



Welcome to DSCI 101

Introduction to Data Science



Week 2 Recap

- What type of questions can be answered using data science?
 - Question formulation – 3 types of questions
- Where do data come from?
 - Observational studies vs. randomized experiments
- Data basic concepts
 - data structure and variable types
 - population and random sample
 - sampling bias in surveys



Week 3 Preview

- Introduction to Python programming
 - Python
 - Jupyter notebook
- Basic data type in Python
 - Numpy
 - Pandas
- Database overview
 - RDBMS
 - SQL
- Pandas (and the next 2 weeks)
 - tabular data in Python
 - data manipulation and summary

Intro to pythonTM

- What is Python?
 - open source and free!
 - huge user community
- What can you do with it?
 - multi purpose: software design, data analysis, AI/ML, Apps, testing, hacking...
- Why is it so popular?
 - the world's fastest growing programming language
 - simple and beginner-friendly syntax

Basic Concepts

- **computer program:**

- a set of lines of code executing one at a time
- each line of code contains a single instruction



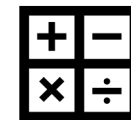
- **variable:**

- a named item to store a value
- created by performing an **assignment**



- **function:**

- pre-defined command that performs computations
- parameter, argument, and return



Python modules

- Modules are organized collections of functions
- For example, math module contains useful functions and operations
 - You have to import the module first in order to use it!
 - ***import math***
 - ***math.sqrt(9)*** returns 3
- Methods are functions that operates on a particular type of object
 - ***object.method(more parameters)***

Python modules for Data Science

NumPy 

 **pandas**

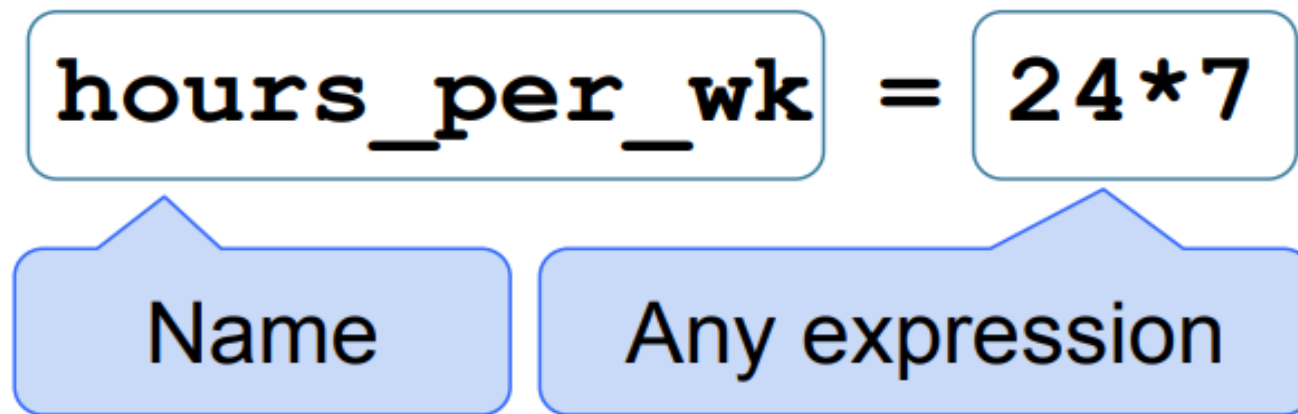
matplotlib 

 **seaborn**

 **statsmodels**

 **scikit
learn**

Assignment statements



- Left of = symbol is the name of the variable, and right of = is the value
- This statement assigns the value to the name
- Statements don't have a value; they perform an action

Rules for Python variable name

- must start with a letter or the underscore character
- cannot start with a number
- can only contain alpha-numeric characters and underscores
 - A-z, 0-9, and _
- are case-sensitive
 - age, Age and AGE are three different variables!



A variable can have:

- ***a short name (like x and y)***

or

- ***a more descriptive name (age, carname, total_volume)***

Balance between code conciseness and readability

Call expression

- You can call a variable:
 - Print the value assigned to that variable

What
function
to call

Argument to the
function

`f(27)`

- You can call a function:
 - function takes in arguments
 - output a return value

"Call f on 27."

What
function
to call

First argument

Second
argument

- You can call a method:
 - method is also a function

`max(15, 27)`

Arithmetic operations

Operation	Operator	Example	Value
Addition	+	$2 + 3$	5
Subtraction	-	$2 - 3$	-1
Multiplication	*	$2 * 3$	6
Division	/	$7 / 3$	2.66667
Remainder	%	$7 \% 3$	1
Exponentiation	**	$2 ** 0.5$	1.41421

Data types in Python

- Numbers
 - int: an integer of any size
 - float: a number with optional fractional parts
- Strings
 - a string is a snippet of text of any length
 - always in “string” or ‘string’
 - string methods: ***str.lower***, ***str.replace('i', 'text')*** and many more
- Numbers and strings can be converted
 - `str(123) → '123'`
 - `int('12') → 12` , `float('1.2') → 1.2`

Data types in Python

- **Boolean:** **True**, **False**
 - reserved words in Python
 - can be automatically convert to 1 and 0
- **Container**
 - each container consists of multiple elements
 - an **element** is a value of any type
 - 4 built-in containers: **list**, tuple, set, and **dictionary**
 - container in Numpy module: **array**
 - container in Pandas module: **dataframe**

Numpy array

- Arrays: a sequence of values organized by shape
 - 1d array -> vector, 2d array -> matrix, 3+d array-> tensor...
 - all elements of an array should have the **same** data type
 - arithmetic is applied to each element individually
 - elementwise operations can be done on arrays of the same shape
- Special 1d array: `np.arange(start, end, step)`
 - a range with step between consecutive values from start to end
 - always includes the start but excludes the end

List, dictionary and dataframe

- List = [element1, element2, ...]
 - a sequence of values (just like an array) that can have different data types!
 - defined by square brackets
- Dictionary = {key1: value1, key2: value2, ...}
 - defined by curly braces and key : value pairs
- A dataframe is a tabular data set
 - a sequence of labeled columns
 - each column represents one attribute of the individuals / feature
 - each row represents one individual / observation

Container in Python

- **Ordered:** elements have specific order
- **Indexed:** can access particular element(s) by its index(es)
- **Mutable vs Immutable:** whether elements can be changed

Container	Ordered	Indexed	Mutable
Array	Yes	Yes	Yes
List	Yes	Yes	Yes
Tuple	Yes	Yes	No
Set	No	No	Yes
Dictionary	No	Yes	Yes

Function in Python

- Remember a Python module is an organized collections of functions and methods, so you will be using lots of predefined functions.
- At some point, you will need to write a function yourself
 - a user defined function
 - a chunk of code you can reuse
 - **good practice in coding: modular and reusable**

A function in Python

Function header

```
def function_name (inputs)
```

Parameter(s)

```
    blah blah...
```

Return

Function
body

```
    return outputs
```

```
def double (x)
```

```
    return x*2
```

Function call

```
double (x = 3.14)
```

argument

More on Function

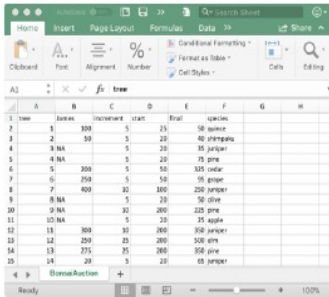
- Name scope:
 - **scope** of a variable or function name is the portion of a program in which the name is valid
- Anything defined inside a function is **local**
- Anything defined outside a functions is **global**
- Consequences:
 - parameters are local
 - a function call cannot precede the function header
 - functions defined inside a function is local
 - since function scope begins at the header, **a function can call itself**

More on Function

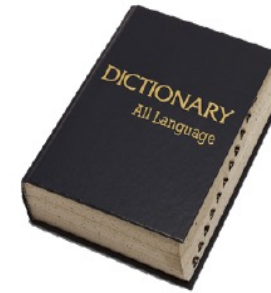
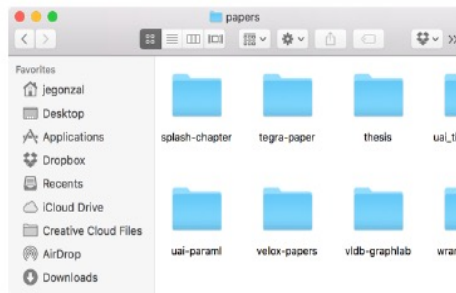
- Arguments --> parameters by either position or keyword:
 - *my_function(arg1, arg2, ...)* has to be in correct order
 - *my_function(par1 = arg1, par2 = arg2)* same as *my_function(par2 = arg2, par1 = arg1)* recommended less error-prone
- A default parameter is followed by a default value:
 - *Def my_function(par_default = arg_default)*
 - call *my_function()* using default value
 - call *my_function(par_default = arg)* using specified value

Brief database overview

- A database is an organized collection of data.



	A	B	C	D	E	F	G	H
1	Year	Species	Increment	Start	End	Species		
2	1	100	5	25	50	species		
3	2	50	5	25	50	species		
4	3	NA	5	25	50	species		
5	4	NA	5	25	50	species		
6	5	300	5	50	100	species		
7	6	250	5	50	100	species		
8	7	400	30	180	220	species		
9	8	NA	5	25	50	species		
10	9	NA	30	280	320	species		
11	10	NA	5	25	50	species		
12	11	300	30	280	320	species		
13	12	250	25	280	320	species		
14	13	275	25	280	320	species		
15	14	20	5	25	50	species		



- A relational database management system (RDBMS) is a software system that stores, manages and facilitates access to databases.



Why use RDBMS?

- Data is not always stored in a simple-to-read format such as a CSV,
 - data in the real world is messy...
- Data storage:
 - reliable storage to survive system crashes and disk failures
 - optimize to compute on data that does not fit in memory
 - special data structures to improve performance
- Data management:
 - configure how data is locally organized and who has access
 - enforce guarantees, prevent data anomalies, ensure safe concurrent operations...

RDBMS terminology

- In a relational database, each data table is called a relation
- Each row of table is called a record or tuple
- Each column of table is called an attribute or field, and has:
 - a name
 - a data type
 - maybe some constraints
- For each table, one attribute is called primary key
 - uniquely identify each record in this table
 - can be used to link tables together

SQL-how you interact with RDBMS

- Structured Query Language
 - domain-specific: manage data in RDBMS
 - declarative: say what you want, not how to get it
- Advantages:
 - enable the system to find the best way to achieve the result
 - more compact and easier to learn for non-programmers
- Challenges:
 - performance depends heavily on automatic optimization
 - limited language

Common DBMS systems

Rank			DBMS	Database Model	Score		
Jan 2020	Dec 2019	Jan 2019			Jan 2020	Dec 2019	Jan 2019
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1346.68	+0.29	+77.85
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1274.65	-1.01	+120.39
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	1098.55	+2.35	+58.29
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	507.19	+3.82	+41.08
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	426.97	+5.85	+39.78
6.	6.	6.	IBM Db2 +	Relational, Multi-model ⓘ	168.70	-2.65	-11.15
7.	7.	↑ 8.	Elasticsearch +	Search engine, Multi-model ⓘ	151.44	+1.19	+8.00
8.	8.	↓ 7.	Redis +	Key-value, Multi-model ⓘ	148.75	+2.51	-0.27
9.	9.	9.	Microsoft Access	Relational	128.58	-0.89	-13.04
10.	↑ 11.	10.	SQLite +	Relational	122.14	+1.78	-4.66


The 4 most popular SQL RDBMS implementations

I have some experience with these two

SQL syntax

SELECT [**DISTINCT**] *<column expression list>*
FROM *<list of tables>*
[**WHERE** *<predicate>*]
[**GROUP BY** *<column list>*]
[**HAVING** *<predicate>*]
[**ORDER BY** *<column list>*]
[**LIMIT** *<number of rows>*];



- We will not cover SQL in this class  DSCI 302
- Python Pandas and R dplyr are both based on SQL-like queries

Brainstorm

- How would you design a database for Rice?

- Tables for people:



- Tables for classes:



- Tables for activities:



- ...