many lists replaced with iterables (this breaks a lot of code)
- Division doesn’t truncate; `long()` is gone
- Library cleanup

Print function

```
print(a, b, c, …, end="", file=f, sep="")
```

```
Old: print "The answer is", 2*2
New: print("The answer is", 2*2)

Old: print x,                # Trailing comma suppresses newline
New: print(x, end=" ")  # Appends a space instead of a newline

Old: print                   # Prints a newline
New: print()                 # You must call the function!
```

Removed Default Comparisons

- In Python 2.x the default comparisons are overly forgiving

  `1 < "foo"` → True

- In Py3k incomparable types raise an error

```
1 < "foo"

Traceback …
TypeError: unorderable types: int() < str()
```

Integers

- Old long has been removed
- Python 3 has only one built-in integer type named `int` (which behaves mostly like the Python 2 `long` type)
- Division between integers returns a float: 1/2 == 0.5
  - Use 1//2 to get the truncating behavior

str.format

- `"%d %s"%(i, s)` → `"{} {}".format(i, s)`
  → `"{i} {s}".format(i=i, s=s)`
- `"%s-%s" % ("X", "X")` → `"{0}-{0}".format("X")`

```
": Fill Align Sign Width .Prec Type"

    "{:*>+10.1f}".format(12345.67) →  '**+12345.7'
    "{:~<-10.2f}".format(12345.67) →  '12345.67~~'
```

Exceptions

```
try:
   pass
except ValueError, err:
   print err
```

```
try:
   pass
except ValueError as err:
   print err
```

Function Annotations

```
def posint(n: int) -> bool:
    return n > 0


def f(x: “this is a number”):
    return x


f.__annotations__
```

Iterables

- Python 3 returns iterables in many places where Python 2 returns lists
- This includes
  `dict.items()  dict.keys()  dict.values(), filter(), map(),  range(),  zip()`