Idioms All Versions

Examples

Dictionary key initializations

Prefer dict.get method if you are not sure if the key is present. It allows you to return a default value if key is not found. The traditional method dict[key] would raise a KeyError exception.

Rather than doing

```
def add_student():
    try:
        students['count'] += 1
    except KeyError:
        students['count'] = 1
```

Do

```
def add_student():
        students['count'] = students.get('count', 0) + 1
```

Switching variables

To switch the value of two variables you can use tuple unpacking.

```
x = True
y = False
x, y = y, x
x
# False
y
# True
```

Test for "__main__" to avoid unexpected code execution

It is good practice to test the calling program's __name__ variable before executing your code.

```
import sys

def main():
    # Your code starts here

    # Don't forget to provide a return code
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

Using this pattern ensures that your code is only executed when you expect it to be; for example, when you run your file explicitly:

```
python my_program.py
```

The benefit, however, comes if you decide to import your file in another program (for example if you are writing it as part of a library). You can then import your file, and the __main__ trap will ensure that no code is executed unexpectedly:

```
# A new program file
import my_program        # main() is not run

# But you can run main() explicitly if you really want it to run:
my_program.main()
```

Use truth value testing

Python will implicitly convert any object to a Boolean value for testing, so use it wherever possible.

```
# Good examples, using implicit truth testing
if attr:
    # do something

if not attr:
    # do something

# Bad examples, using specific types
if attr == 1:
    # do something

if attr == True:
    # do something

if attr != '':
    # do something

# If you are looking to specifically check for None, use 'is' or 'is not'
if attr is None:
    # do something
```

This generally produces more readable code, and is usually much safer when dealing with unexpected types.

Click here for a list of what will be evaluated to `False` .

Syntax

Parameters

Remarks