# Python Curriculum

Part 03 - Data Containers and Repetitions (1/3)

# Lists

```python
'''norse_shop.py'''
header = ['poi', 'revenue', 'cost', 'visits', 'unique_visitors']
row1 = ['Yggdrasil', 790.2, 477.85, 53, 7]
row2 = ['Valhalla', 1700.65, 1500, 11, 10]
```

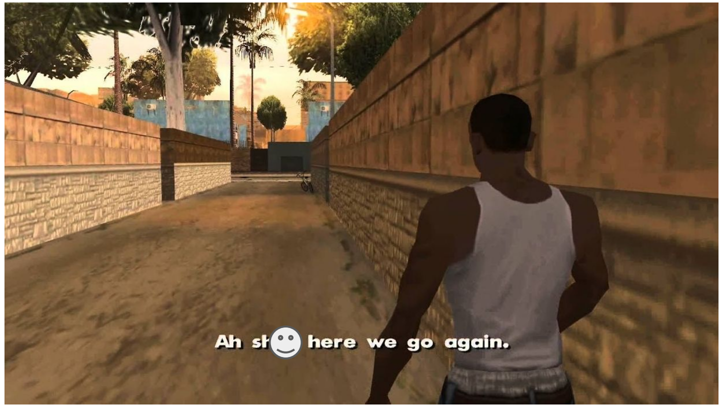
```python
'''norse_shop.py'''
# ...
csv_header = ','.join(header)
print(csv_header)
```

| poi | revenue | cost | visits | unique_visitors |
|---|---|---|---|---|
| | | | | |

```python
'''norse_shop.py'''
# ...
csv_row1 = ','.join(row1)
print(csv_row1)

csv_row2 = ','.join(row2)
print(csv_row2)
```

```
% python norse_shop.py
Traceback (most recent call last):
  File "norse_shop.py", line 9, in <module>
    csv_row1 = ','.join(row1)
TypeError: sequence item 1: expected str instance, float found
```



```python
'''norse_shop.py'''
# ...
row1[1] = str(row1[1])  # index 1 (second item)
row1[2] = str(row1[2])  # index 2 (third item)
row1[3] = str(row1[3])  # index 3 (fourth item)
row1[4] = str(row1[4])  # index 3 (fourth item)
csv_row1 = ','.join(row1)
print(csv_row1)
```

| poi | revenue | cost | visits | unique_visitors |
|---|---|---|---|---|
| Yggdrasil | 790.2 | 477.85 | 53 | 7 |

```
>>> s = 'Canada'
>>> s[0] = 'B'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> l = ['C', 'a', 'n', 'a', 'd', 'a']
>>> l[0] = 'B'
>>> l[-2] = 'n'
>>> l
['B', 'a', 'n', 'a', 'n', 'a']
>>> ''.join(l)
'Banana'
```

## For Loop

```python
'''norse_shop.py'''
# ...
row1[1] = str(row1[1])  # index 1 (second item)
row1[2] = str(row1[2])  # index 2 (third item)
row1[3] = str(row1[3])  # index 3 (fourth item)
row1[4] = str(row1[4])  # index 3 (fourth item)
csv_row1 = ','.join(row1)
print(csv_row1)
```

```python
'''norse_shop.py'''
# ...
for i in range(len(row1)):
    if type(row1[i]) is not str:
        row1[i] = str(row1[i])

csv_row1 = ','.join(row1)
print(csv_row1)
```

```python
'''norse_shop.py'''
# ...
def mutate_row(row):
    for i in range(len(row)):
        row[i] = str(row[i])

for row in [row1, row2]:
    mutate_row(row)
    csv_row = ','.join(row)
    print(csv_row)
```

| poi | revenue | cost | visits | unique_visitors |
|---|---|---|---|---|
| Yggdrasil | 790.2 | 477.85 | 53 | 7 |
| Valhalla | 1700.65 | 1500 | 11 | 10 |

# Mutations

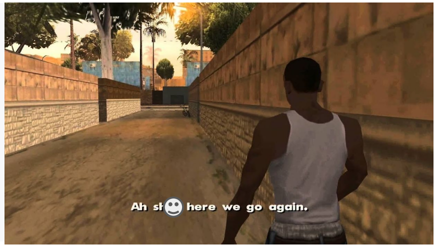
beware

```python
'''norse_shop.py'''
# ...
# add profit header
header.append('profit')

csv_header = ','.join(header)
print(csv_header)

for row in [row1, row2]:
    mutate_row(row)
    csv_row = ','.join(row)
    # compute profit for each row and concatenate to the csv_row
    profit = row[1] - row[2]
    # another way to concatenate strings
    csv_row = ','.join([csv_row, str(profit)])
    print(csv_row)
```

```
% python norse_shop.py
Traceback (most recent call last):
  File "norse_shop.py", line 17, in <module>
    profit = row[1] - row[2]
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```



```python
'''norse_shop.py'''
# ...
def convert_row(row):
    new_row = []

    for i in range(len(row)):
        new_row.append(str(row[i]))

    return new_row

for row in [row1, row2]:
    new_row = convert_row(row)
    csv_row = ','.join(new_row)
    # compute profit for each row and concatenate to the csv_row
    profit = row[1] - row[2]
    # another way to concatenate strings
    csv_row = ','.join([csv_row, str(profit)])
    print(csv_row)
```

| poi | revenue | cost | visits | unique_visitors | profit |
|---|---|---|---|---|---|
| Yggdrasil | 790.2 | 477.85 | 53 | 7 | 312.35 |
| Valhalla | 1700.65 | 1500 | 11 | 10 | 200.6500000000001 |

```python
# users have a flexible choice with an immutable approach
new_row1 = convert_row(row1)  # assign anew
row1 = convert_row(row1)  # override the original to emulate mutation if desired

# workaround with a mutable approach
# basically re-implement `convert_row()` itself
new_row1 = []

for i in range(len(row1)):
    new_row1.append(row1[i])

mutate_row(new_row1)  # new_row1 is now mutated
```

# Questions?

## Extra - List Mechanisms

## Shallow Copy

```python
'''norse_shop.py'''
# ...
def convert_row(row):
    new_row = []

    for i in range(len(row)):
        new_row.append(str(row[i]))

    return new_row
```

```python
def convert_copy_row(row):
    new_row = row.copy()

    for i in range(len(new_row)):
        new_row[i] = str(new_row[i])

    return new_row
```

```python
a = ['a', [1, 2, 3]]
b = a.copy()
# mutation tests
b[0] = 'b'
assert b[0] == 'b'
assert a[0] == 'a'  # list a still intact
b[1][0] = 10
assert b[1][0] == 10
assert a[1][0] == 1  # would raise AssertionError
```

```
Traceback (most recent call last):
  ...
    assert a[1][0] == 1
AssertionError
```

```python
a = ['a', [1, 2, 3]]
# custom deeper copy
b = []  # outer new list
for i in range(len(a)):
    if type(a[i]) is list:
        inner = []  # inner new list
        for ii in range(len(a[i])):
            inner.append(a[i][ii])  # make "deeper" of the nested items
        b.append(inner)
    else:
        b.append(a[i])
# mutation tests
b[0] = 'b'
assert b[0] == 'b'
assert a[0] == 'a'  # list a still intact
b[1][0] = 10
assert b[1][0] == 10
assert a[1][0] == 1
```

```python
a = ['a', 1, 2, 3]
b = a.copy()
# mutation tests
b[0] = 'b'
assert b[0] == 'b'
assert a[0] == 'a'  # list a still intact
b[2] = 10
assert b[2] == 10
assert a[2] == 1
```

“Flat is better than nested”

## Comprehensions

```python
a = ['a', 1, 2, 3]
# copy `a` through list comprehension
b = [v for v in a]
```

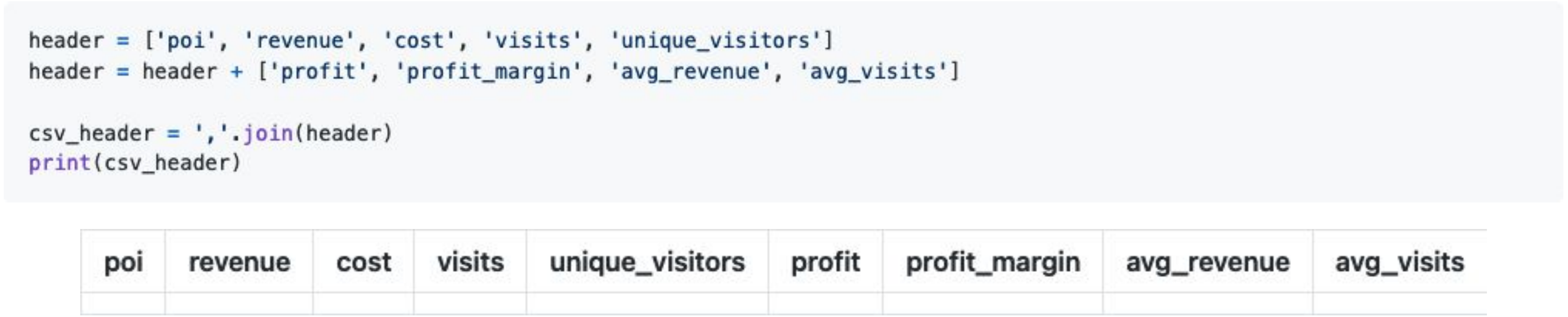
```python
'''norse_shop.py'''
header = ['poi', 'revenue', 'cost', 'visits', 'unique_visitors']
row1 = ['Yggdrasil', 790.2, 477.85, 53, 7]
row2 = ['Valhalla', 1700.65, 1500, 11, 10]

header.append('profit')
csv_header = ','.join(header)
print(csv_header)

def get_profit(row):
    return row[1] - row[2]

for row in [row1, row2]:
    # list comprehension to replace `convert_row()`
    new_row = [str(v) for v in row]
    # compute profit
    profit = get_profit(row)
    new_row.append(str(profit))
    # transform to CSV string and print out
    csv_row = ','.join(new_row)
    print(csv_row)
```

## Concatenations



```
header = ['poi', 'revenue', 'cost', 'visits', 'unique_visitors']
header = header + ['profit', 'profit_margin', 'avg_revenue', 'avg_visits']

csv_header = ','.join(header)
print(csv_header)
```

| poi | revenue | cost | visits | unique_visitors | profit | profit_margin | avg_revenue | avg_visits |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |