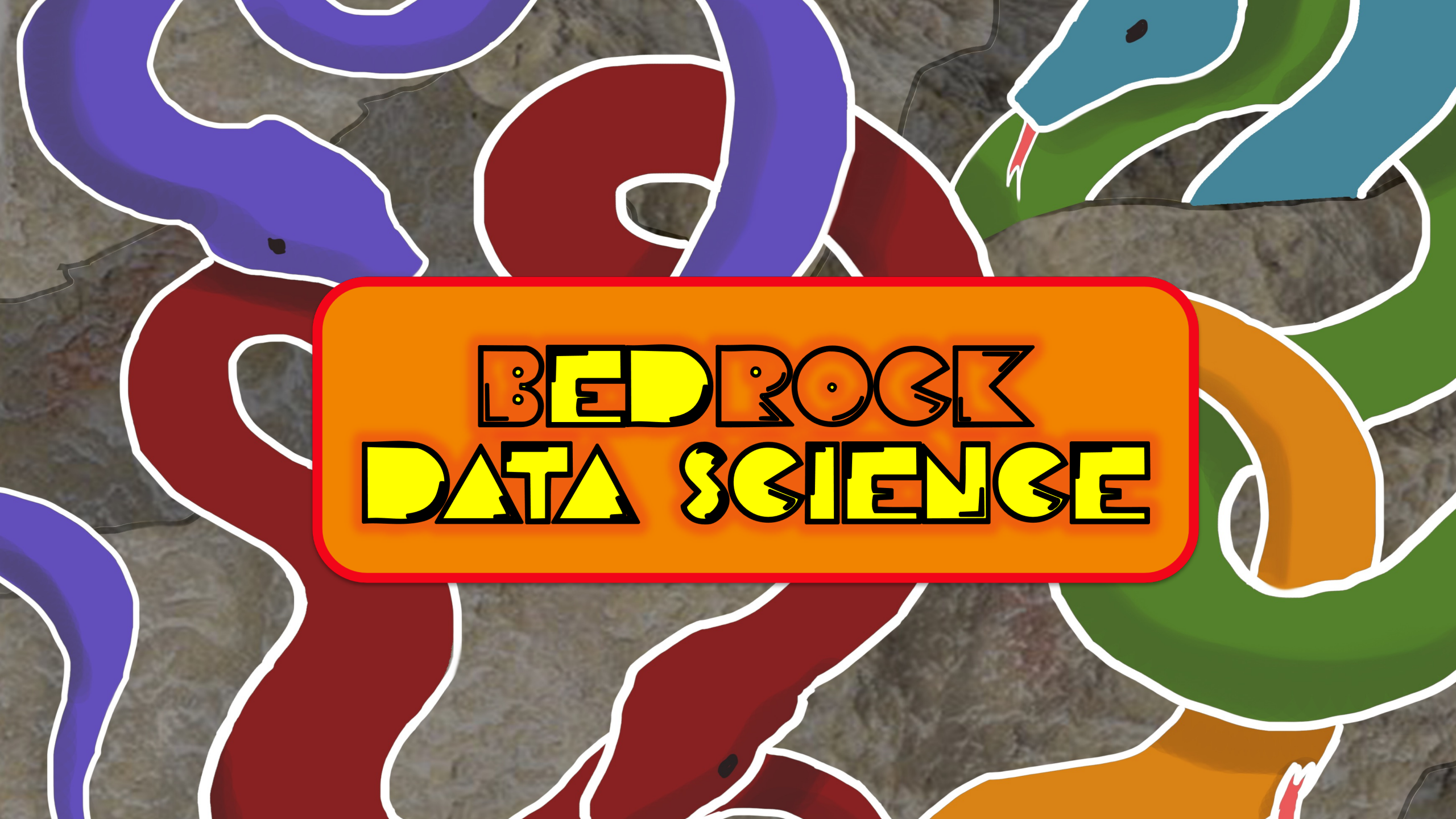

BEDROCK
DATA SCIENCE

BRDS

GAME TIME

READY?

```
In [2]: list_1 = [1,2,3,4,5]
   ...: list_2 = list_1
   ...: list_1[0] = 100
```

Options

A. The operation is invalid because list_1 is immutable

B. id(list_1) == id(list_2)

C. list_2 == [1,2,3,4,5]

D. list_1 == [1,2,3,4,5]

```
In [2]: list_1 = [1,2,3,4,5]
   ...: list_2 = list_1
   ...: list_1[0] = 100
```

Options

A.The operation is invalid because list_1 is immutable

B.id(list_1) == id(list_2)

C.list_2 == [1,2,3,4,5]

D.list_1 == [1,2,3,4,5]

```
In [2]: 1 & 2
>> ?
```

Options

A.0

B.1

C.2

D.Error TypeError: unsupported operand type(s)

```
In [2]: 1 & 2
>> ?
```

Options

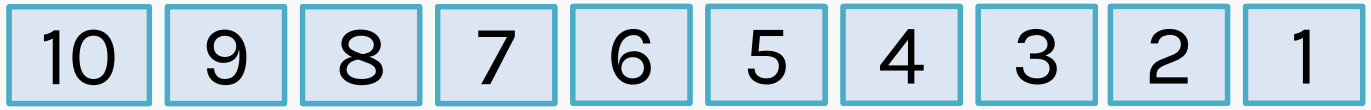
A.0

B.1

C.2

D.Error TypeError: unsupported operand type(s)

harlist



1,2,3,4,5,6,7,8,9,10

Options

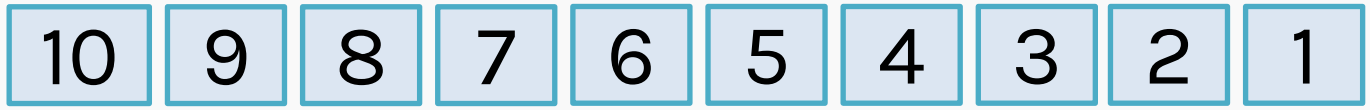
A. harlist[0:10:-1]

B. harlist[::-1]

C. harlist[-1::]

D. harlist[:-1:-1]

harlist



Options

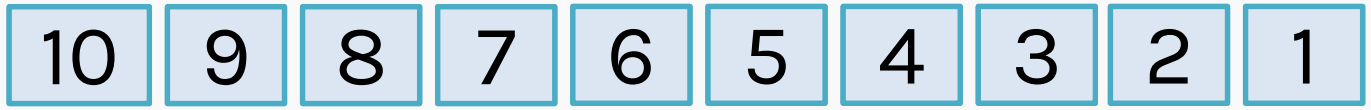
A. `harlist[0:10:-1]`

B. `harlist[::-1]`

C. `harlist[-1::]`

D. `harlist[:-1:-1]`

harlist



2,4,6,8,10

Options

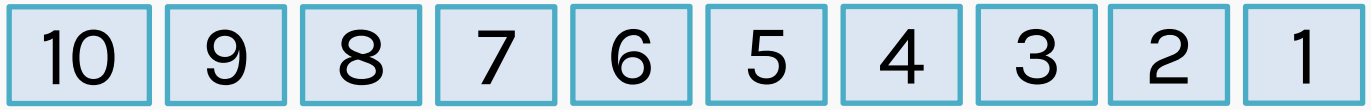
A. harlist[0:10:-2]

B. harlist[::-2]

C. harlist[-1::-2]

D. harlist[-2::-2]

harlist



2,4,6,8,10

Options

A. `harlist[0:10:-2]`

B. `harlist[::-2]`

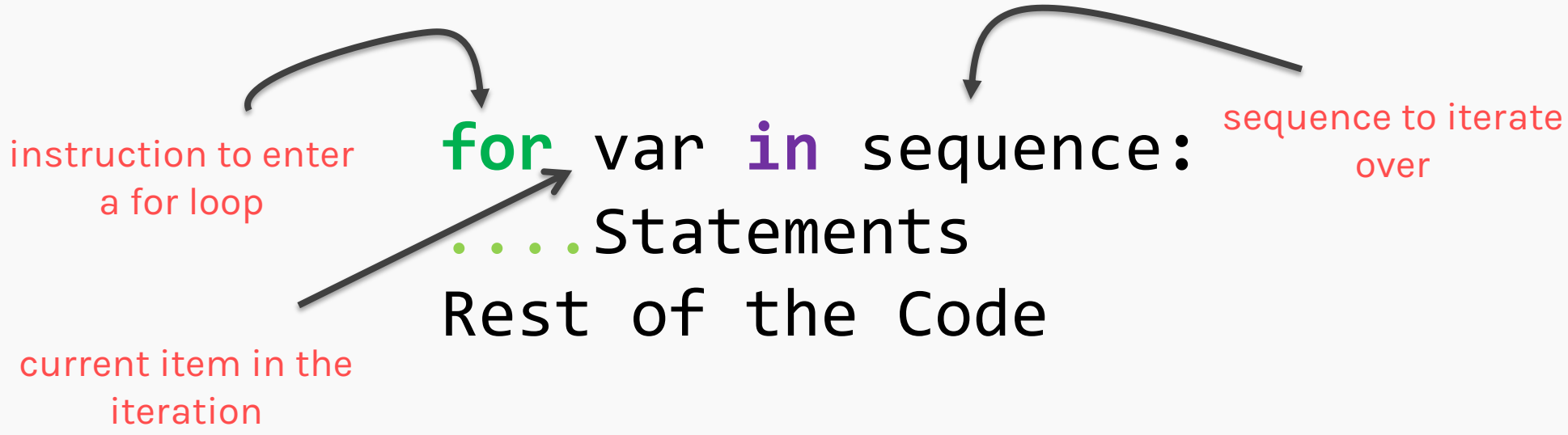
C. `harlist[-1::-2]`

D. `harlist[-2::-2]`

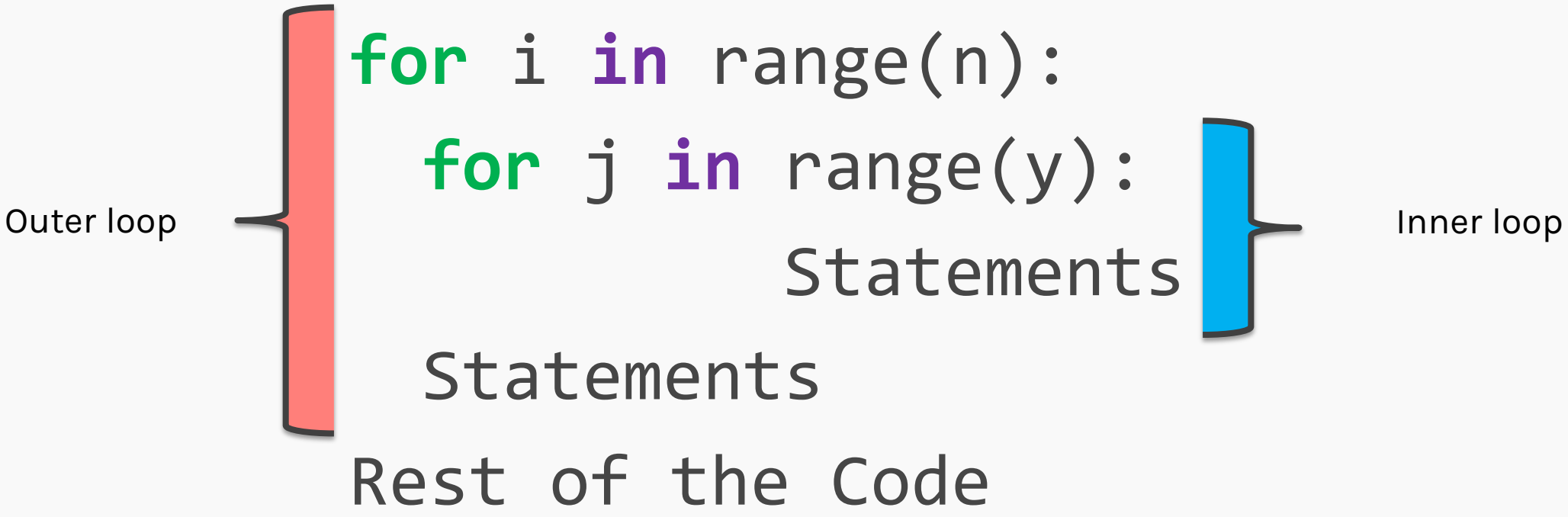
Loops

For loop

The for loop is useful to iterate over the elements of sequence such as string, list, tuple, range(), etc.

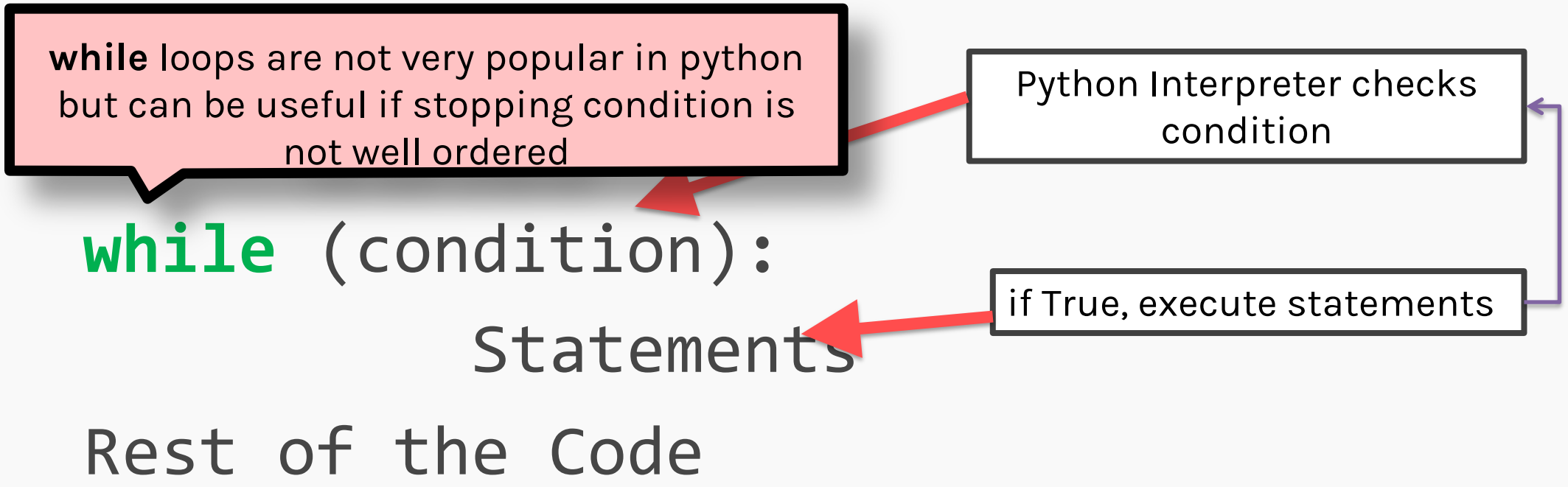


Nested for loops



While loop

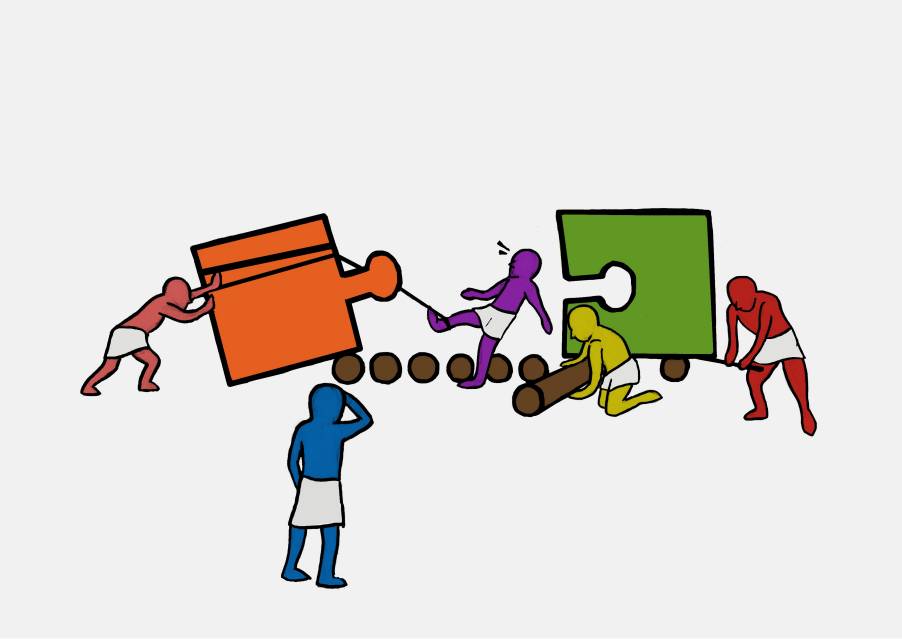
The while loop keeps repeating an action until an associated condition returns false



Infinite Loops

```
while True:
    print('With great power
    comes great responsibility')
```

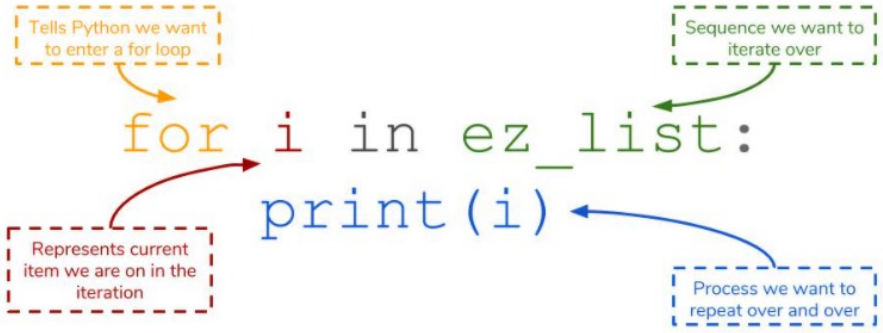
Exercise #3



Description

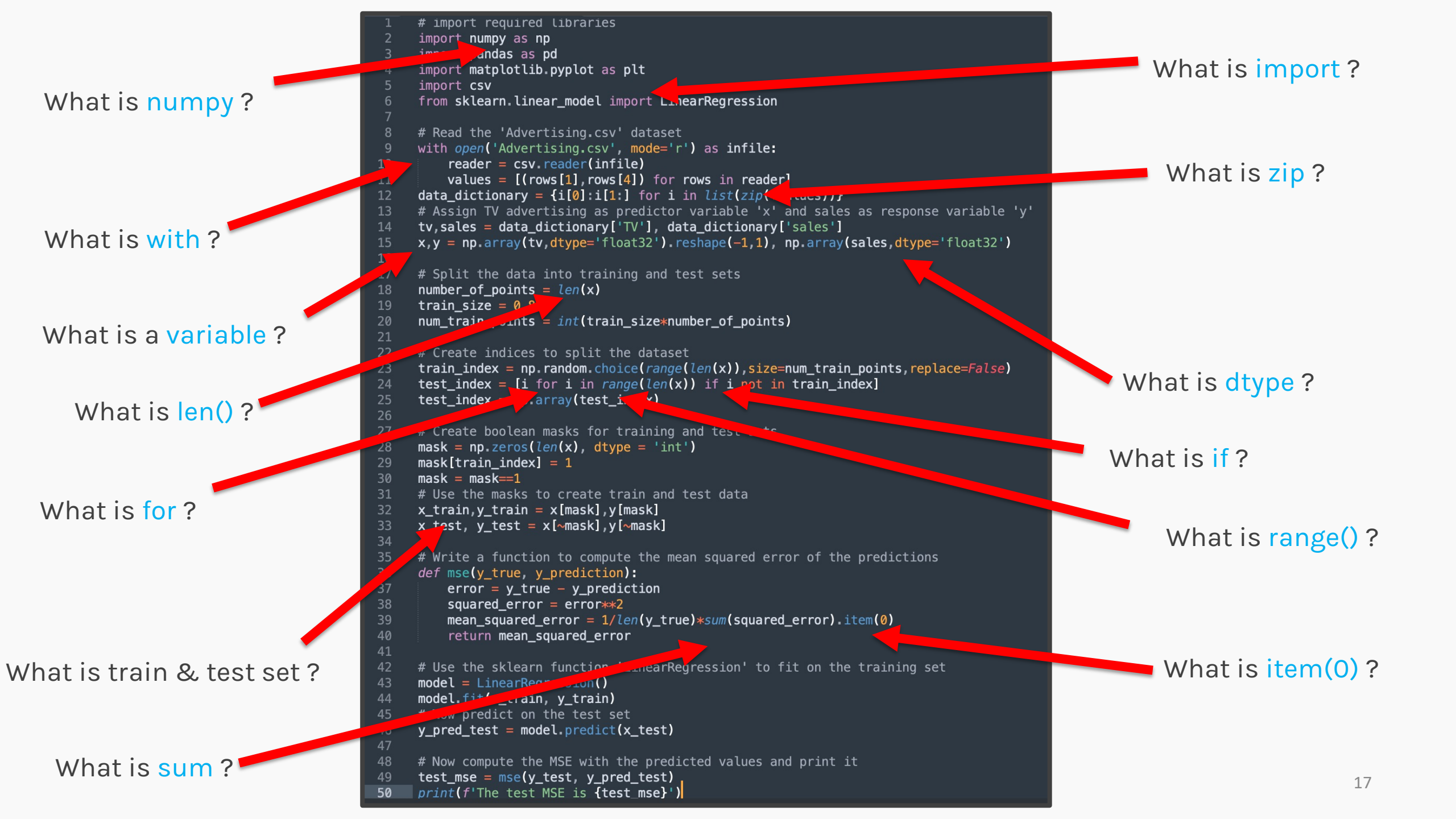
Exercise: Loops

The goal of this exercise is to learn to use `for` loops to manipulate lists.



Instructions

1. Like the previous exercise, make a new list of *even* & *odd numbers* but use loops instead.
2. Make a new list of *primes* from the given list.
3. Take a given list of lists which represents a chess board. Flip it by *90, 180 & 270 degrees* using for loops.




```
# import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv
from sklearn.linear_model import LinearRegression

# Read the 'Advertising.csv' dataset
with open('Advertising.csv', mode='r') as infile:
    reader = csv.reader(infile)
    values = [(rows[1],rows[4]) for rows in reader]
data_dictionary = {i[0]:i[1:] for i in list(zip(*values))}
# Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
tv,sales = data_dictionary['TV'], data_dictionary['sales']
x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')

# Split the data into training and test sets
number_of_points = len(x)
train_size = 0.8
num_train_points = int(train_size*number_of_points)

# Create indices to split the dataset
train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
test_index = [i for i in range(len(x)) if i not in train_index]
test_index = np.array(test_index)

# Create boolean masks for training and test sets
mask = np.zeros(len(x), dtype = 'int')
mask[train_index] = 1
mask = mask==1
# Use the masks to create train and test data
x_train,y_train = x[mask],y[mask]
x_test, y_test = x[~mask],y[~mask]

# Write a function to compute the mean squared error of the predictions
def mse(y_true, y_prediction):
    error = y_true - y_prediction
    squared_error = error**2
    mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
    return mean_squared_error

# Use the sklearn function 'LinearRegression' to fit on the training set
model = LinearRegression()
model.fit(x_train, y_train)
# Now predict on the test set
y_pred_test = model.predict(x_test)

# Now compute the MSE with the predicted values and print it
test_mse = mse(y_test, y_pred_test)
print(f'The test MSE is {test_mse}')
```

What is numpy ?

What is import ?

What is with ?

What is zip ?

~~What is a variable ?~~

~~What is dtype ?~~

~~What is len() ?~~

~~What is if ?~~

~~What is for ?~~

~~What is range() ?~~

What is train & test set ?

What is item(0) ?

~~What is sum ?~~

```python
# import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv
from sklearn.linear_model import LinearRegression

# Read the 'Advertising.csv' dataset
with open('Advertising.csv', mode='r') as infile:
    reader = csv.reader(infile)
    values = [(rows[1],rows[4]) for rows in reader]
data_dictionary = {i[0]:i[1:] for i in list(zip(*values))}
# Assign TV advertising as predictor variable 'x' and sales as response variable 'y'
tv,sales = data_dictionary['TV'], data_dictionary['sales']
x,y = np.array(tv,dtype='float32').reshape(-1,1), np.array(sales,dtype='float32')

# Split the data into training and test sets
number_of_points = len(x)
train_size = 0.8
num_train_points = int(train_size*number_of_points)

# Create indices to split the dataset
train_index = np.random.choice(range(len(x)),size=num_train_points,replace=False)
test_index = [i for i in range(len(x)) if i not in train_index]
test_index = np.array(test_index)

# Create boolean masks for training and test sets
mask = np.zeros(len(x), dtype = 'int')
mask[train_index] = 1
mask = mask==1
# Use the masks to create train and test data
x_train,y_train = x[mask],y[mask]
x_test, y_test = x[~mask],y[~mask]

# Write a function to compute the mean squared error of the predictions
def mse(y_true, y_prediction):
    error = y_true - y_prediction
    squared_error = error**2
    mean_squared_error = 1/len(y_true)*sum(squared_error).item(0)
    return mean_squared_error

# Use the sklearn function 'LinearRegression' to fit on the training set
model = LinearRegression()
model.fit(x_train, y_train)
# Now predict on the test set
y_pred_test = model.predict(x_test)

# Now compute the MSE with the predicted values and print it
test_mse = mse(y_test, y_pred_test)
print(f'The test MSE is {test_mse}')
```