AIAP Tooling

Go to www.menti.com
and use the code 47 16 90

Contents

PEP8

Documentation: https://pep8.org/

Main Points

1. Consistency
2. Be like English

"It's just a guide"

-Me

Introduction

This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing style guidelines for the C code in the C implementation of Python. This document and PEP 257 (Docstring Conventions) were adapted from Guido’s original Python Style Guide essay, with some additions from Barry’s style guide. This style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself. Many projects have their own coding style guidelines. In the event of any conflicts, such project-specific guides take precedence for that project.

Introduction

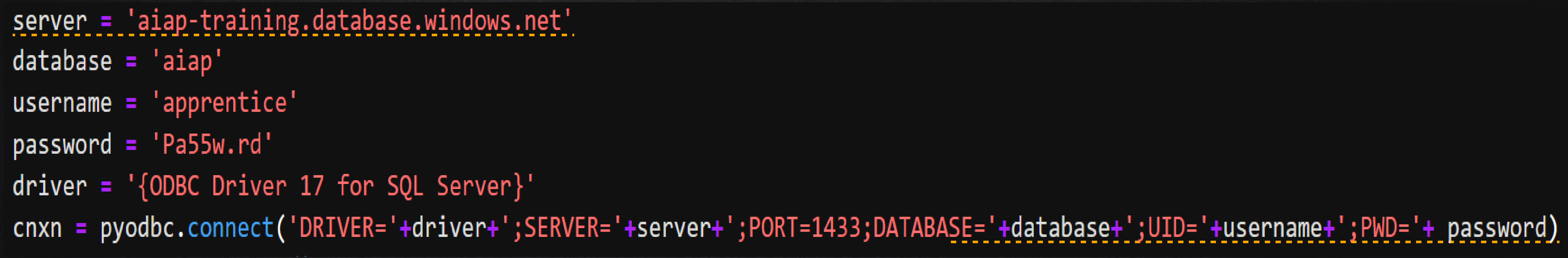
This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing style guidelines for the C code in the C implementation of Python.

This document and PEP 257 (Docstring Conventions) were adapted from Guido’s original Python Style Guide essay, with some additions from Barry’s style guide.

This style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself.

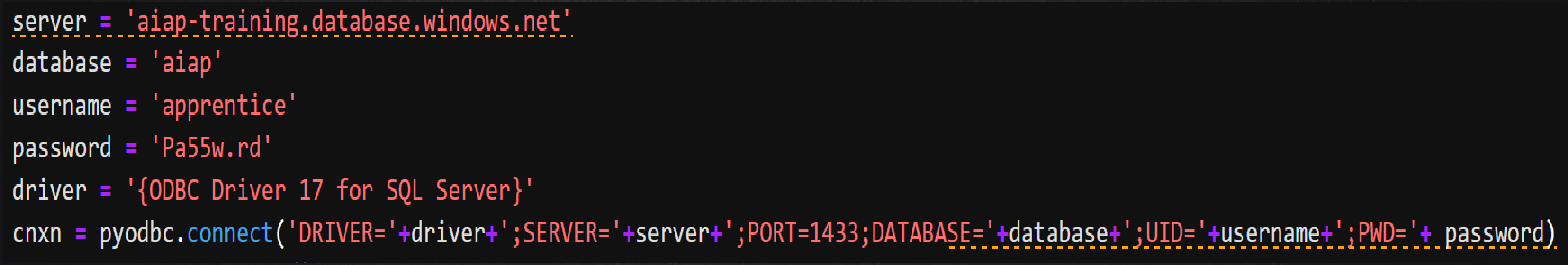
Many projects have their own coding style guidelines. In the event of any conflicts, such project-specific guides take precedence for that project.

Line Length

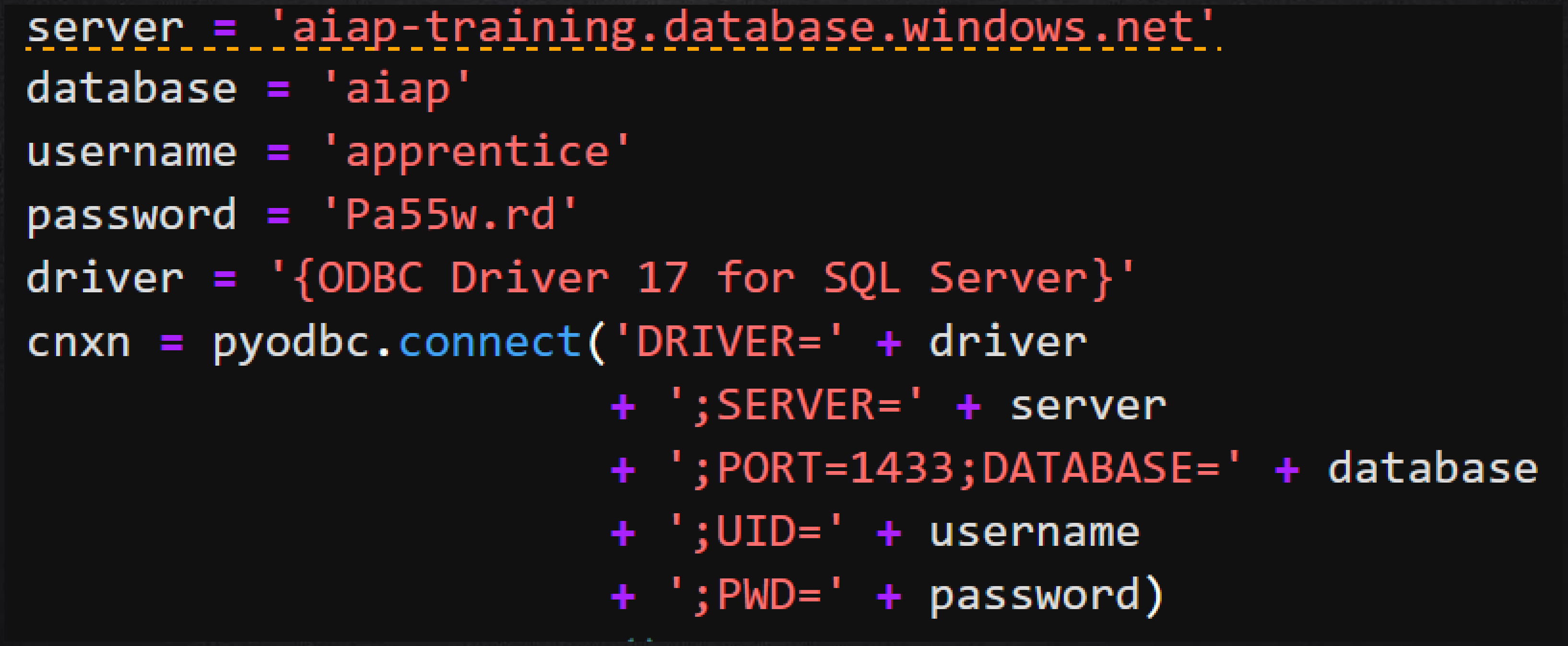


```
server = 'aiap-training.database.windows.net'
database = 'aiap'
username = 'apprentice'
password = 'Pa55w.rd'
driver = '{ODBC Driver 17 for SQL Server}'
cnxn = pyodbc.connect('DRIVER='+driver+';SERVER='+server+';PORT=1433;DATABASE='+database+';UID='+username+';PWD='+ password)
```

Line Length

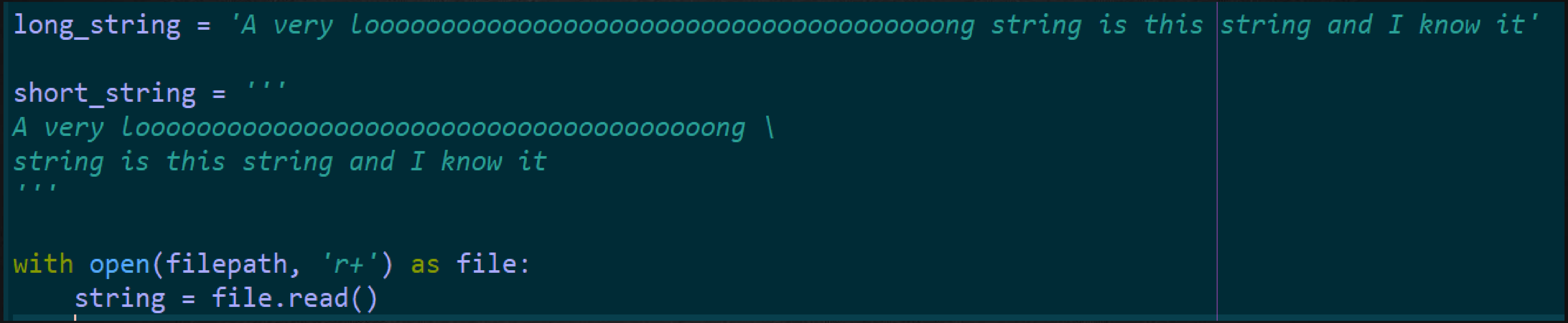


```python
server = 'aiap-training.database.windows.net'
database = 'aiap'
username = 'apprentice'
password = 'Pa55w.rd'
driver = '{ODBC Driver 17 for SQL Server}'
cnxn = pyodbc.connect('DRIVER='+driver+';SERVER='+server+';PORT=1433;DATABASE='+database+';UID='+username+';PWD='+ password)
```



```python
server = 'aiap-training.database.windows.net'
database = 'aiap'
username = 'apprentice'
password = 'Pa55w.rd'
driver = '{ODBC Driver 17 for SQL Server}'
cnxn = pyodbc.connect('DRIVER=' + driver
                      + ';SERVER=' + server
                      + ';PORT=1433;DATABASE=' + database
                      + ';UID=' + username
                      + ';PWD=' + password)
```

Line Length



```
long_string = 'A very loooooooooooooooooooooooooooooooooooooong string is this string and I know it'

short_string = '''
A very loooooooooooooooooooooooooooooooooooooong \
string is this string and I know it
'''


with open(filepath, 'r+') as file:
    string = file.read()
```

Function Writing

```python
def show_all_null_values(df):
    return df[df.isnull().any(1)]
```

Function Writing

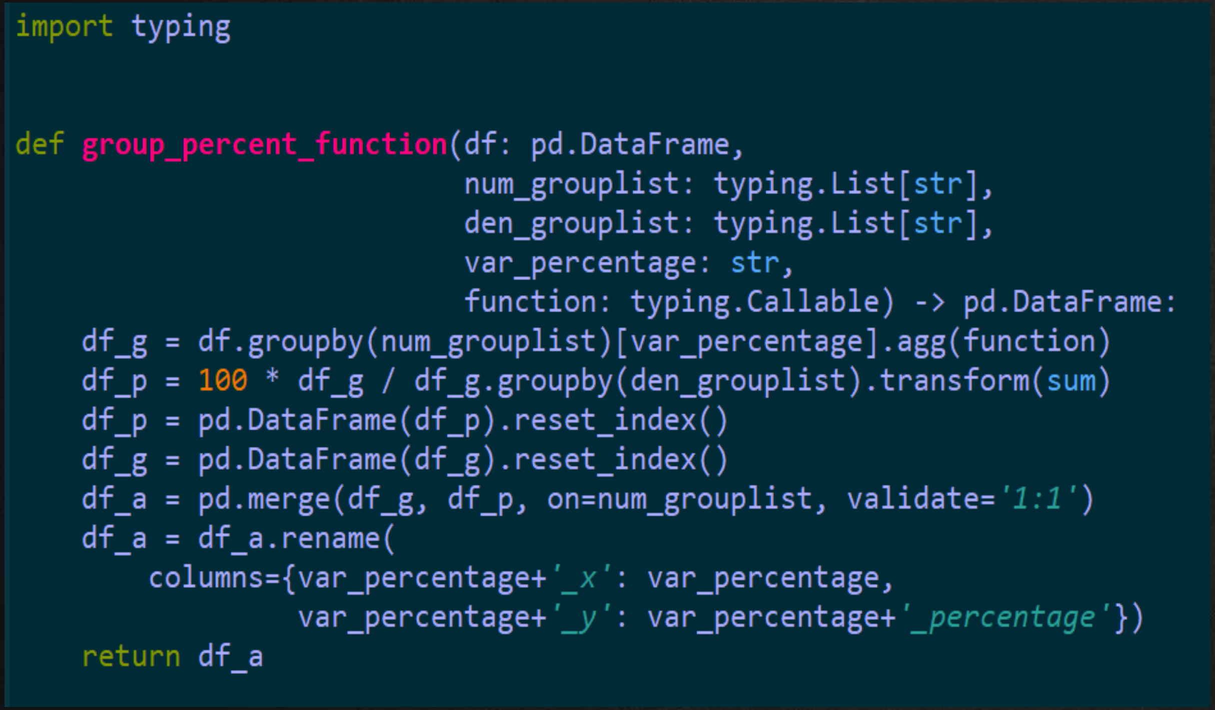
```
def show_all_null_values(df):
    return df[df.isnull().any(1)]
```

```
def show_all_null_values_from_df(df):
    '''Pass in DataFrame to display all rows with null values in them.
    '''
    return df[df.isnull().any(1)]
```

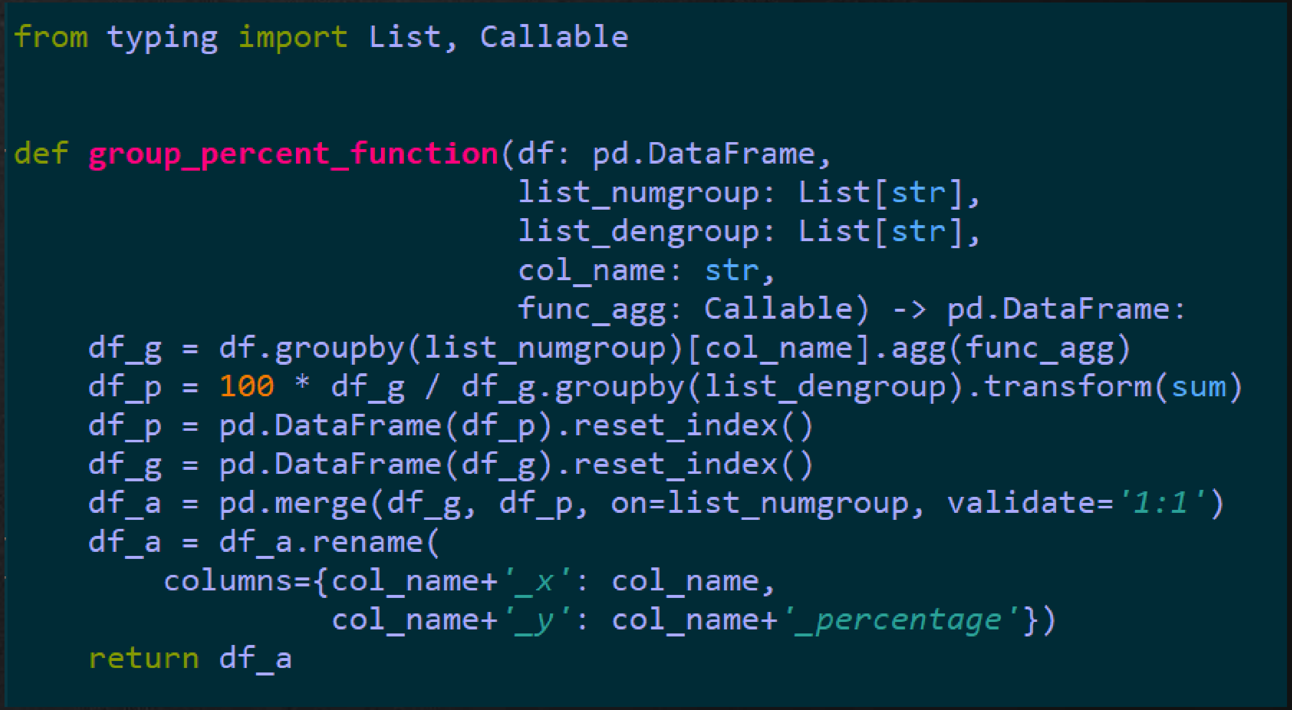
Function Writing

```python
def show_all_null_values_from_df(df: pd.DataFrame) -> pd.DataFrame:
    '''Pass in DataFrame to display all rows with null values in them.
    ----------
    Parameters:
    df: pandas.DataFrame
    ----------
    Returns DataFrame
    ----------
    Example
    Input:
    >>> df = pd.DataFrame({'Mick': [np.nan, 2, 3], 'Hick': [5, 7, np.nan]})
    >>> df_ans = show_all_null_values_from_df(df)
    >>> print(df_ans)
    Output:
       Mick  Hick
    0   NaN   5.0
    2   3.0   NaN
    '''
    return df[df.isnull().any(1)]
```

Function Writing

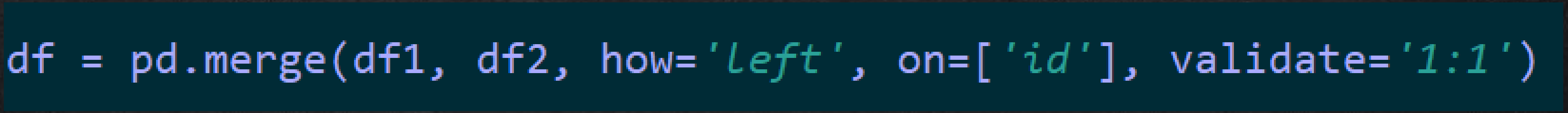


```python
import typing


def group_percent_function(df: pd.DataFrame,
                           num_grouplist: typing.List[str],
                           den_grouplist: typing.List[str],
                           var_percentage: str,
                           function: typing.Callable) -> pd.DataFrame:
    df_g = df.groupby(num_grouplist)[var_percentage].agg(function)
    df_p = 100 * df_g / df_g.groupby(den_grouplist).transform(sum)
    df_p = pd.DataFrame(df_p).reset_index()
    df_g = pd.DataFrame(df_g).reset_index()
    df_a = pd.merge(df_g, df_p, on=num_grouplist, validate='1:1')
    df_a = df_a.rename(
        columns={var_percentage+'_x': var_percentage,
                 var_percentage+'_y': var_percentage+'_percentage'})
    return df_a
```



```python
from typing import List, Callable


def group_percent_function(df: pd.DataFrame,
                           list_numgroup: List[str],
                           list_dengroup: List[str],
                           col_name: str,
                           func_agg: Callable) -> pd.DataFrame:
    df_g = df.groupby(list_numgroup)[col_name].agg(func_agg)
    df_p = 100 * df_g / df_g.groupby(list_dengroup).transform(sum)
    df_p = pd.DataFrame(df_p).reset_index()
    df_g = pd.DataFrame(df_g).reset_index()
    df_a = pd.merge(df_g, df_p, on=list_numgroup, validate='1:1')
    df_a = df_a.rename(
        columns={col_name+'_x': col_name,
                 col_name+'_y': col_name+'_percentage'})
    return df_a
```

Dictionaries are good, use them



```
df = pd.merge(df1, df2, how='left', on=['id'], validate='1:1')
```

Dictionaries are good, use them

```python
df = pd.merge(df1, df2, how='left', on=['id'], validate='1:1')
```

```python
dict_config = {'left': df1,
               'right': df2,
               'how': 'left',
               'on': ['id'],
               'left_on': None,
               'validate': '1:1'}

df = pd.merge(**dict_config)
```

Dictionaries are good, use them

```python
df = pd.merge(df1, df2, how='left', on=['id'], validate='1:1')
```

```python
dict_config = {'left': df1,
               'right': df2,
               'how': 'left',
               'on': ['id'],
               'left_on': None,
               'validate': '1:1'}

df = pd.merge(**dict_config)
```

```python
import json

with open(r'config', 'r') as string:
    dict_config = json.load(string)

df = pd.merge(**dict_config)
```

What is good code?

Main Points

1. Consistency
2. Be like English

Consider:

- Wikipedia
- Your own essay/journal
- Revisits
- import this

Virtual Environments

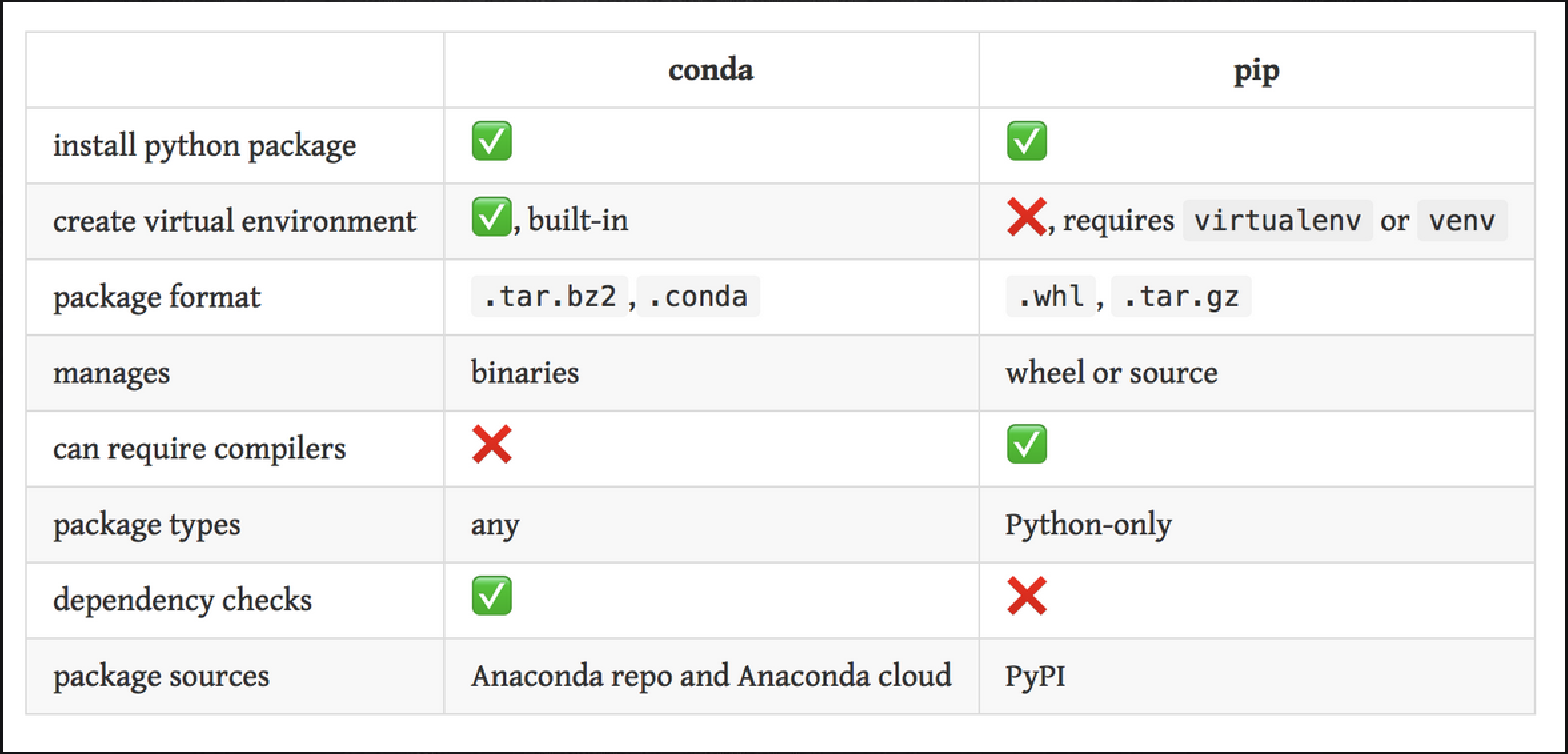
Conda vs pip

Smartphone vs Alarm Clock

Things to consider:

- Size
- Control (Deployment/Maintenance)

Virtual Environments



	conda	pip
install python package	✅	✅
create virtual environment	✅, built-in	❌, requires `virtualenv` or `venv`
package format	`.tar.bz2`, `.conda`	`.whl`, `.tar.gz`
manages	binaries	wheel or source
can require compilers	❌	✅
package types	any	Python-only
dependency checks	✅	❌
package sources	Anaconda repo and Anaconda cloud	PyPI

Virtual Environments

Simple tutorial on virtualenv:

```
pip install virtualenv -U
cd /directory
virtualenv /env_name

For Linux:
source /env_name/bin/activate
Or for Windows:
source /env_name/Scripts/activate

pip install -r requirements.txt
pip freeze -l > requirements.txt
deactivate
```

Jupyter Tooling

Jupyterlab

- Console
- Directory
- Tabs
- Extensions

```
pip install jupyterlab
```

Jupyter Tooling

To install Extensions, first install Node.js.

```
jupyter labextension install jupyterlab-drawio

pip install jupyterlab-git
```

Git Bash aliases

```
alias python='winpty python.exe'
alias jupyterlab='python -m jupyterlab & disown’
```

Other useful stuff

```
pip install spyder
pip install youtube-dl
pip install tagui
nbconvert
```

Slides