Alternatives to switch statement from other languages All Versions

Examples

Use a dict of functions

Another straightforward way to go is to create a dictionary of functions:

```
switch = {
    1: lambda: 'one',
    2: lambda: 'two',
    42: lambda: 'the answer of life the universe and everything',
}
```

then you add a default function:

```
def default_case():
    raise Exception('No case found!')
```

and you use the dictionary's get method to get the function given the value to check and run it. If value does not exists in dictionary, then `default_case` is run.

```
>>> switch.get(1, default_case)()
one
>>> switch.get(2, default_case)()
two
>>> switch.get(3, default_case)()
…
Exception: No case found!
>>> switch.get(42, default_case)()
the answer of life the universe and everything
```

you can also make some syntactic sugar so the switch looks nicer:

```
def run_switch(value):
    return switch.get(value, default_case)()

>>> run_switch(1)
one
```

Use class introspection

You can use a class to mimic the switch/case structure. The following is using introspection of a class (using the `getattr()` function that resolves a string into a bound method on an instance) to resolve the "case" part.

Then that introspecting method is aliased to the `__call__` method to overload the `()` operator.

```
class SwitchBase:
    def switch(self, case):
        m = getattr(self, 'case_{}'.format(case), None)
        if not m:
            return self.default
        return m

    __call__ = switch
```

Then to make it look nicer, we subclass the `SwitchBase` class (but it could be done in one class), and there we define all the `case` as methods:

```
class CustomSwitcher:
    def case_1(self):
        return 'one'

    def case_2(self):
        return 'two'

    def case_42(self):
        return 'the answer of life, the universe and everything!'

    def default(self):
        raise Exception('Not a case!')
```

so then we can finally use it:

```
>>> switch = CustomSwitcher()
>>> print(switch(1))
```

```
>>> print(switch(1))
one
>>> print(switch(2))
two
>>> print(switch(3))
…
Exception: Not a case!
>>> print(switch(42))
the answer of life, the universe and everything!
```

Use what the language offers: the if/else construct.

Well, if you want a `switch` / `case` construct, the most straightforward way to go is to use the good old `if` / `else` construct:

```
def switch(value):
    if value == 1:
        return "one"
    if x == 2:
        return "two"
    if x == 42:
        return "the answer to the question about life, the universe and everything"
    raise Exception("No case found!")
```

it might look redundant, and not always pretty, but that's by far the most efficient way to go, and it does the job:

```
>>> switch(1)
one
>>> switch(2)
two
>>> switch(3)
…
Exception: No case found!
>>> switch(42)
the answer to the question about life the universe and everything
```

Using a context manager

Another way, which is very readable and elegant, but far less efficient than a if/else structure, is to build a class such as follows, that will read and store the value to compare with, expose itself within the context as a callable that will return true if it matches the stored value:

```
class Switch:
    def __init__(self, value):
        self._val = value
    def __enter__(self):
        return self
    def __exit__(self, type, value, traceback):
        return False # Allows traceback to occur
    def __call__(self, cond, *mconds):
        return self._val in (cond,)+mconds
```

then defining the cases is almost a match to the real `switch` / `case` construct (exposed within a function below, to make it easier to show off):

```
def run_switch(value):
    with Switch(value) as case:
        if case(1):
            return 'one'
        if case(2):
            return 'two'
        if case(3):
            return 'the answer to the question about life, the universe and everything'
        # default
        raise Exception('Not a case!')
```

So the execution would be:

```
>>> run_switch(1)
one
>>> run_switch(2)
two
>>> run_switch(3)
…
Exception: Not a case!
>>> run_switch(42)
the answer to the question about life, the universe and everything
```

Nota Bene :

- This solution is being offered as the *switch* module available on pypi .

Syntax

Parameters

Remarks

There is *NO* switch statement in python as a language design choice. There has been a PEP (PEP-3103) covering the topic that has been rejected.

You can find many list of recipes on how to do your own switch statements in python, and here I'm trying to suggest the most sensible options. Here are a few places to check:

- Replacements for switch statement in Python?
- http://code.activestate.com/recipes/269708-some-python-style-switches/
- http://code.activestate.com/recipes/410692-readable-switch-construction-without-lambdas-or-di/
- …