H_2O.ai

Numeric Data Transformations

```python
import math

def pseudo_log10(x):
    return math.asinh(x / 2) / math.log(10)

print("pseudo_log10(\261{}) = \261{:0.6f}".format(100000, pseudo_log10(100000)))
print("pseudo_log10(\261{})  = \261{:0.6f}".format(10000, pseudo_log10(10000)))
print("pseudo_log10(\261{})   = \261{:0.6f}".format(1000, pseudo_log10(1000)))
print("pseudo_log10(\261{})    = \261{:0.6f}".format(100, pseudo_log10(100)))
print("pseudo_log10(\261{})     = \261{:0.6f}".format(10, pseudo_log10(10)))
print("pseudo_log10(\261{})      = \261{:0.6f}".format(1, pseudo_log10(1)))
print("pseudo_log10({})       = {}".format(0, pseudo_log10(0)))
```

```
pseudo_log10(±100000) = ±5.000000
pseudo_log10(±10000)  = ±4.000000
pseudo_log10(±1000)   = ±3.000000
pseudo_log10(±100)    = ±2.000043
pseudo_log10(±10)     = ±1.004279
pseudo_log10(±1)      = ±0.208988
pseudo_log10(0)       = 0.0
```

Transformation for skewed data with positive and negative values

```
pseudoLog10(x) = asinh(x/2) / log(10)
```

Numeric Data Transformations

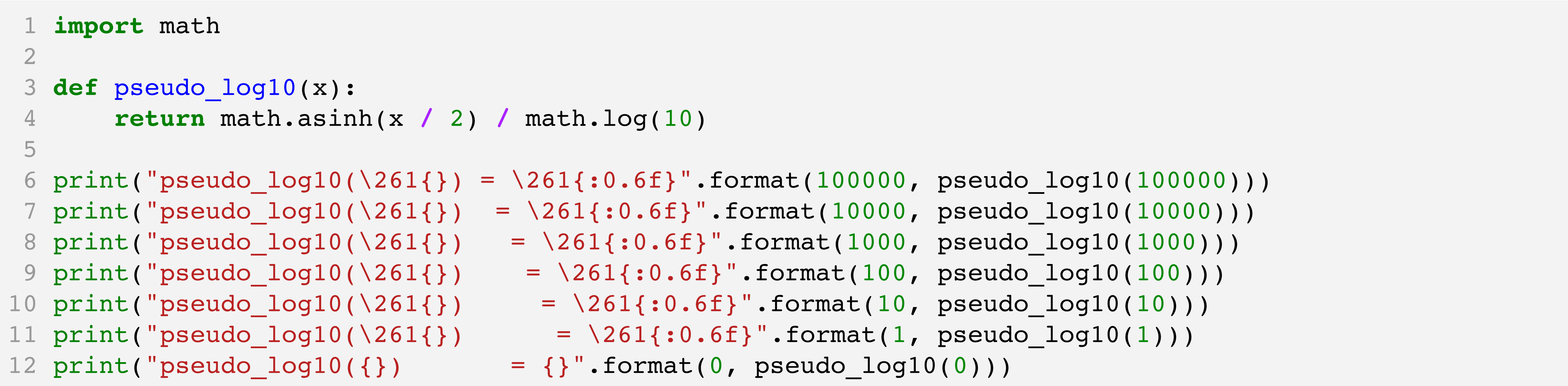
```python
import numpy as np
breaks = np.linspace(10, 90, 9).tolist()
census_data["age_group"] = census_data["age"].cut(breaks)

census_data["log1p_capital-gain"] = census_data["capital-gain"].log1p()
census_data["log1p_capital-loss"] = census_data["capital-loss"].log1p()
print(census_data["age_group"].table())
```

age_group	Count
(10.0,20.0]	2410
(20.0,30.0]	8162
(30.0,40.0]	8546
(40.0,50.0]	6983
(50.0,60.0]	4128
(60.0,70.0]	1792
(70.0,80.0]	441
(80.0,90.0]	99

Numeric Data Transformations

Transformation for skewed data with positive and negative values

```
pseudoLog10(x) = asinh(x/2) / log(10)
```



```python
import math

def pseudo_log10(x):
    return math.asinh(x / 2) / math.log(10)

print("pseudo_log10(\261{}) = \261{:0.6f}".format(100000, pseudo_log10(100000)))
print("pseudo_log10(\261{})  = \261{:0.6f}".format(10000, pseudo_log10(10000)))
print("pseudo_log10(\261{})   = \261{:0.6f}".format(1000, pseudo_log10(1000)))
print("pseudo_log10(\261{})    = \261{:0.6f}".format(100, pseudo_log10(100)))
print("pseudo_log10(\261{})     = \261{:0.6f}".format(10, pseudo_log10(10)))
print("pseudo_log10(\261{})      = \261{:0.6f}".format(1, pseudo_log10(1)))
print("pseudo_log10({})       = {}".format(0, pseudo_log10(0)))
```

```
pseudo_log10(±100000) = ±5.000000
pseudo_log10(±10000)  = ±4.000000
pseudo_log10(±1000)   = ±3.000000
pseudo_log10(±100)    = ±2.000043
pseudo_log10(±10)     = ±1.004279
pseudo_log10(±1)      = ±0.208988
pseudo_log10(0)       = 0.0
```