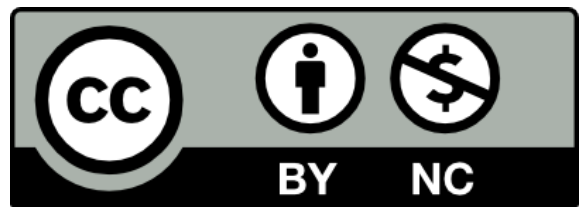
Business 4720 - Class 6

Data Management in Python using Pandas

Joerg Evermann

Faculty of Business Administration
Memorial University of Newfoundland
jevermann@mun.ca



MEMORIAL UNIVERSITY

This Class

What You Will Learn:

- Introduction to Python
- Introduction to the the Numpy package
- Introduction to the Pandas package

MEMORIAL UNIVERSITY

What is Python?

- Readability and simplicity
- Dynamic typing enhancing flexibility
- Extensive libraries
- Procedural, object-oriented, and functional programming
- Widely used in data analysis, AI, scientific computing, etc.
- Easy to learn
- Active community support

Intro Tutorial:

`https://python.swaroopch.com/`
`https://github.com/swaroopch/byte-of-python/releases/`

MEMORIAL UNIVERSITY

Running Python

1. Interactive Python Shell (command line)
2. Jupyter Notebooks
3. PyCharm IDE

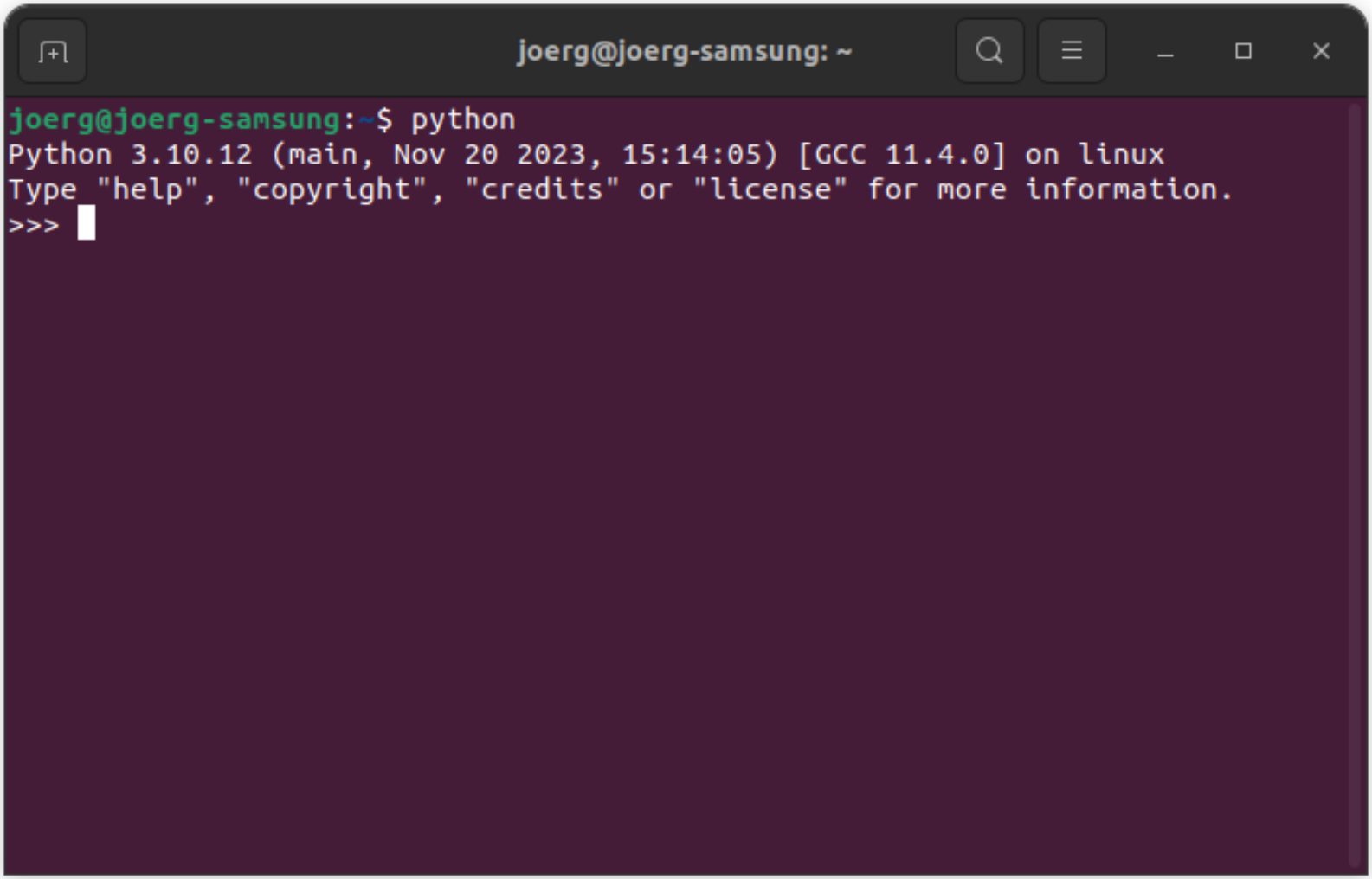
MEMORIAL UNIVERSITY

Interactive Python Shell

- Similar to R
- Type ”python” to launch Python interpreter
- Prompt is ”> > >”, type ENTER to execute a command
- Use `quit()` to exit
- **Tip**: Use a notepad app to assemble commands and to keep results

MEMORIAL UNIVERSITY

Interactive Python Shell



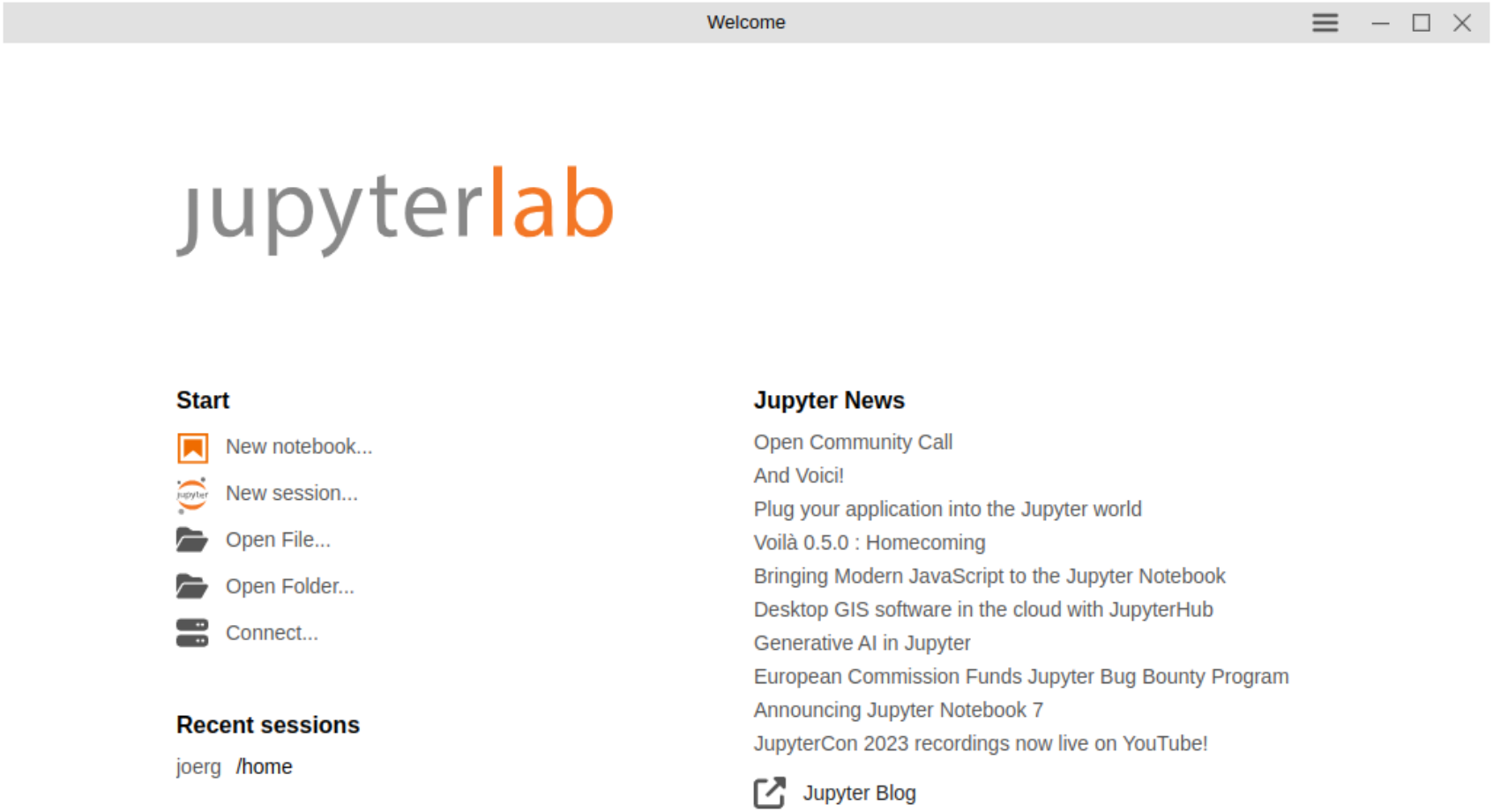
MEMORIAL UNIVERSITY

Jupyter Notebooks

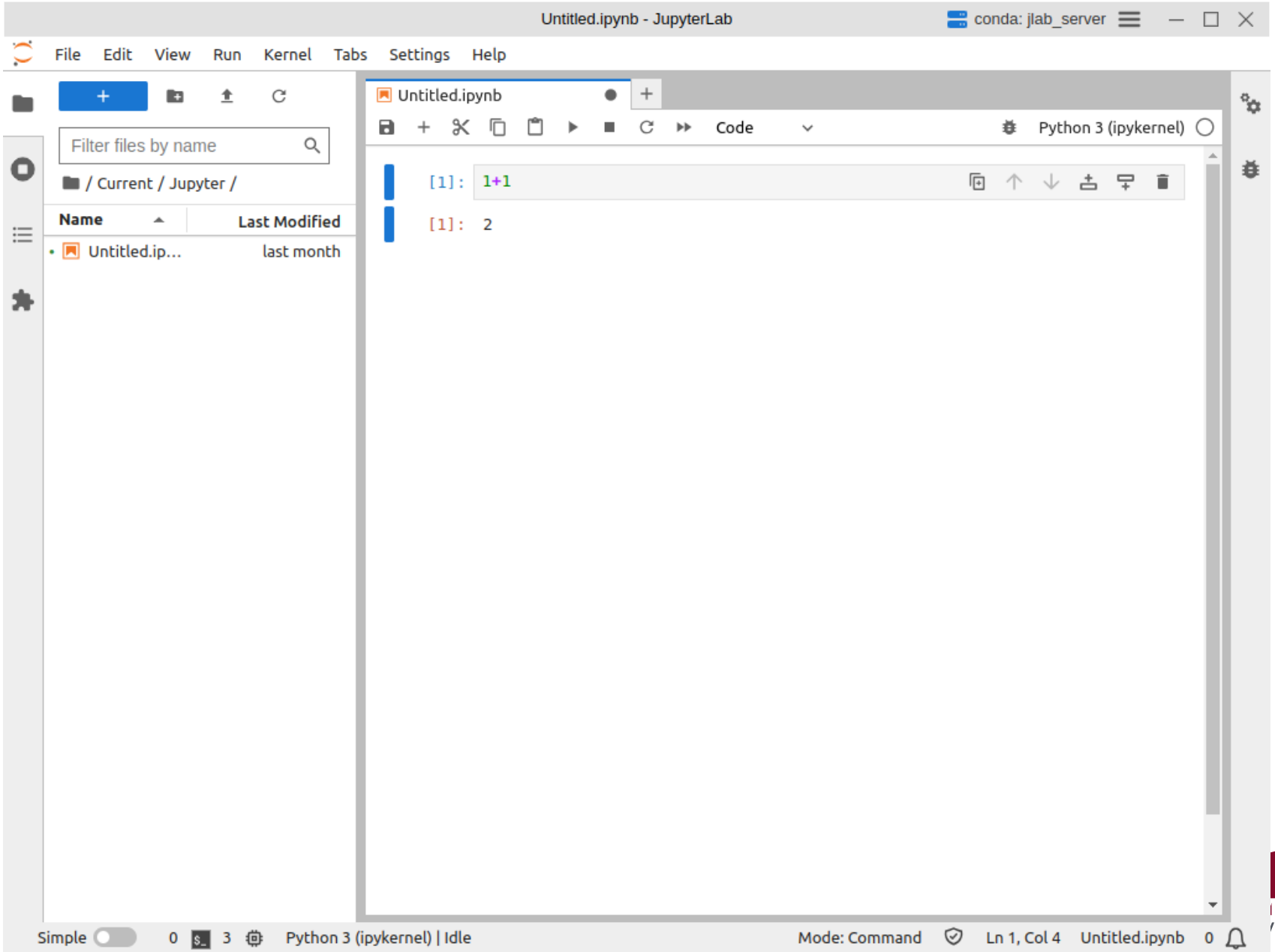
- Interactive computing environment
- Notebook Interface
- Combine executable code, text, visualizations
- Create and share documents with live code, equations, and explanatory text
- Collaborative editing of notebooks (on web-based services)
- Popular for Python, but can handle other languages

MEMORIAL UNIVERSITY

JupyterLabs Desktop



JupyterLabs Desktop



Jupyter Notebooks

- ”Kernel” is the Python interpreter and environment that runs your code
- Enter code into empty cell
- Press CTRL-ENTER to execute a cell
- Merge, split, move, copy, delete cells
- Save, import, export notebooks

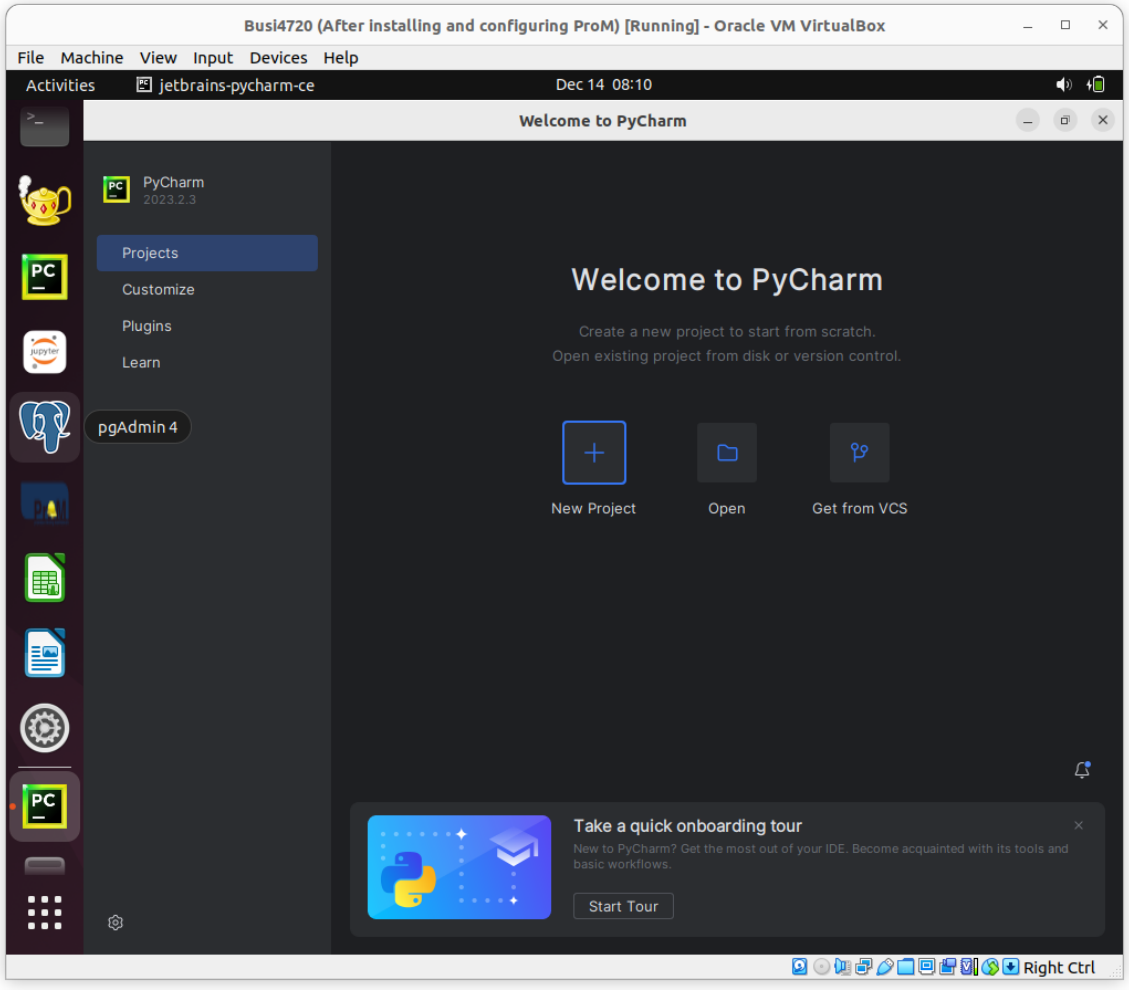
MEMORIAL UNIVERSITY

PyCharm IDE

- When working with multiple Python files in your project
- Useful for *programming* (defining functions, classes; using control structures, etc.) rather than just *scripting* (executing a few Python commands one after the other)
- Contains built-in debugging tools

MEMORIAL UNIVERSITY

PyCharm IDE


Busi4720 (After installing and configuring ProM) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
jetbrains-pycharm-ce
Dec 14 08:10
Welcome to PyCharm
PyCharm
2023.2.3
Projects
Customize
Plugins
Learn
pgAdmin 4
Welcome to PyCharm
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
Take a quick onboarding tour
New to PyCharm? Get the most out of your IDE. Become acquainted with its tools and basic workflows.
Start Tour
Right Ctrl

Printing a string:

```
print('hello world')
```

String format method:

```
age = 19
name = 'Malina'
print('{0} is {1} years old'.format(name, age))
print('{name} is {age} years old'.format(name=name,age=age))
print('{} is {} years old'.format(name, age))
print(f'{name} is {age} years old')
print(name+' is '+str(age)+' years old')
```

Backslashes split and continue lines:

```
print('This is a very long \
string and needs a second line')
i = \
5
print(i)
```

Multiline strings:

```
s = '''This is line 1
and here is line 2
and now this is line 3'''
print(s)
```

Basic Python

Python knows math:

```
2 + 2
2**4
13 // 3
-13 // 3
13 % 3
-25.5 % 2.25
3 < 5
3 > 5
3 == 5
(3 < 5) and (4 < 2)
(3 < 5) or not (4 < 2)
```

MEMORIAL UNIVERSITY

Strings

Python knows strings:

```
language = 'Innuktitut'
if language.startswith('Innu'):
    print('Yes, the string starts with "Innu"')
if 'u' in language:
    print('Yes, it contains the string "u"')
if language.find('nuk') != -1:
    print('Yes, it contains the string "nuk"')

# Joining and Splitting
delimiter = '_*_'
mylist = ['Nain', 'Hopedale', 'Makkovik', 'Rigolet']
mystring = delimiter.join(mylist)
print(mystring)
thelist = mystring.split(delimiter)
print(thelist)
```

MEMORIAL UNIVERSITY

Lists

Lists are ordered collections of items:

```
# Inuit deities
gods = ['Sedna', 'Nanook', 'Akna', 'Pinga']
print('There are', len(gods), 'deities:')
for item in gods:
    print(item, end=' ')

gods.append('Amaguq')
print('\nThe list of deities is now', gods)

gods.sort()
print('The sorted list is', gods)

print('The first deity is', gods[0])
olditem = gods[0]
del gods[0]
print('I removed', olditem)
print('The list is now', gods)
```

MEMORIAL UNIVERSITY

Tuples

Tuples are immutable:

```
# Inuit Nunangat
regions = ('Inuvialuit', 'Nunavut', \
           'Nunavik', 'Nunatsiavut')
print('Number of regions is', len(regions))

all_regions = 'NunatuKavummiut', 'Kalaallit', \
              'Inupiaq', regions
print('Number of all Inuit regions:',len(all_regions))
print('All Inuit regions are', all_regions)
print('Regions in Inuit Nunangat are', all_regions[3])
print('First region in Inuit Nunangat is', \
      all_regions[3][1])
print('Number of all Inuit regions is', \
      len(all_regions)-1+len(all_regions[3]))
```

MEMORIAL UNIVERSITY

Dictionaries

- Key–value pairs
- Associative arrays
- Map

```
# Largest citites
c = {
    'Inuvialuit': 'Inuvik',
    'Nunavut': 'Iqaluit',
    'Nunavik': 'Kuujjuaq',
    'Nunatsiavut': 'Nain'
}
print(c.keys())
print(c.values())

print("Nunavik's largest city is", c['Nunavik'])
# Deleting a key-value pair
del c['Nunavut']
print('\nThere are {} cities left\n'.format(len(c)))
for region, city in c.items():
    print('{} is largest city of {}'.format(city, region))
# Adding a key-value pair
c['Nunavut'] = 'Iqaluit'
if 'Nunavut' in c:
    print("\nNunavut's largest city is", c['Nunavut'])
```

Structured Data Types

Important

- Indexing begins at 0 (different from R!)
- Can contain any data type

Sequences

- List, tuples, strings are sequences
- Membership tests using `in` or `not in`
- Indexing and slicing

MEMORIAL UNIVERSITY

Slicing

```
regions = ('Inuvialuit', 'Nunavut',
           'Nunavik', 'Nunatsiavut')
language = 'Innuktitut'

# Slicing on a tuple
print('Item 1 to 3 is', regions[1:3])
print('Item 2 to end is', regions[2:])
print('Item 1 to -1 is', regions[1:-1])
print('Item start to end is', regions[:])
# Slicing on a string
print('characters 1 to 3 is', language[1:3])
print('characters 2 to end is', language[2:])
print('characters 1 to -1 is', language[1:-1])
print('characters start to end is', language[:])
# Slicing with step
print(regions[::1])
print(regions[::2])
print(regions[::3])
print(regions[::-1])
```

MEMORIAL UNIVERSITY

Hands-On Exercises

Lists

1. Create a list containing the numbers 1 to 10. Use list slicing to create a sublist with only the even numbers.
2. Using a `for` loop, sum all the items in the list.
3. Using a `for` loop, iterate over the list and print each number squared.
4. Write a program to append the square of each number in the range [1:5] to a new list.

MEMORIAL UNIVERSITY

Hands-On Exercises

Tuples

1. Create a tuple with different data types (string, int, float).
2. Demonstrate how tuples are immutable by attempting to change its first element.
3. Write a program to convert the tuple into a list.

MEMORIAL UNIVERSITY

Hands-On Exercises

Dictionaries

1. Create a dictionary with at least three key-value pairs, where the keys are strings and the values are numbers.
2. Write a Python script to add a new key-value pair to the dictionary and then print the updated dictionary.
3. Create a nested dictionary and demonstrate accessing elements at various levels.

MEMORIAL UNIVERSITY

What is Numpy?

- High-performance scientific computing and data analysis.
- Multidimensional arrays
- Comprehensive mathematical function library
- Foundational package for other scientific libraries like SciPy, Pandas, Matplotlib, scikit-learn, scikit-image, etc.

Intro Tutorials:

`https://numpy.org/doc/stable/user/quickstart.html`
`https://numpy.org/doc/stable/user/absolute_beginners.html`

MEMORIAL UNIVERSITY

NumPy Array

N-Dimensional Array, type "`ndarray`"

`ndarray.ndim`	Number of axes
`ndarray.shape`	Typle describing the size of each dimension (axis)
`ndarray.size`	Total number of elements
`ndarray.dtype`	The datatype of the elements, e.g. `numpy.int32`, `numpy.int16`, `numpy.float32`, `numpy.float64`
`ndarray.itemsize`	Number of bytes for each element

MEMORIAL UNIVERSITY

NumPy Basics

```
# Import the numpy package
import numpy as np

# Create an array
a = np.arange(15).reshape(3, 5)
print(a.shape)
print(a.ndim)
print(a.dtype.name)
print(a.size)
print(type(a))
```

MEMORIAL UNIVERSITY

NumPy Basics

```
# Create an array from Python lists and tuples
b = np.array([(1.5, 2., 3), (4, 5, 6)])
print(b)

# Elementwise operations
print(3 * b)
print(b + 5)
print(np.sqrt(b))

# Array operations
print(np.max(b))
print(np.max(b, axis=0))
print(np.max(b, axis=1))
print(np.std(b))
print(np.cov(b))
print(np.sum(b))
```

MEMORIAL UNIVERSITY

NumPy Basics [cont'd]

```
# Create an array of zeros with shape (3,4)
x = np.zeros((3,4))
print(x)

# Create an array of ones with shape (2,3,4)
y = np.ones((2,3,4))
print(y)
```

MEMORIAL UNIVERSITY

Array Slicing

- Each axis can be sliced using `[:]` or `[::]`

```
b = np.array([[ 0,  1,  2,  3],
              [10, 11, 12, 13],
              [20, 21, 22, 23],
              [30, 31, 32, 33],
              [40, 41, 42, 43]])

print(b[2, 3])
print(b[0:5, 1])
print(b[:, 1])
print(b[1:3, :])
print(b[-1])
```

MEMORIAL UNIVERSITY

Array Slicing & Iterators

```
c = np.array([[[  0,  1,  2],
               [ 10, 12, 13]],
              [[100, 101, 102],
               [110, 112, 113]]])

print(c.shape)
print(c[1, ...])
print(c[1, : , : ])
print(c[..., 2])
print(c[: , : , 2])
print(c[..., : , 1])

for row in b:
    print(row)

for element in b.flat:
    print(element)
```

MEMORIAL UNIVERSITY

Array Reshaping

```
rg = np.random.default_rng(1)
a = np.floor(10 * rg.random((3, 4)))

print(a.shape)
print(a.flatten())
print(a.reshape(6, 2))
print(a.T)
print(a.T.shape)

b = np.floor(5 * rg.random((3, 4)))
print(np.vstack((a, b)))
print(np.hstack((b, a)))
```

MEMORIAL UNIVERSITY

Array Indexing

```
a = np.array([[1, 2, 3, 4],
              [5, 6, 7, 8],
              [9, 10, 11, 12]])

print(a[a < 5])
print(a < 5)
print(a[a%2 == 0])
print(a%2 == 0)
```

Unique Elements and Counts

```
a = np.array([11, 11, 12, 13, 14, 15, 16,
              17, 12, 13, 11, 14, 18, 19, 20])

print(np.unique(a))

values, indices = np.unique(a, return_index=True)
print(list(zip(values, indices)))

values, counts = np.unique(a, return_counts=True)
print(list(zip(values, counts)))
```

MEMORIAL UNIVERSITY

Hands-On Exercises

1. Create an array with random numbers in the shape indicated by the last four digits of your student number (if your student number contains a 0, use a 1 instead)
2. Construct a new array by swapping the first half of rows (axis 0) with the second half of rows (axis 0)
3. Calculate all covariance matrices formed by the last two axes of your array. *Tip:* Iterate over the first two axes/dimensions with a `for` loop
4. Subtract the mean of the array from each element in the array (mean normalization)
5. Select all elements that are greater than the overall mean
6. Sort the selected elements from the previous step

MEMORIAL UNIVERSITY

Data Management with Pandas

What is Pandas?

- Open-source library for data analysis
- High-performance, easy-to-use data structures and data analysis tools
- Can handle tabular data, time series, matrix data, etc.
- Tools for data cleaning, transformation, and preparation
- Importing data from CSV, Excel, SQL databases, etc.
- Functions for aggregating, pivoting, joining, and sorting data

Intro Tutorial:

`http://pandas.pydata.org/docs/user_guide/10min.html`

MEMORIAL UNIVERSITY

Pandas Series

- 1-Dimensional *labeled* array
- Axis labels are called *index*

```
# Import the Pandas package
import pandas as pd

s = pd.Series(np.random.randn(5))
print(s.index)

s = pd.Series(np.random.randn(5),
              index=["a", "b", "c", "d", "e"])
print(s.index)

d = {"a": 0.0, "b": 1.0, "c": 2.0}
print(pd.Series(d))
pd.Series(d, index=["b", "c", "d", "a"])
```

MEMORIAL UNIVERSITY

Pandas Series [cont'd]

```
# Series behave like an ndarray
print(s.iloc[0])
print(s.iloc[:3])
print(s[s > s.median()])
print(s.iloc[[4, 3, 1]])
print(np.exp(s))

# Series behave like a dict
print(s['a'])
print(s['e'])
print('e' in s)
print('f' in s)

# Series have a datatype and name
s.name = 'My First Series'
print(s.dtype)
```

MEMORIAL UNIVERSITY

Pandas Dataframe

- 2-Dimensional
- Columns may have different data types
- Conceptually a dict of pandas series

```
d = {
    "one": pd.Series([1.0, 2.0, 3.0],
                     index=['a', 'b', 'c']),
    "two": pd.Series([1.0, 2.0, 3.0, 4.0],
                     index=['a', 'b', 'c', 'd'])
}
df = pd.DataFrame(d)
print(df)
print(df.index)
print(df.columns)

print(pd.DataFrame(d, index=['d', 'b', 'a'],
                      columns=['two', 'three']))
```

MEMORIAL UNIVERSITY

Pandas Dataframe Columns

```
print(df['one'])
df['three'] = df['one'] * df['two']
df['flag'] = df['one'] > 2
print(df)

del df['two']
three = df.pop('three')
df['foo'] = 'bar'
df['one_trunc'] = df['one'][:2]
df.insert(1, 'bar', df['one'])
print(df)

# Similar to 'mutate' in R/Dplyr
df = df.assign(four = df['one'] * np.sqrt(df['bar']))
print(df)
```

MEMORIAL UNIVERSITY

Dataframe Indexing

Select column	`df[col]`	Series
Select row by label	`df.loc[label]`	Series
Select row by integer location	`df.iloc[loc]`	Series
Slice rows	`df[:]`	DataFrame
Select rows by boolean vector	`df[bool]`	DataFrame

MEMORIAL UNIVERSITY

Dataframe Alignment and Arithmetic

- Data is aligned on column labels and row indices

```
df = pd.DataFrame(np.random.randn(10, 4),
                  columns=["A", "B", "C", "D"])
df2 = pd.DataFrame(np.random.randn(7, 3),
                   columns=["A", "B", "C"])
print(df + df2)

# Elementwise operators
print(df * 5 + 2)
print(1/df)
print(df**4)
# Transpose
print(df.T)
# Using Numpy functions
print(np.exp(df))
print(np.asarray(df))
```

MEMORIAL UNIVERSITY

Pandas Dataframe

```
df.info()
df.head()
df.tail(3)

# Boolean reductions
(df > 0).all()
(df > 0).any()
(df > 0).any().any()

# NaN's are not the same
df.iloc[0,0] = np.nan
(df+df == df*2).all()
(df + df).equals(df*2)
```

MEMORIAL UNIVERSITY

Descriptive Statistics, Aggregation, & Strings

```
# Descriptive statistics
df.mean(0)
df.mean(1, skipna=False)
df_std = (df - df.mean()) / df.std()
df.describe()

# Aggregation with 'agg'
df.agg(['sum', 'mean', 'std'], 0)

# Sort by values
df.sort_values(by=['A', 'B'])
df.nsmallest(3, 'A')
df.nlargest(3, 'A')

# String functions with 'str'
s = pd.Series(
    ["A", "B", "C", "Aaba", "Baca", np.nan,
     "CABA", "dog", "cat"], dtype="string")
s.str.lower()
```

MEMORIAL UNIVERSITY

Selection with Query

```
df = pd.DataFrame(np.random.rand(n, 3),
                  columns=list('abc'))

# Pure python
df[(df['a'] < df['b']) & (df['b'] < df['c'])]

# Shorter with Query
df.query('(a < b) & (b < c)')
df.query('a < b & b < c')
df.query('a < b and b < c')
df.query('a < b < c')
```

MEMORIAL UNIVERSITY

Selection with Query [cont'd]

```
df = pd.DataFrame({'a': list('aabbccddeeff'),
                   'b': list('aaaabbbbcccc'),
                   'c': np.random.randint(5, size=12),
                   'd': np.random.randint(9, size=12)})

# Pure Python versus Query
df[df['a'].isin(df['b'])]
df.query('a in b')

df[~df['a'].isin(df['b'])]
df.query('a not in b')

df[df['b'].isin(df['a']) & (df['c'] < df['d'])]
df.query('a in b and c < d')

df[df['b'].isin(["a", "b", "c"])]
df.query('b == ["a", "b", "c"]')

df[df['c'].isin([1, 2])]
df.query('[1, 2] in c')
```

MEMORIAL UNIVERSITY

Duplicate Data

```
df2 = df.copy()

df2.duplicated(['a', 'b'])
df2.drop_duplicates(['a', 'b'], keep='last')
df2.drop_duplicates(['a', 'b'], keep='first')
```

MEMORIAL UNIVERSITY

Reading Data into Pandas

- Wide variety of format: CSV, JSON, Excel, SQL, ...
- `https://pandas.pydata.org/docs/user_guide/io.html#`

```
rentals = pd.read_csv('pagila/rentals.csv')

rentals['rental_date'] = \
     pd.to_datetime(rentals['rental_date'], utc=True)
rentals['return_date'] = \
     pd.to_datetime(rentals['return_date'], utc=True)
rentals['payment_date'] = \
     pd.to_datetime(rentals['payment_date'], utc=True)

rentals.info()
rentals.describe()
rentals.index
rentals.columns
rentals.shape
```

MEMORIAL UNIVERSITY

Examine the NA's

```
filtered_rentals = \
    rentals[rentals.isna().any(axis=1)]

selected_rentals = \
    filtered_rentals[
        ['last_name', 'rental_date',
         'return_date', 'title', 'amount']]

pd.set_option('display.max_rows', None)
pd.set_option('display.width', None)

print(selected_rentals)
```

MEMORIAL UNIVERSITY

Pagila Database in Python]

Find all films and the actors that appeared in them, ordered by film category and year, for those films that are rated PG:

```
actors = pd.read_csv('pagila/actors.categories.csv')

result = pd.merge(rentals, actors, on='title',
          suffixes=('_customer', '_actor'),
          how='outer')

result = result[result['rating'] == 'PG']
result['actor'] = result['last_name_actor'] + \
          ', ' + result['first_name_actor']
result.rename(columns={'release_year': 'year'},
          inplace=True)
result = result[['actor', 'title', 'category', 'year']]
result.drop_duplicates(['actor', 'title', 'category', 'year'],
          inplace=True)
result.sort_values(['category', 'year', 'title'],
          inplace=True)
grouped = result.groupby(['category', 'year', 'title'])
g_result = grouped['actor'].apply(list).reset_index()

print(g_result)
```

Pagila Database in Python

Find the most popular actors in the rentals in each city:

```
addresses = pd.read_csv('pagila/addresses.csv')
addresses['phone'] = addresses['phone'].astype(str)

joined_df = pd.merge(rentals, addresses,
                     left_on='customer_address',
                     right_on='address_id')
joined_df = pd.merge(joined_df, actors,
                     on='title',
                     suffixes=('_customer', '_actor'))
joined_df['actor'] = joined_df['last_name_actor'] + \
              ', ' + joined_df['first_name_actor']

grouped = joined_df.groupby(['city', 'actor']). \
            size().reset_index(name='count')
grouped['ranking'] = grouped.groupby('city')['count']. \
            rank(method='min', ascending=False)
filtered = grouped[grouped['ranking'] < 4]
sorted_df = filtered.sort_values(
               by=['city', 'ranking', 'actor'])

print(sorted_df.head(25))
```

Pagila Database in Python

Find the customers who spend the most on rentals, and the number of rentals with the higest total rental payments for each category grouped by rental duration.

```
full_data = pd.merge(rentals, addresses,
                     left_on='customer_address',
                     right_on='address_id')
full_data = pd.merge(full_data, actors,
                     on='title',
                     suffixes=('_customer', '_actor'))

full_data['customer'] = \
   full_data['first_name_customer'] + \
   ' ' + full_data['last_name_customer']
selected_data = \
   full_data[['customer', 'amount', 'rental_duration', \
              'category', 'phone', 'city']]
```

MEMORIAL UNIVERSITY

Pagila Database in Pythonc [cont'd]

. . . continued from previous slide . . .

```
grouped_data = selected_data.groupby( \
  ['category', 'rental_duration', 'customer']).agg( \
    payments=pd.NamedAgg('amount', 'sum'), \
    num_rentals=pd.NamedAgg('amount', 'count')).reset_index()

grouped_data['ranking'] = \
  grouped_data.groupby(['category', 'rental_duration']) \
    ['payments'].rank(method='min', ascending=False)

top_entries = grouped_data.loc[
    grouped_data.groupby(['category', 'rental_duration']) \
    ['ranking'].idxmin() \
  ]

print(top_entries)
```

MEMORIAL UNIVERSITY

Pagila Database in Python

Get the top 5 and the bottom 5 grossing customers for each quarter.

```
full_data['customer'] = \
    full_data['first_name_customer'] + ' ' + \
    full_data['last_name_customer']

full_data['q'] = pd.to_datetime( \
    full_data['rental_date']).dt.to_period("Q")

selected_data = \
    full_data[['customer', 'q', 'amount', 'rental_date']]

grouped_data = \
    selected_data.groupby(['q', 'customer']) \
        .agg(payments=('amount', 'sum')).reset_index()

distinct_data = \
    grouped_data.drop_duplicates(
        subset=['customer', 'q', 'payments'])
```

MEMORIAL UNIVERSITY

Pagila Database in Python [cont'd]

. . . continued from previous slide . . .

```
distinct_data['rank_top'] = \
    distinct_data.groupby('q')['payments'] \
    .rank(method='min', ascending=False)

distinct_data['rank_bot'] = \
    distinct_data.groupby('q')['payments'] \
    .rank(method='min', ascending=True)

filtered_data = \
    distinct_data[
        (distinct_data['rank_top'] < 6) |
        (distinct_data['rank_bot'] < 6)]

sorted_data = \
    filtered_data.sort_values(
        by=['q', 'payments'],
        ascending=[True, False])
print(sorted_data)
```

MEMORIAL UNIVERSITY

Pagila Database in Python [cont'd]

Find the set of film titles by rental customer and the total number rentals for each customer

```
full_data['customer'] = \
    full_data['first_name_customer'] + ' ' + \
    full_data['last_name_customer']

selected_data = full_data[['customer', 'title']]

grouped_data = \
    selected_data.groupby('customer')['title'] \
        .apply(list).reset_index(name='titles')
grouped_data['rentals'] = \
     grouped_data['titles'].apply(len)
grouped_data['unique_titles'] = \
     grouped_data['titles'].apply(lambda x: list(set(x)))

grouped_data = grouped_data.drop(columns=['titles'])
sorted_data = grouped_data.sort_values(by='customer')

print(sorted_data)
```

Hands-On Exercises

1. Find all films with a rating of 'PG'
2. List all customers who live in Canada (with their address)
3. Find the average *actual* rental duration for all films
   - This requires date arithmetic
4. Find the average overdue time for each customer
   - This requires date arithmetic
5. List all films that have never been rented
6. List the names of actors who have played in more than 15 films

MEMORIAL UNIVERSITY