Flow Control, Functions, and Scientific Computing Data Structures

CCT490H5F - Social Data Analytics

Professor Alex Hanna

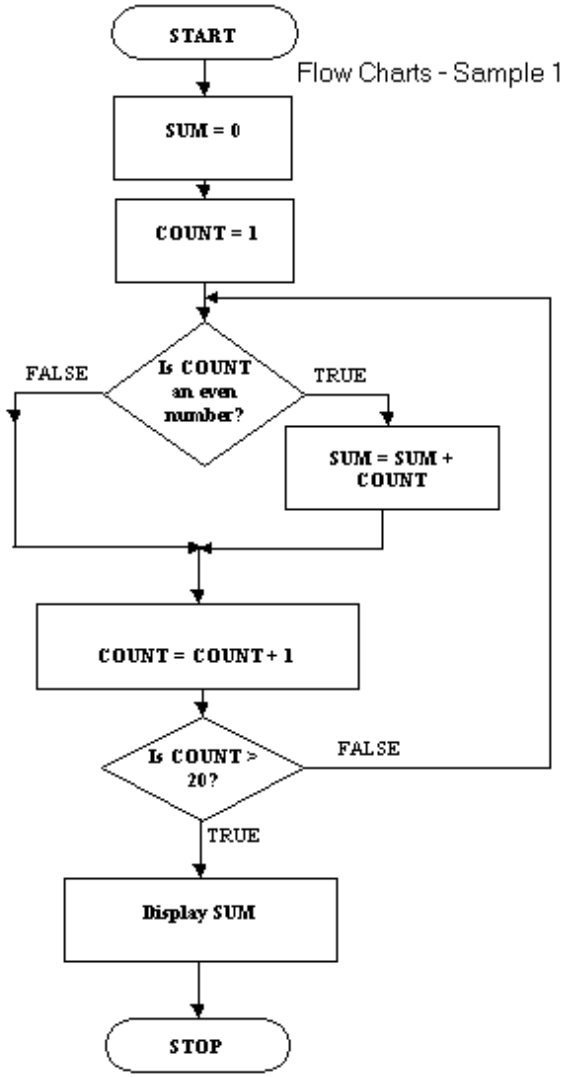
September 29, 2016

Flow Control

Flow control refers which parts of the program are executed, in what order.

Types of flow control

- Loops
- Conditional statements
- Functions

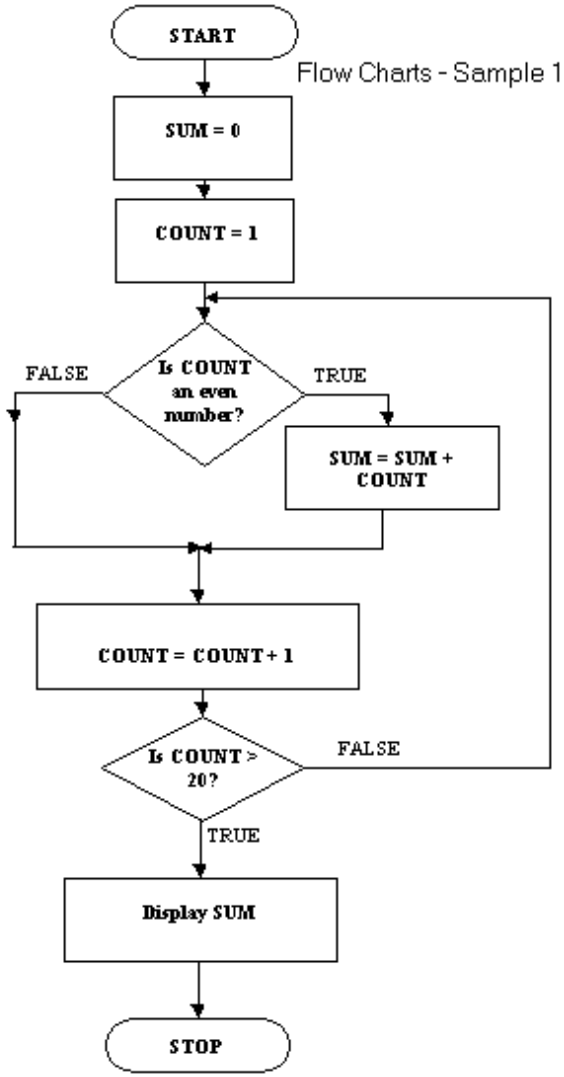


Flow Control

Flow control refers which parts of the program are executed, in what order.

Types of flow control

- Loops
- Conditional statements
- Functions



Loops

`for` loop

```
for i in range(100):
    print(i)
```

Loops

`while` loop

```
i = 0
while i < 100:
    print(i)
    i = i + 1
```

Conditional statements

if it is raining then
the sky is cloudy
else if it is snowing then
the sky is cloudy
else
the sky is clear

```
sky = None
is_raining = True
is_snowing = False
if is_raining:
    sky = "cloudy"
elif is_snowing:
    sky = "cloudy"
else:
    sky = "clear"
```

Conditional statements

if it is raining then
the sky is cloudy
else if it is snowing then
the sky is cloudy
else
the sky is clear

```
sky = None
is_raining = True
is_snowing = False
if is_raining:
    sky = "cloudy"
elif is_snowing:
    sky = "cloudy"
else:
    sky = "clear"
```

Conditional statements

if it is raining then
the sky is cloudy
else if it is snowing then
the sky is cloudy
else
the sky is clear

`is_raining` and
`is_snowing` are booleans

```
sky = None
is_raining = True
is_snowing = False
if is_raining == True:
    sky = "cloudy"
elif is_snowing == True:
    sky = "cloudy"
else:
    sky = "clear"
```

Comparison Operators

Numerical operators

- ==
- !=
- >
- >=
- <
- <=

Examples

```
3 == 4 # False
3 != 4 # True
3 > 4  # False
3 < 4  # True
```

Comparison Operators

Numerical operators

- ==
- !=
- >
- >=
- <
- <=

Examples

```
3 == 4 # False
3 != 4 # True
3 > 4  # False
3 < 4  # True
```

Comparison Operators

Numerical operators

- ==
- !=
- >
- >=
- <
- <=

Examples

```
3 == 4 # False
3 != 4 # True
3 > 4  # False
3 < 4  # True
```

Comparison Operators

Numerical operators

- ==
- !=
- >
- >=
- <
- <=

Examples

```
3 == 4 # False
3 != 4 # True
3 > 4  # False
3 < 4  # True
```

Comparison Operators

Numerical operators

- ==
- !=
- >
- >=
- <
- <=

Examples

```
3 == 4 # False
3 != 4 # True
3 > 4  # False
3 < 4  # True
```

Comparison Operators

Numerical operators

- ==
- !=
- >
- >=
- <
- <=

Examples

```
3 == 4 # False
3 != 4 # True
3 > 4  # False
3 < 4  # True
```

Comparison Operators

Numerical operators

- `==`
- `!=`
- `>`
- `>=`
- `<`
- `<=`

Examples

```
3 == 4 # False
3 != 4 # True
3 > 4  # False
3 < 4  # True
```

Logical Operators

Logical operators tie various logical statements together

- `and`
- `or`

```
True and True
True and False
False and False
True or True
True or False
False or False
```

Logical Operators

Logical operators tie various logical statements together

- `and`
- `or`

```
True and True    # True
True and False   # False
False and False  # False
True or True     # True
True or False    # True
False or False   # False
```

Logical Operators

You can combine logical operations with operators

```
(4 > 3) and (8 > 5)                # True
True or (0.2 < -2)                 # True
(-3 == 2) and (3 == 3)             # False
("hello" == "kitty") and True      # False
```

You can also assign them to variables

```
is_4gt3 = 4 > 3
is_8gt5 = 8 > 5
is_4gt3 and is_8gt5 # True
```

Logical Operators

You can combine logical operations with operators

```
(4 > 3) and (8 > 5)                # True
True or (0.2 < -2)                 # True
(-3 == 2) and (3 == 3)             # False
("hello" == "kitty") and True      # False
```

You can also assign them to variables

```
is_4gt3 = 4 > 3
is_8gt5 = 8 > 5
is_4gt3 and is_8gt5 # True
```

Using Conditional Statements

`if`, `elif`, and `while` all work with a conditional statement

```
import random
count = random.randint(-10, 10)
if count < 0:
    print("count is negative.")
elif count == 0:
    print("count is zero.")
else:
    print("count is positive.")
```

```
count = -10
while count < 0:
    print("count is negative")
    print(count)
    count = random.randint(-10, 10)
```

Using Conditional Statements

`if`, `elif`, and `while` all work with a conditional statement

```
import random
count = random.randint(-10, 10)
if count < 0:
    print("count is negative.")
elif count == 0:
    print("count is zero.")
else:
    print("count is positive.")
```

```
count = -10
while count < 0:
    print("count is negative")
    print(count)
    count = random.randint(-10, 10)
```

Using Conditional Statements

`if`, `elif`, and `while` all work with a conditional statement

```
import random
count = random.randint(-10, 10)
if count < 0:
    print("count is negative.")
elif count == 0:
    print("count is zero.")
else:
    print("count is positive.")
```

```
count = -10
while count < 0:
    print("count is negative")
    print(count)
    count = random.randint(-10, 10)
```

Exercises

1. Write code which goes through each number from 0 to 20 and prints the number if it is divisible by 5.
2. Write code which begins with a number equal to 1000. Print the number, then subtract 1 from the variable until the number equals zero.

Functions

- A *function* is a piece of code which can be written once and run many times
- Examples of built-in functions: `range`, `print`, `randint`
- Two most important parts of a function
  - *arguments* - what goes between the parentheses
  - *return value* - what comes back
  - Both are optional

Functions

- A *function* is a piece of code which can be written once and run many times
- Examples of built-in functions: `range`, `print`, `randint`
- Two most important parts of a function
  - *arguments* - what goes between the parentheses
  - *return value* - what comes back
  - Both are optional

Functions

- A *function* is a piece of code which can be written once and run many times
- Examples of built-in functions: `range`, `print`, `randint`
- Two most important parts of a function
  - *arguments* - what goes between the parentheses
  - *return value* - what comes back
  - Both are optional

Functions

- A *function* is a piece of code which can be written once and run many times
- Examples of built-in functions: `range`, `print`, `randint`
- Two most important parts of a function
  - *arguments* - what goes between the parentheses
  - *return value* - what comes back
  - Both are optional

Functions

- A *function* is a piece of code which can be written once and run many times
- Examples of built-in functions: `range`, `print`, `randint`
- Two most important parts of a function
  - *arguments* - what goes between the parentheses
  - *return value* - what comes back
  - Both are optional

Functions

- A *function* is a piece of code which can be written once and run many times
- Examples of built-in functions: `range`, `print`, `randint`
- Two most important parts of a function
  - *arguments* - what goes between the parentheses
  - *return value* - what comes back
  - Both are optional

Built-in Functions

List functions

- `len` - length of list
- `sorted` - sorts a list
- `list` - converts to list

```
len([1, 2, 3])
# 3
```

```
sorted([3, 2, 1])
# [1, 2, 3]
```

```
list( (1, 2, 3) )
# [1, 2, 3]
```

Built-in Functions

List functions

- `len` - length of list
- `sorted` - sorts a list
- `list` - converts to list

```
len([1, 2, 3])
# 3
```

```
sorted([3, 2, 1])
# [1, 2, 3]
```

```
list( (1, 2, 3) )
# [1, 2, 3]
```

Built-in Functions

List functions

- `len` - length of list
- `sorted` - sorts a list
- `list` - converts to list

```
len([1, 2, 3])
# 3
```

```
sorted([3, 2, 1])
# [1, 2, 3]
```

```
list( (1, 2, 3) )
# [1, 2, 3]
```

Built-in Functions

Type functions

- `type` - get type of object
- `int` - convert to integer
- `float` - convert to float
- `str` - convert to string

```
type([1, 2, 3])
# list

type(1.3)
# float

int("10")
# 10

float(1000)
# 1000.0

str(100)
Out[]: '100'
```

Built-in Functions

Type functions

- `type` - get type of object
- `int` - convert to integer
- `float` - convert to float
- `str` - convert to string

```
type([1, 2, 3])
# list

type(1.3)
# float

int("10")
# 10

float(1000)
# 1000.0

str(100)
Out[]: '100'
```

Built-in Functions

Type functions

- `type` - get type of object
- `int` - convert to integer
- `float` - convert to float
- `str` - convert to string

```
type([1, 2, 3])
# list

type(1.3)
# float

int("10")
# 10

float(1000)
# 1000.0

str(100)
Out[]: '100'
```

Built-in Functions

Type functions

- `type` - get type of object
- `int` - convert to integer
- `float` - convert to float
- `str` - convert to string

```
type([1, 2, 3])
# list

type(1.3)
# float

int("10")
# 10

float(1000)
# 1000.0

str(100)
Out[]: '100'
```

User-defined functions

Define your own functions if you plan using the same piece of code over and over.

```
def my_mean(my_list):
    sum = 0
    for i in my_list:
        sum += i
    return sum / len(my_list)

my_mean([1, 2, 4, 8, 10])
# 5.0
```

User-defined functions

`def` says that this will be a user-defined function.

```
def my_mean(my_list):
    sum = 0
    for i in my_list:
        sum += i
    return sum / len(my_list)

my_mean([1, 2, 4, 8, 10])
# 5.0
```

User-defined functions

What comes after `def` is the function name.

```
def my_mean(my_list):
    sum = 0
    for i in my_list:
        sum += i
    return sum / len(my_list)

my_mean([1, 2, 4, 8, 10])
# 5.0
```

User-defined functions

In between the parentheses are the arguments. You use them as variables within the function.

```
def my_mean(my_list):
    sum = 0
    for i in my_list:
        sum += i
    return sum / len(my_list)

my_mean([1, 2, 4, 8, 10])
# 5.0
```

User-defined functions

`return` is a special function which gives a value back from the function.

```
def my_mean(my_list):
    sum = 0
    for i in my_list:
        sum += i
    return sum / len(my_list)

my_mean([1, 2, 4, 8, 10])
5.0
```

Methods

A *method* is a function which is associated with an object. The big difference is the syntax, which is

```
object.method(args)
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
# [1, 2, 3, 4]
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
# [1, 2, 3, 4]

my_list.extend([1, 2, 4])
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
# [1, 2, 3, 4]

my_list.extend([1, 2, 4])
# [1, 2, 3, 4, 1, 2, 4]
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
# [1, 2, 3, 4]

my_list.extend([1, 2, 4])
# [1, 2, 3, 4, 1, 2, 4]

my_list.count(4)
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
# [1, 2, 3, 4]

my_list.extend([1, 2, 4])
# [1, 2, 3, 4, 1, 2, 4]

my_list.count(4)
# 2
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
# [1, 2, 3, 4]

my_list.extend([1, 2, 4])
# [1, 2, 3, 4, 1, 2, 4]

my_list.count(4)
# 2

my_list.remove(1)
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
# [1, 2, 3, 4]

my_list.extend([1, 2, 4])
# [1, 2, 3, 4, 1, 2, 4]

my_list.count(4)
# 2

my_list.remove(1)
# [2, 3, 4, 1, 2, 4]
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
# [1, 2, 3, 4]

my_list.extend([1, 2, 4])
# [1, 2, 3, 4, 1, 2, 4]

my_list.count(4)
# 2

my_list.remove(1)
# [2, 3, 4, 1, 2, 4]

my_list.clear()
```

List methods

```
my_list = [1, 2, 3]
my_list.append(4)
# [1, 2, 3, 4]

my_list.extend([1, 2, 4])
# [1, 2, 3, 4, 1, 2, 4]

my_list.count(4)
# 2

my_list.remove(1)
# [2, 3, 4, 1, 2, 4]

my_list.clear()
# []
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
# 'This is a string. it is a good string'
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
# 'This is a string. it is a good string'

my_string.find("it")
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
# 'This is a string. it is a good string'

my_string.find("it")
# 18
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
# 'This is a string. it is a good string'

my_string.find("it")
# 18

my_string.count("is")
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
# 'This is a string. it is a good string'

my_string.find("it")
# 18

my_string.count("is")
# 2
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
# 'This is a string. it is a good string'

my_string.find("it")
# 18

my_string.count("is")
# 2

my_string.replace("string", "bit of text")
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
# 'This is a string. it is a good string'

my_string.find("it")
# 18

my_string.count("is")
# 2

my_string.replace("string", "bit of text")
# 'this is a bit of text. it is a good bit of text'
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
# 'This is a string. it is a good string'

my_string.find("it")
# 18

my_string.count("is")
# 2

my_string.replace("string", "bit of text")
# 'this is a bit of text. it is a good bit of text'

my_string.split()
```

String methods

```
my_string = "this is a string. it is a good string"
my_string.capitalize()
# 'This is a string. it is a good string'

my_string.find("it")
# 18

my_string.count("is")
# 2

my_string.replace("string", "bit of text")
# 'this is a bit of text. it is a good bit of text'

my_string.split()
# ['this', 'is', 'a', 'string.', 'it', 'is',
    'a', 'good', 'string']
```

Exercises

1. Write a function which checks if a string is in another string. If so, return True. If not, return False.
2. Write a function which takes two numbers x and y, raises x to the y power, and subtracts one, i.e. $f(x, y) = x^y - 1$. Return the result.

Modules

Modules are prewritten libraries of objects and functions.

Three ways of importing modules

```
import random
random.randint(-1, 1)
```

Modules

Modules are prewritten libraries of objects and functions.

Three ways of importing modules

```
import random
random.randint(-1, 1)
```

```
import random as ra
ra.randint(-1, 1)
```

Modules

Modules are prewritten libraries of objects and functions.

Three ways of importing modules

```
import random
random.randint(-1, 1)
```

```
import random as ra
ra.randint(-1, 1)
```

```
from random import randint
randint(-1, 1)
```

SciPy and pandas

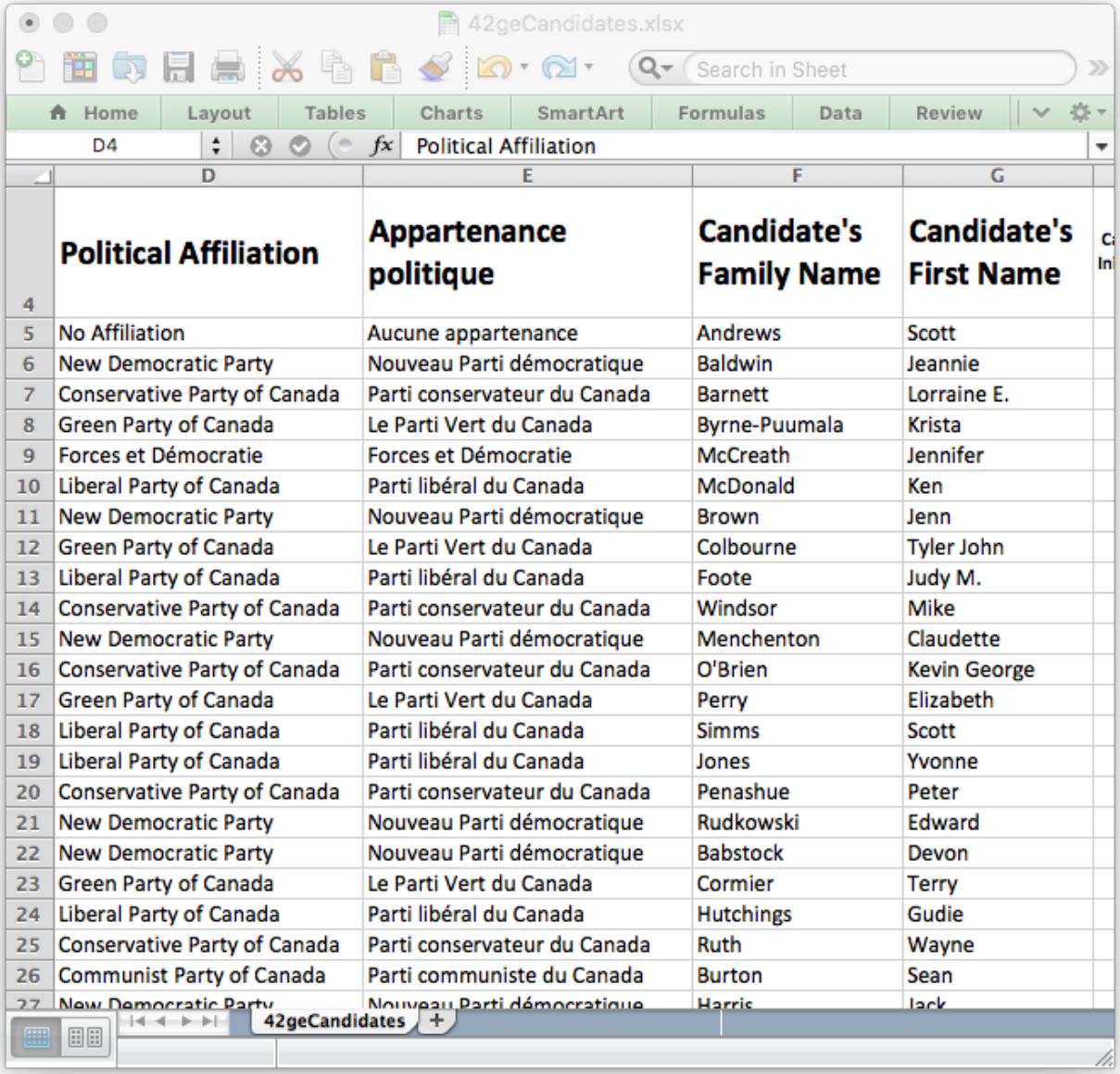
SciPy is a module for scientific and technical computing.

pandas is a module for data manipulation and analysis.

We will usually import these two modules like so:

```
import numpy as np
import pandas as pd
```

Data Analysis data format



42geCandidates.xlsx

Search in Sheet

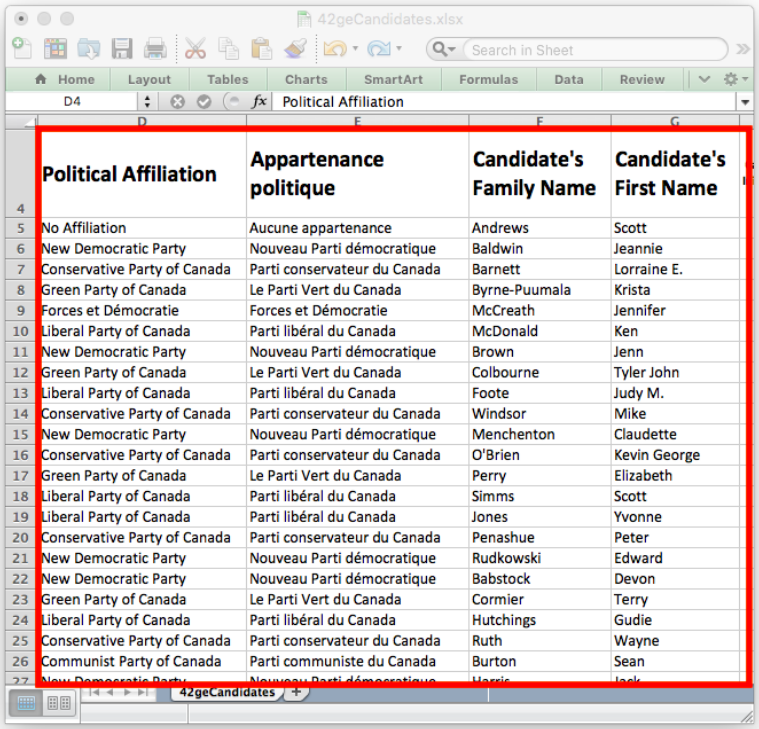
Home | Layout | Tables | Charts | SmartArt | Formulas | Data | Review

D4 | Political Affiliation

	D	E	F	G
4	**Political Affiliation**	**Appartenance politique**	**Candidate's Family Name**	**Candidate's First Name**
5	No Affiliation	Aucune appartenance	Andrews	Scott
6	New Democratic Party	Nouveau Parti démocratique	Baldwin	Jeannie
7	Conservative Party of Canada	Parti conservateur du Canada	Barnett	Lorraine E.
8	Green Party of Canada	Le Parti Vert du Canada	Byrne-Puumala	Krista
9	Forces et Démocratie	Forces et Démocratie	McCreath	Jennifer
10	Liberal Party of Canada	Parti libéral du Canada	McDonald	Ken
11	New Democratic Party	Nouveau Parti démocratique	Brown	Jenn
12	Green Party of Canada	Le Parti Vert du Canada	Colbourne	Tyler John
13	Liberal Party of Canada	Parti libéral du Canada	Foote	Judy M.
14	Conservative Party of Canada	Parti conservateur du Canada	Windsor	Mike
15	New Democratic Party	Nouveau Parti démocratique	Menchenton	Claudette
16	Conservative Party of Canada	Parti conservateur du Canada	O'Brien	Kevin George
17	Green Party of Canada	Le Parti Vert du Canada	Perry	Elizabeth
18	Liberal Party of Canada	Parti libéral du Canada	Simms	Scott
19	Liberal Party of Canada	Parti libéral du Canada	Jones	Yvonne
20	Conservative Party of Canada	Parti conservateur du Canada	Penashue	Peter
21	New Democratic Party	Nouveau Parti démocratique	Rudkowski	Edward
22	New Democratic Party	Nouveau Parti démocratique	Babstock	Devon
23	Green Party of Canada	Le Parti Vert du Canada	Cormier	Terry
24	Liberal Party of Canada	Parti libéral du Canada	Hutchings	Gudie
25	Conservative Party of Canada	Parti conservateur du Canada	Ruth	Wayne
26	Communist Party of Canada	Parti communiste du Canada	Burton	Sean
27	New Democratic Party	Nouveau Parti démocratique	Harris	Jack

42geCandidates

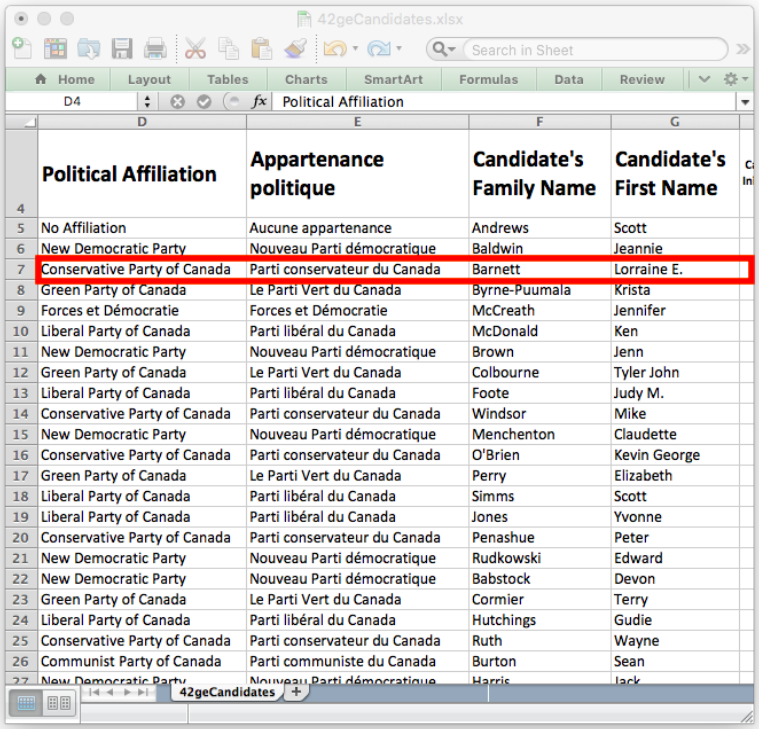
DataFrame



Political Affiliation	Appartenance politique	Candidate's Family Name	Candidate's First Name
No Affiliation	Aucune appartenance	Andrews	Scott
New Democratic Party	Nouveau Parti démocratique	Baldwin	Jeannie
Conservative Party of Canada	Parti conservateur du Canada	Barnett	Lorraine E.
Green Party of Canada	Le Parti Vert du Canada	Byrne-Puumala	Krista
Forces et Démocratie	Forces et Démocratie	McCreath	Jennifer
Liberal Party of Canada	Parti libéral du Canada	McDonald	Ken
New Democratic Party	Nouveau Parti démocratique	Brown	Jenn
Green Party of Canada	Le Parti Vert du Canada	Colbourne	Tyler John
Liberal Party of Canada	Parti libéral du Canada	Foote	Judy M.
Conservative Party of Canada	Parti conservateur du Canada	Windsor	Mike
New Democratic Party	Nouveau Parti démocratique	Menchenton	Claudette
Conservative Party of Canada	Parti conservateur du Canada	O'Brien	Kevin George
Green Party of Canada	Le Parti Vert du Canada	Perry	Elizabeth
Liberal Party of Canada	Parti libéral du Canada	Simms	Scott
Liberal Party of Canada	Parti libéral du Canada	Jones	Yvonne
Conservative Party of Canada	Parti conservateur du Canada	Penashue	Peter
New Democratic Party	Nouveau Parti démocratique	Rudkowski	Edward
New Democratic Party	Nouveau Parti démocratique	Babstock	Devon
Green Party of Canada	Le Parti Vert du Canada	Cormier	Terry
Liberal Party of Canada	Parti libéral du Canada	Hutchings	Gudie
Conservative Party of Canada	Parti conservateur du Canada	Ruth	Wayne
Communist Party of Canada	Parti communiste du Canada	Burton	Sean

```
pd.DataFrame()
```

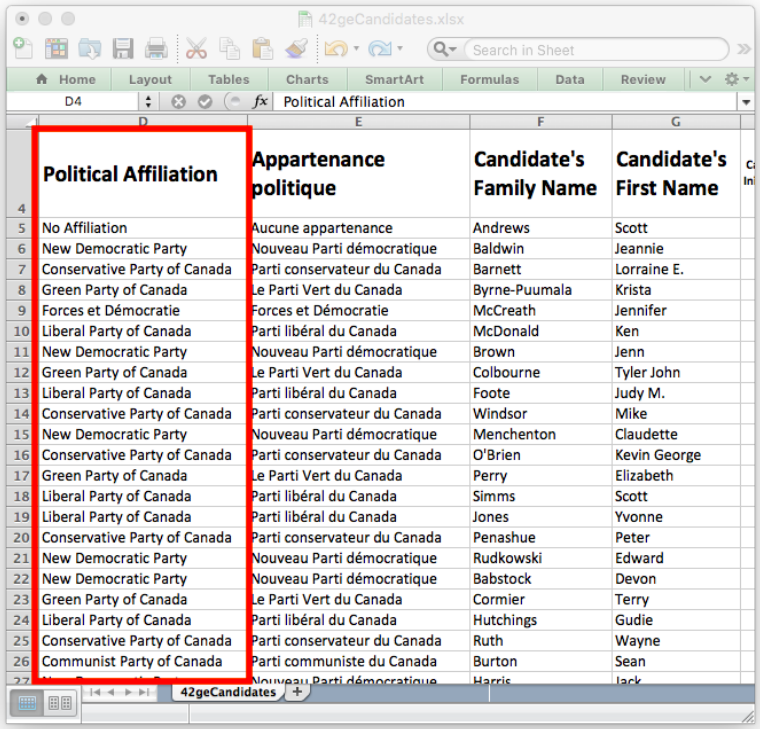
DataFrame: Row



	D	E	F	G
4	**Political Affiliation**	**Appartenance politique**	**Candidate's Family Name**	**Candidate's First Name**
5	No Affiliation	Aucune appartenance	Andrews	Scott
6	New Democratic Party	Nouveau Parti démocratique	Baldwin	Jeannie
7	Conservative Party of Canada	Parti conservateur du Canada	Barnett	Lorraine E.
8	Green Party of Canada	Le Parti Vert du Canada	Byrne-Puumala	Krista
9	Forces et Démocratie	Forces et Démocratie	McCreath	Jennifer
10	Liberal Party of Canada	Parti libéral du Canada	McDonald	Ken
11	New Democratic Party	Nouveau Parti démocratique	Brown	Jenn
12	Green Party of Canada	Le Parti Vert du Canada	Colbourne	Tyler John
13	Liberal Party of Canada	Parti libéral du Canada	Foote	Judy M.
14	Conservative Party of Canada	Parti conservateur du Canada	Windsor	Mike
15	New Democratic Party	Nouveau Parti démocratique	Menchenton	Claudette
16	Conservative Party of Canada	Parti conservateur du Canada	O'Brien	Kevin George
17	Green Party of Canada	Le Parti Vert du Canada	Perry	Elizabeth
18	Liberal Party of Canada	Parti libéral du Canada	Simms	Scott
19	Liberal Party of Canada	Parti libéral du Canada	Jones	Yvonne
20	Conservative Party of Canada	Parti conservateur du Canada	Penashue	Peter
21	New Democratic Party	Nouveau Parti démocratique	Rudkowski	Edward
22	New Democratic Party	Nouveau Parti démocratique	Babstock	Devon
23	Green Party of Canada	Le Parti Vert du Canada	Cormier	Terry
24	Liberal Party of Canada	Parti libéral du Canada	Hutchings	Gudie
25	Conservative Party of Canada	Parti conservateur du Canada	Ruth	Wayne
26	Communist Party of Canada	Parti communiste du Canada	Burton	Sean

```
df = pd.DataFrame(...)
df.ix[6]
```

DataFrame: Column



Political Affiliation	Appartenance politique	Candidate's Family Name	Candidate's First Name
No Affiliation	Aucune appartenance	Andrews	Scott
New Democratic Party	Nouveau Parti démocratique	Baldwin	Jeannie
Conservative Party of Canada	Parti conservateur du Canada	Barnett	Lorraine E.
Green Party of Canada	Le Parti Vert du Canada	Byrne-Puumala	Krista
Forces et Démocratie	Forces et Démocratie	McCreath	Jennifer
Liberal Party of Canada	Parti libéral du Canada	McDonald	Ken
New Democratic Party	Nouveau Parti démocratique	Brown	Jenn
Green Party of Canada	Le Parti Vert du Canada	Colbourne	Tyler John
Liberal Party of Canada	Parti libéral du Canada	Foote	Judy M.
Conservative Party of Canada	Parti conservateur du Canada	Windsor	Mike
New Democratic Party	Nouveau Parti démocratique	Menchenton	Claudette
Conservative Party of Canada	Parti conservateur du Canada	O'Brien	Kevin George
Green Party of Canada	Le Parti Vert du Canada	Perry	Elizabeth
Liberal Party of Canada	Parti libéral du Canada	Simms	Scott
Liberal Party of Canada	Parti libéral du Canada	Jones	Yvonne
Conservative Party of Canada	Parti conservateur du Canada	Penashue	Peter
New Democratic Party	Nouveau Parti démocratique	Rudkowski	Edward
New Democratic Party	Nouveau Parti démocratique	Babstock	Devon
Green Party of Canada	Le Parti Vert du Canada	Cormier	Terry
Liberal Party of Canada	Parti libéral du Canada	Hutchings	Gudie
Conservative Party of Canada	Parti conservateur du Canada	Ruth	Wayne
Communist Party of Canada	Parti communiste du Canada	Burton	Sean

```
df = pd.DataFrame(...)
df['Political Affiliation']
```