Partial functions

Python 3.x 3.0–3.6 , Python 2.x 2.7

As you probably know if you came from OOP school, specializing an abstract class and use it is a practice you should keep in mind when writing your code.

What if you could define an abstract function and specialize it in order to create different versions of it? Thinks it as a sort of *function Inheritance* where you bind specific params to make them reliable for a specific scenario.

Examples

Raise the power

Let's suppose we want raise *x* to a number *y* .

You'd write this as:

```
def raise_power(x, y):
    return x**y
```

What if your *y* value can assume a finite set of values?

Let's suppose *y* can be one of *[3,4,5]* and let's say you don't want offer end user the possibility to use such function since it is very computationally intensive. In fact you would check if provided *y* assumes a valid value and rewrite your function as:

```
def raise(x, y):
    if y in (3,4,5):
        return x**y
    raise NumberNotInRangeException("You should provide a valid exponent")
```

Messy? Let's use the abstract form and specialize it to all three cases: let's implement them **partially** .

```
from functors import partial
raise_to_three = partial(raise, y=3)
raise_to_four = partial(raise, y=4)
raise_to_five = partial(raise, y=5)
```

What happens here? We fixed the *y* params and we defined three different functions.

No need to use the abstract function defined above (you could make it *private*) but you could use **partial applied** functions to deal with raising a number to a fixed value.

Syntax

```
partial(function, **params_you_want_fix)
```

Parameters

Param	details
x	the number to be raised
y	the exponent
raise	the function to be specialized

Remarks

As stated in Python doc the *functools.partial* :

> Return a new partial object which when called will behave like func called with the positional arguments args and keyword arguments keywords. If more arguments are supplied to the call, they are appended to args. If additional keyword arguments are supplied, they extend and override keywords.

Check this link to see how *partial* can be implemented.