



INMAS

Real Data

- Storing Data
 - File Systems
- Flat Files
- Real Data

James Balamuta

Inmas Fall 2021 Statistical Methods Workshop



Lecture Objectives

- *Describe* data storage options on disk.
- *Review* file system logic.
- *Manipulate* data with pandas.
- *Emphasize* how data can be missing.

Storing Data

How does a computer
store data?

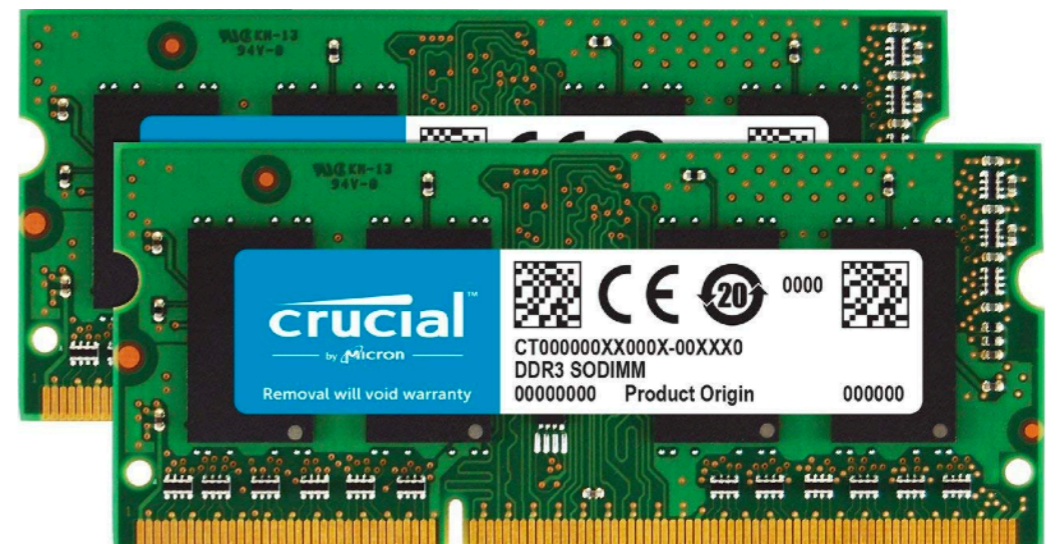
Definition:

Random Access Memory (RAM) stores data for short-term usage on a per-session basis.

Note: *Python* stores data or objects found in the global environment in memory.



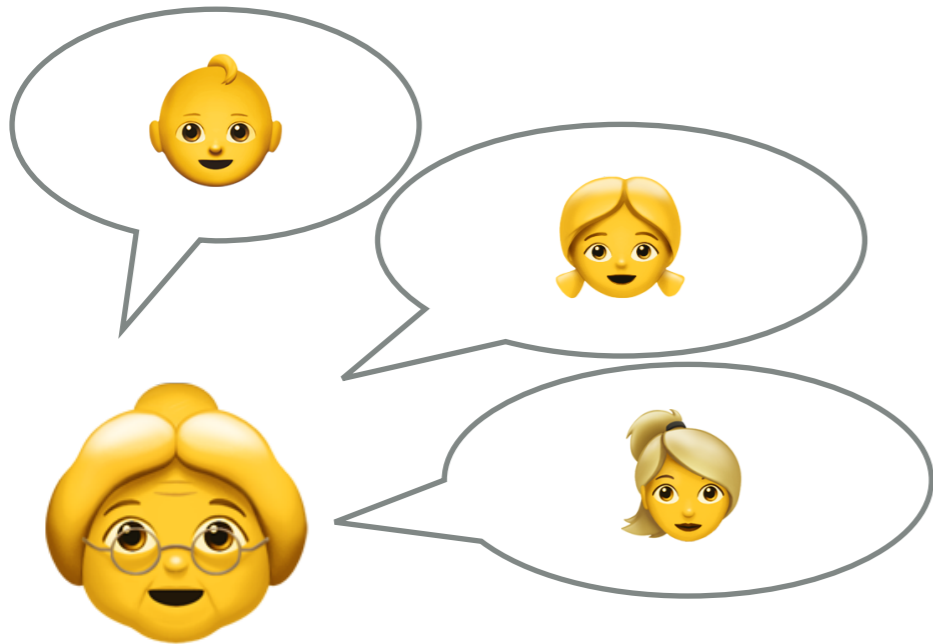
$3 \times 3 = 9$



[Source](#)

Definition:

*Hard Drives (HDs) and Solid State Drives (SSDs) store data for the **long term** (e.g. months from now).*



"Memories"



Source



Source

Definition:

Storage size refers to the amount of space required to hold the data contents on a computer.

Traditional units					Decimal for comparison				
Name	IEC	Binary	Number of bytes	Equal to	Name	IEC	Decimal	Number of bits	Equal to
Kilobyte	KiB	2 ¹⁰	1,024	1024 B	Kilobit	kbit	10 ³	1,000	1000 bit
Megabyte	MiB	2 ²⁰	1,048,576	1024 KiB	Megabit	Mbit	10 ⁶	1,000,000	1000 kbit
Gigabyte	GiB	2 ³⁰	1,073,741,824	1024 MiB	Gigabit	Gbit	10 ⁹	1,000,000,000	1000 Mbit
Terabyte	TiB	2 ⁴⁰	1,099,511,627,776	1024 GiB	Terabit	Tbit	10 ¹²	1,000,000,000,000	1000 Gbit
Petabyte	PiB	2 ⁵⁰	1,125,899,906,842,624	1024 TiB	Petabit	Pbit	10 ¹⁵	1,000,000,000,000,000	1000 Tbit
Exabyte	EiB	2 ⁶⁰	1,152,921,504,606,846,976	1024 PiB	Exabit	Ebit	10 ¹⁸	1,000,000,000,000,000,000	1000 Pbit
Zettabyte	ZiB	2 ⁷⁰	1,180,591,620,717,411,303,424	1024 EiB	Zettabit	Zbit	10 ²¹	1,000,000,000,000,000,000,000	1000 Ebit
Yottabyte	YiB	2 ⁸⁰	1,208,925,819,614,629,174,706,176	1024 ZiB	Yottabit	Ybit	10 ²⁴	1,000,000,000,000,000,000,000,000	1000 Zbit

Byte: 8 bits per byte
Large (2⁸)

bit: binary digit
Small

WHAT IS A PETABYTE?

TO UNDERSTAND A PETABYTE WE
MUST FIRST UNDERSTAND A
GIGABYTE.

1
GIGABYTE

▪ 7 MINUTES OF
HD-TV VIDEO

2
GIGABYTES

▪ 20 YARDS OF BOOKS ON
A SHELF

4.7
GIGABYTES

▪ SIZE OF A STANDARD
DVD-R

THERE ARE A MILLION GIGABYTES
IN A PETABYTE



(1024 MEGABYTES)

1 GIGABYTE

[Source](#)

Definition:

File Formats are standardized ways that data may be stored on a computer.



.xlsx



.docx



.pdf

Types of Data Formats

Pandas IO Tools

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	Fixed-Width Text File	read_fwf	
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	LaTeX		Styler.to_latex
text	XML	read_xml	to_xml
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	SPSS	read_spss	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google BigQuery	read_gbq	to_gbq

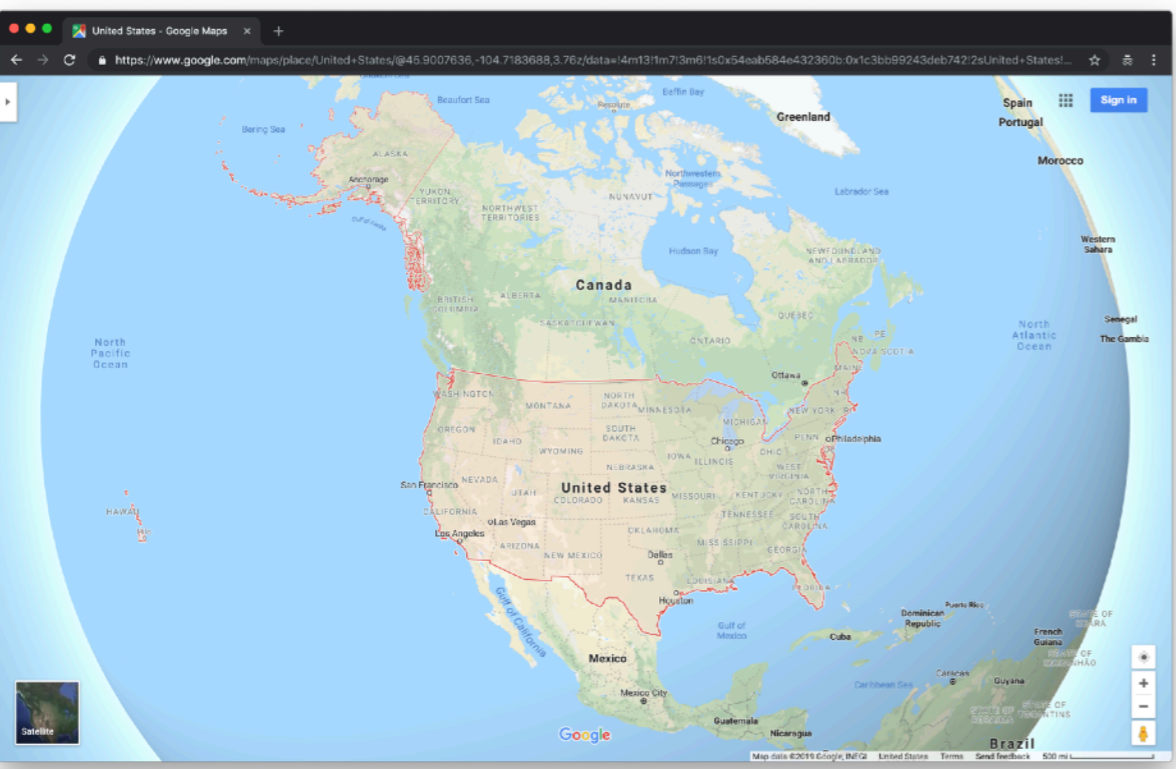
File Systems

How is
data organized
on a computer?

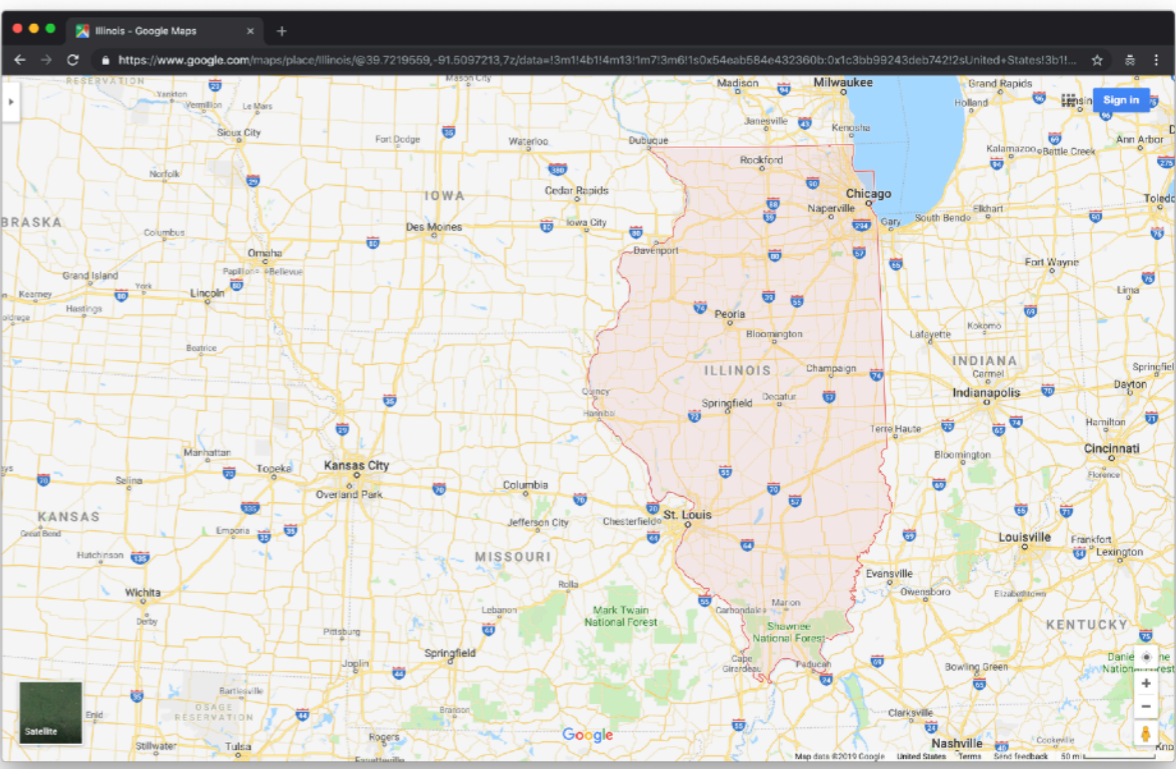
Definition:

Directories or folders acts as a catalog that contains content in a way similar to a file cabinet.

World > United States



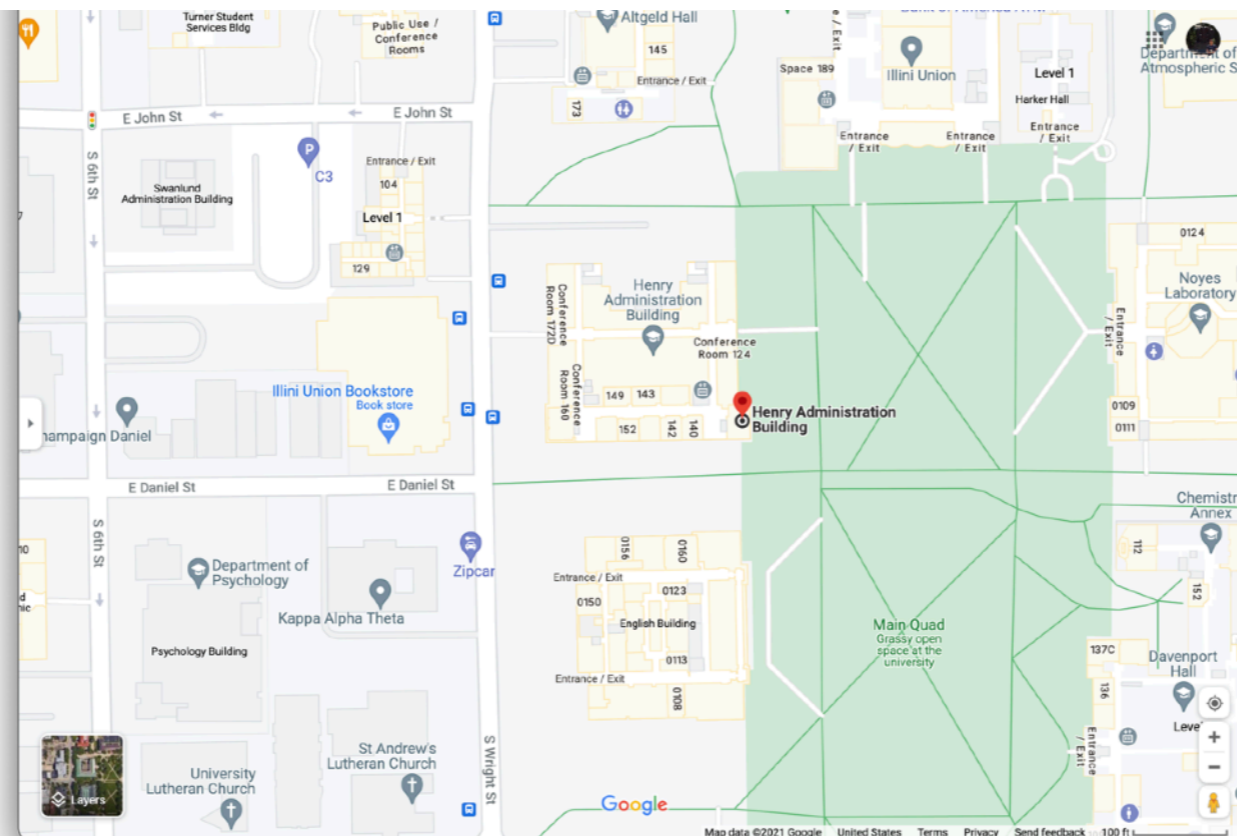
World > United States > IL



Definition:

File Path or *Path* provides a unique address to access content on the computer.

World > United States > IL > 61820 > 505 S. Wright St.

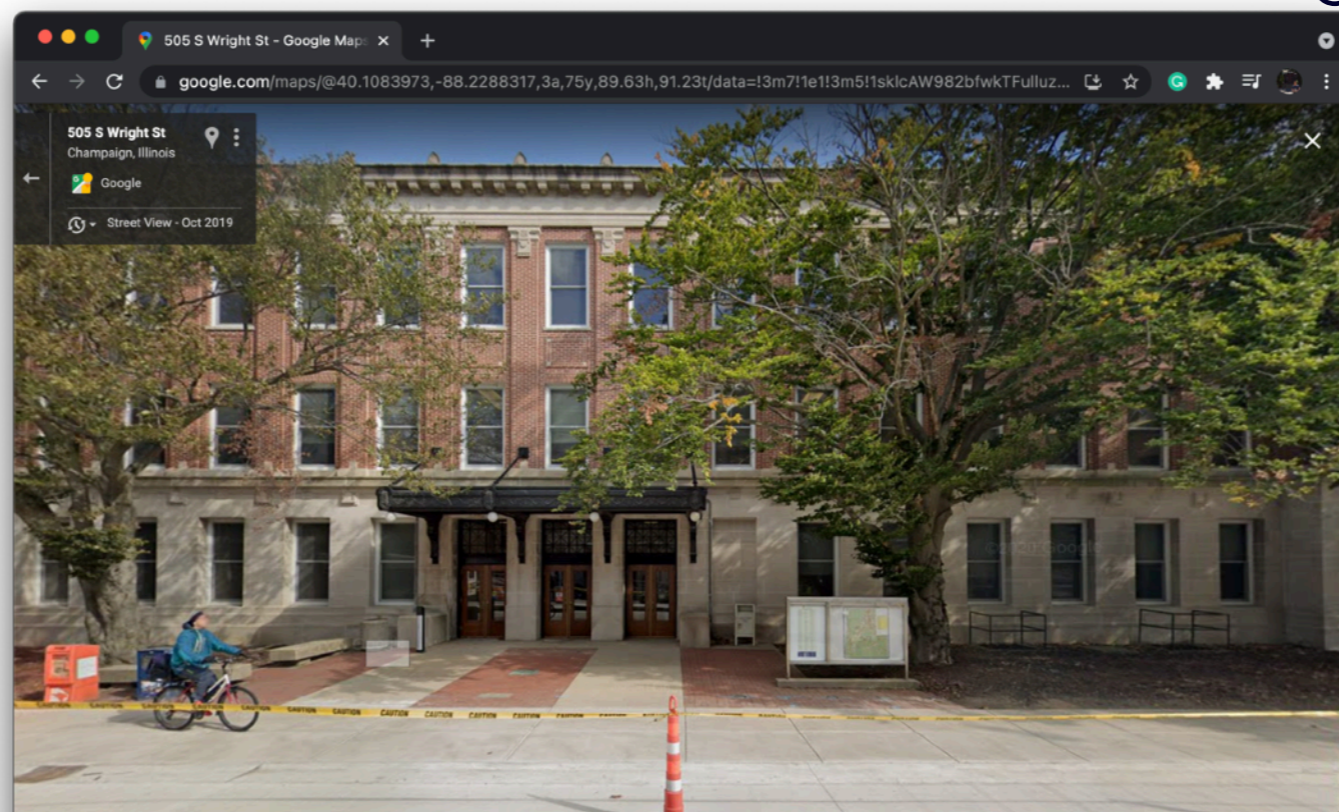


Definition:

A *File* contains the content found the end of the path.

Contents of:

World > United States > IL > 61820 > 505 S. Wright St.



Working Directories

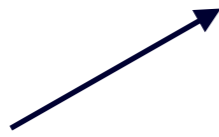
... where you are ...

```
# Retrieve working directory
import os

current_path = os.getcwd()
print(current_path)
# [1] "/content"
```



You are here



Definition:

Fixed or Absolute paths are computer/user specific based on the root (/) directory.

C:/Users/James/Hypno/Toad.py

/etc/python/hosts

/Users/James/URA/reports/summary.docx



Definition:

Relative paths resolve from the present working directory location

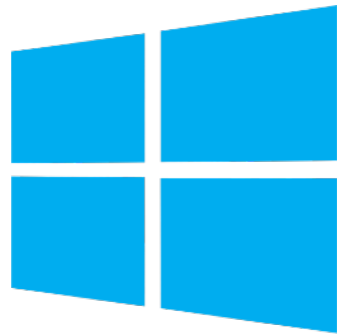
Hypno/Toad.py

hosts

reports/summary.docx

Absolute Paths by OS

... identifying operating systems (OS) by path prefixes ...



C:/Users/balamut2



/Users/balamut2



/home/balamut2



/content/drive/My\ Drive

Your Turn

Consider a file located at:

C:/Users/James/Hypno/Toad.py

1. What kind of path is it?
2. Which operating system is the path on?
3. What would be the relative path if the working directory is James?

Reading Flat Files

Definition:

Flat Files store data in text that may or may not easily be human readable. However, changes between data versions can be detected under a version control system.

id,sex,height
1,M,6.1
2,F,5.5
3,F,5.2
...
55,M,5.9

Line	Original File	Modified File
...	@@ -1,144 +1,146 @@	
1	-term,unit,lname,fname,role,ranking,courses	+term,unit,lname,fname,role,ranking,course
2	fa1993,"ACCOUNTANCY",DIETRICH,R,Instructor,Excellent,201	fa1993,"ACCOUNTANCY",DIETRICH,R,Instructor,Excellent,201
3	-fa1993,"ACCOUNTANCY",ONES,R,TA,Outstanding,311	+fa1993,"ACCOUNTANCY",JONES,R,TA,Outstanding,311
4	fa1993,"ACCOUNTANCY",MOLLOY,K,Instructor,Excellent,456	fa1993,"ACCOUNTANCY",MOLLOY,K,Instructor,Excellent,456
5	fa1993,"ACCOUNTANCY",MOLLOY,K,Instructor,Excellent,405	fa1993,"ACCOUNTANCY",MOLLOY,K,Instructor,Excellent,405
6	fa1993,"ACCOUNTANCY",OMER,T,Instructor,Excellent,251	fa1993,"ACCOUNTANCY",OMER,T,Instructor,Excellent,251
7	fa1993,"ACCOUNTANCY",SCHOENFELD,H,Instructor,Excellent,401	fa1993,"ACCOUNTANCY",SCHOENFELD,H,Instructor,Excellent,401
8	-fa1993,"ACCOUNTANCY",YATT,A,Instructor,Outstanding,211	+fa1993,"ACCOUNTANCY",WYATT,A,Instructor,Outstanding,211
9	-fa1993,"ACCOUNTANCY",IEBART,D,Instructor,Outstanding,494	+fa1993,"ACCOUNTANCY",ZIEBART,D,Instructor,Outstanding,494
10	fa1993,"ACCOUNTANCY",ZIEGLER,R,Instructor,Excellent,341	fa1993,"ACCOUNTANCY",ZIEGLER,R,Instructor,Excellent,341
11	fa1993,"ADMINISTRATION,HIGHER,&CONTINUING EDUCATION",MCGREAL,T,Instructor,Excellent,438	fa1993,"ADMINISTRATION,HIGHER,&CONTINUING EDUCATION",MCGREAL,T,Instructor,Excellent,438
12	fa1993,"ADMINISTRATION,HIGHER,&CONTINUING EDUCATION",MERCHANT,B,Instructor,Excellent,461	fa1993,"ADMINISTRATION,HIGHER,&CONTINUING EDUCATION",MERCHANT,B,Instructor,Excellent,461
13	-fa1993,"ADMINISTRATION,HIGHER,&CONTINUING EDUCATION",RESTINE,N,Instructor,Outstanding,490	+fa1993,"ADMINISTRATION,HIGHER,&CONTINUING EDUCATION",PRESTINE,N,Instructor,Outstanding,490
14	-fa1993,"ADVERTISING",ENNETT,R,TA,Outstanding,382	+fa1993,"ADVERTISING",BENNETT,R,TA,Outstanding,382
15	-fa1993,"ADVERTISING",ORNELIUS,V,TA,Outstanding,382	+fa1993,"ADVERTISING",CORNELIUS,V,TA,Outstanding,382
16	fa1993,"ADVERTISING",OTNES,C,Instructor,Excellent,382	fa1993,"ADVERTISING",OTNES,C,Instructor,Excellent,382
17	fa1993,"ADVERTISING",ROTZOLL,K,Instructor,Excellent,450	fa1993,"ADVERTISING",ROTZOLL,K,Instructor,Excellent,450
18	fa1993,"ADVERTISING",SCOTT,L,Instructor,Excellent,490	fa1993,"ADVERTISING",SCOTT,L,Instructor,Excellent,490
19	fa1993,"ADVERTISING",SCOTT,L,Instructor,Excellent,393	fa1993,"ADVERTISING",SCOTT,L,Instructor,Excellent,393
20	-fa1993,"ADVERTISING",HELDON,P,TA,Outstanding,382	+fa1993,"ADVERTISING",SHELDON,P,TA,Outstanding,382

Flat File "diff"

CSV Data

... “**C**omma” **S**eparated **V**alues (**CSV**) ...

subject_heights.csv



Header

id,sex,height

Row

1,M,6.1

2,F,5.5

3,F,5.2

...

55,M,5.9

Comma / Separator

Column

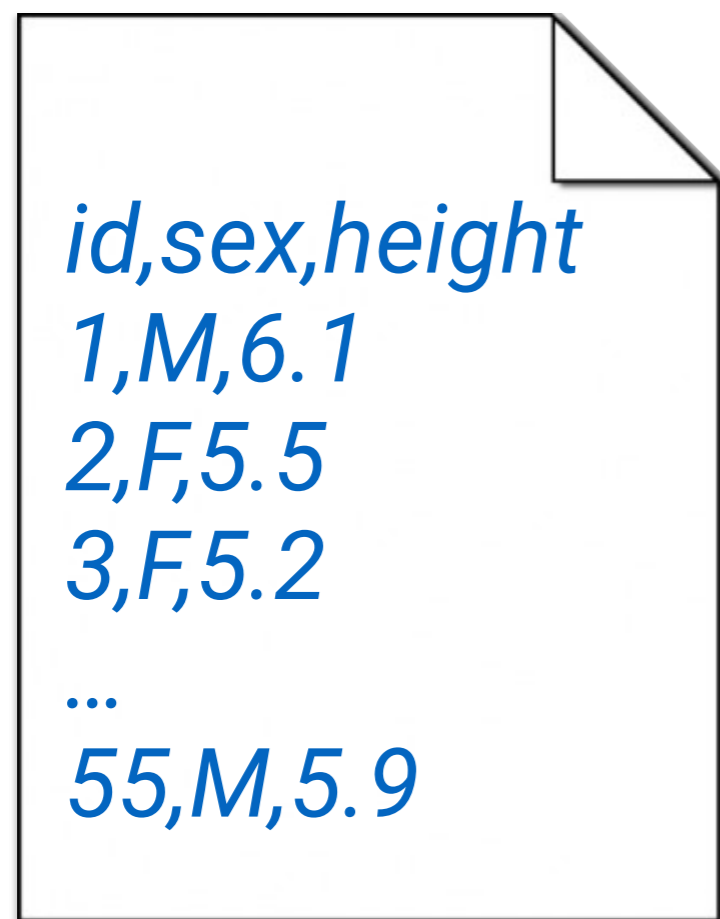
	A	B	C	D	E	F	G	H
1	id	sex	height					
2	1	M	6.1					
3	2	F	5.5					
4	3	F	5.2					
5	4	M	4.2311897					
6	5	M	6.22808397					
7	6	M	4.38029873					
8	7	F	5.23898598					
9	8	M	7.54548871					
10	9	F	4.88778891					
11	10	F	4.76057357					
12	11	M	5.26893677					
13	12	M	3.85277151					
14	13	F	3.47028748					
15	14	F	3.97996808					
16	15	M	3.98189647					

Column

Reading a CSV into *Python*

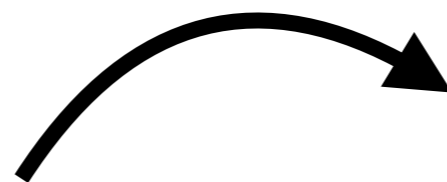
... loading data ...

```
subject_heights = pd.read_csv("subject_heights.csv")
```



id,sex,height
1,M,6.1
2,F,5.5
3,F,5.2
...
55,M,5.9

subject_heights.csv



id	sex	height
1	M	6.1
2	F	5.5
3	F	5.2
...
55	M	5.9

subject_heights

American vs. European

... differing styles with the same extension ...

subject_heights.csv

American

```
id,sex,height
1,M,6.1
2,F,5.5
3,F,5.2
...
55,M,5.9
```

```
ds = pd.read_csv("subject_heights.csv",
                 sep = ",")
```

subject_heights2.csv

European


```
id;sex;height
1;M;6,1
2;F;5,5
3;F;5,2
...
55;M;5,9
```

```
ds = pd.read_csv("subject_heights2.csv",
                 sep = ";")
```

“Comma” Separated Values (CSV)

Delimited Data

... custom separator between variables ...



```
id|sex|height  
1|M|6.1  
2|F|5.5  
3|F|5.2  
  
...  
55|M|5.9
```

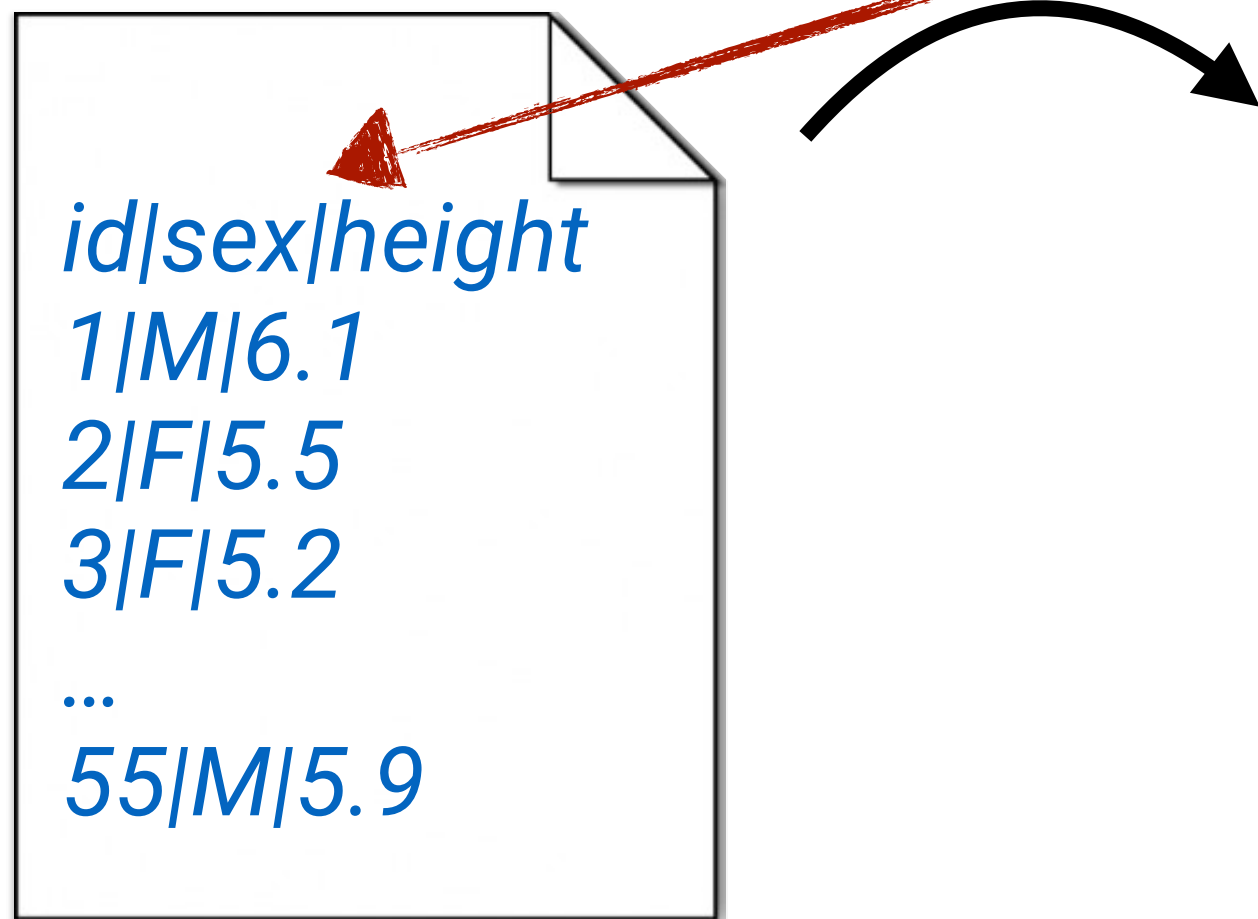
subject_heights.txt

Delimited Text File (**TXT**)

Reading a Delimited File

... loading data with a custom separator ...

```
subject_heights = pd.read_csv("subject_heights.txt",  
                             delimiter = "|")
```



id|sex|height
1|M|6.1
2|F|5.5
3|F|5.2

...
55|M|5.9

subject_heights.txt

id	sex	height
1	M	6.1
2	F	5.5
3	F	5.2
...
55	M	5.9

subject_heights

Your Turn

Consider the following files, what kind of separator or delimiter should be used to read the data into Python?

*day*month*count*

*21*May*50*

*22*May*32*

*23*May*5*

*24*May*8*

*25*May*25*

...

*1*June*130*

index;value

DJI;31,032

DJI;32,257

DJI;33,141

...

DJI;32,921

Reading a CSV from a URL

Imports CSV as DataFrame and Set Variable Names

```
# Read in Data
```

```
uci_adult_df = pd.read_csv(  
    "https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data",  
    sep=";",          # Explicitly say the format.  
    header = None,    # Indicate column names are missing  
    na_values=['NA','?']) # Set NA values
```

```
# Add column names
```

```
uci_adult_df.columns = [  
    "Age", "WorkClass", "fnlwgt", "Education", "EducationNum",  
    "MaritalStatus", "Occupation", "Relationship", "Race", "Gender",  
    "CapitalGain", "CapitalLoss", "HoursPerWeek", "NativeCountry", "Income"  
]
```

Displaying Data

```
   Age      WorkClass  fnlwgt  ...  HoursPerWeek  NativeCountry  Income
0    39      State-gov   77516  ...      40  United-States  <=50K
1    50  Self-emp-not-inc  83311  ...      13  United-States  <=50K
2    38      Private   215646  ...      40  United-States  <=50K
3    53      Private   234721  ...      40  United-States  <=50K
4    28      Private   338409  ...      40      Cuba    <=50K
...  ...      ...      ...  ...      ...      ...      ...
32556 27      Private   257302  ...      38  United-States  <=50K
32557 40      Private   154374  ...      40  United-States  >50K
32558 58      Private   151910  ...      40  United-States  <=50K
32559 22      Private   201490  ...      20  United-States  <=50K
32560 52  Self-emp-inc   287927  ...      40  United-States  >50K
```

[32561 rows x 15 columns]

	Age	WorkClass	fnlwgt	Education	EducationNum	MaritalStatus	Occupation
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty
...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial

32561 rows x 15 columns

```
# See raw data form
print(uci_adult_df)
```

```
# View formatted data
display(uci_adult_df)
```

```
# What is different between
# the two views?
```

Binary Files

How can we
reduce stored data size?

Definition:

Binary Files save space by simplifying file contents under a pre-determined format. To re-open a file, you must translate it back.

A screenshot of a text editor window titled "erroRs.xlsx — Downloads". The window displays a file containing mostly null characters represented as "<NUL>". There are two lines of text that appear to be fragments of XML or binary data. The first line starts with "1 PK^C^D^T<NUL>^F<NUL><BS><NUL><NUL><NUL>!<NUL>t6Z;z^A<NUL><NUL>^E<NUL><NUL>^S<NUL><BS>^B [Content_Types] ." and ends with "...<NUL><NUL>". The second line starts with "2 y♦3♦Ã♦Hw♦J♦I♦Â♦°♦q♦i♦0♦ú♦?<NP>♦i♦É♦ÿ♦v♦q♦i♦ü♦<FS>♦É♦Z♦(♦A♦*♦ñ♦♦j♦0♦»♦\;♦ù♦U♦0♦â♦3♦♦E♦y♦<FS>♦ä♦Ü♦ö♦ä♦T♦♦♦²♦p♦G♦+♦^♦Q♦p♦|♦^♦Y♦e♦z♦w♦V♦0♦ú♦É♦Ä♦'♦t♦^♦P♦x♦X♦E♦ü♦½♦\♦B♦♦B♦V♦¾♦ó♦«♦^♦K♦ø♦♦}♦B♦♦VN♦♦j♦.♦+♦'♦a♦♦{♦♦♦[♦z♦♦BD♦♦^♦Z♦-♦I♦^♦W♦ð♦0♦¢♦m♦t♦'♦2♦^♦P♦\$♦²♦°♦o♦0♦C♦Ã♦¾♦g♦ä♦s♦±♦c♦h♦g♦♦^♦A♦s♦♦[♦æ♦^♦Y♦:♦ø♦^♦F♦<NUL><NUL>♦ÿ♦ÿ♦^♦C♦<NUL>PK^C^D^" and ends with "...<BS>^B_rels/.rels". The status bar at the bottom shows "Line: 1 | Plain Text | Soft Tabs: 4 |".

https://github.com/pjmensink/errorR_dictionary/issues/1

Pickle or .pkl

Python's binary data file type

```
import pickle
...
with open('mydata.pickle', 'wb') as mysavedata:
    pickle.dump([1, 2, 'three'], mysavedata)
...
with open('mydata.pickle', 'rb') as myrestoredata:
    a_list = pickle.load(myrestoredata)
print(a_list)
```

Always remember to import the "pickle" module.

To save your data, use "dump()".

Assign your restored data to an identifier.

Once your data is back in your program, you can treat it like any other data object.

The "b" tells Python to open your data files in **BINARY** mode.

Restore your data from your file using "load()".

Excel's XLSX is Binary

... popular workbook format saves as a binary ...



```
subject_heights.xlsx — rstudio-export 2 Add License
1 PK^C^D^T<NUL>^F<NUL><BS><NUL><NUL><NUL>!  
  1 PK^C^D^T<NUL>^F<NUL><BS><NUL><NUL><NUL>!  
  2  
  3  
  4  
  5  
  6  
  7  
  8  
  9  
 10  
 11  
 12  
 13  
 14  
 15  
 16  
 17  
 18  
 19  
 20  
 21  
 22  
 23  
 24  
 25  
 26  
 27  
 28  
 29  
 30  
 31  
 32  
 33  
 34  
 35  
 36  
 37  
 38  
 39  
 40  
 41  
 42  
 43  
 44  
 45  
 46  
 47  
 48  
 49  
 50  
 51  
 52  
 53  
 54  
 55  
 56  
 57  
 58  
 59  
 60  
 61  
 62  
 63  
 64  
 65  
 66  
 67  
 68  
 69  
 70  
 71  
 72  
 73  
 74  
 75  
 76  
 77  
 78  
 79  
 80  
 81  
 82  
 83  
 84  
 85  
 86  
 87  
 88  
 89  
 90  
 91  
 92  
 93  
 94  
 95  
 96  
 97  
 98  
 99  
 100  
 101  
 102  
 103  
 104  
 105  
 106  
 107  
 108  
 109  
 110  
 111  
 112  
 113  
 114  
 115  
 116  
 117  
 118  
 119  
 120  
 121  
 122  
 123  
 124  
 125  
 126  
 127  
 128  
 129  
 130  
 131  
 132  
 133  
 134  
 135  
 136  
 137  
 138  
 139  
 140  
 141  
 142  
 143  
 144  
 145  
 146  
 147  
 148  
 149  
 150  
 151  
 152  
 153  
 154  
 155  
 156  
 157  
 158  
 159  
 160  
 161  
 162  
 163  
 164  
 165  
 166  
 167  
 168  
 169  
 170  
 171  
 172  
 173  
 174  
 175  
 176  
 177  
 178  
 179  
 180  
 181  
 182  
 183  
 184  
 185  
 186  
 187  
 188  
 189  
 190  
 191  
 192  
 193  
 194  
 195  
 196  
 197  
 198  
 199  
 200  
 201  
 202  
 203  
 204  
 205  
 206  
 207  
 208  
 209  
 210  
 211  
 212  
 213  
 214  
 215  
 216  
 217  
 218  
 219  
 220  
 221  
 222  
 223  
 224  
 225  
 226  
 227  
 228  
 229  
 230  
 231  
 232  
 233  
 234  
 235  
 236  
 237  
 238  
 239  
 240  
 241  
 242  
 243  
 244  
 245  
 246  
 247  
 248  
 249  
 250  
 251  
 252  
 253  
 254  
 255  
 256  
 257  
 258  
 259  
 260  
 261  
 262  
 263  
 264  
 265  
 266  
 267  
 268  
 269  
 270  
 271  
 272  
 273  
 274  
 275  
 276  
 277  
 278  
 279  
 280  
 281  
 282  
 283  
 284  
 285  
 286  
 287  
 288  
 289  
 290  
 291  
 292  
 293  
 294  
 295  
 296  
 297  
 298  
 299  
 300  
 301  
 302  
 303  
 304  
 305  
 306  
 307  
 308  
 309  
 310  
 311  
 312  
 313  
 314  
 315  
 316  
 317  
 318  
 319  
 320  
 321  
 322  
 323  
 324  
 325  
 326  
 327  
 328  
 329  
 330  
 331  
 332  
 333  
 334  
 335  
 336  
 337  
 338  
 339  
 340  
 341  
 342  
 343  
 344  
 345  
 346  
 347  
 348  
 349  
 350  
 351  
 352  
 353  
 354  
 355  
 356  
 357  
 358  
 359  
 360  
 361  
 362  
 363  
 364  
 365  
 366  
 367  
 368  
 369  
 370  
 371  
 372  
 373  
 374  
 375  
 376  
 377  
 378  
 379  
 380  
 381  
 382  
 383  
 384  
 385  
 386  
 387  
 388  
 389  
 390  
 391  
 392  
 393  
 394  
 395  
 396  
 397  
 398  
 399  
 400  
 401  
 402  
 403  
 404  
 405  
 406  
 407  
 408  
 409  
 410  
 411  
 412  
 413  
 414  
 415  
 416  
 417  
 418  
 419  
 420  
 421  
 422  
 423  
 424  
 425  
 426  
 427  
 428  
 429  
 430  
 431  
 432  
 433  
 434  
 435  
 436  
 437  
 438  
 439  
 440  
 441  
 442  
 443  
 444  
 445  
 446  
 447  
 448  
 449  
 450  
 451  
 452  
 453  
 454  
 455  
 456  
 457  
 458  
 459  
 460  
 461  
 462  
 463  
 464  
 465  
 466  
 467  
 468  
 469  
 470  
 471  
 472  
 473  
 474  
 475  
 476  
 477  
 478  
 479  
 480  
 481  
 482  
 483  
 484  
 485  
 486  
 487  
 488  
 489  
 490  
 491  
 492  
 493  
 494  
 495  
 496  
 497  
 498  
 499  
 500  
 501  
 502  
 503  
 504  
 505  
 506  
 507  
 508  
 509  
 510  
 511  
 512  
 513  
 514  
 515  
 516  
 517  
 518  
 519  
 520  
 521  
 522  
 523  
 524  
 525  
 526  
 527  
 528  
 529  
 530  
 531  
 532  
 533  
 534  
 535  
 536  
 537  
 538  
 539  
 540  
 541  
 542  
 543  
 544  
 545  
 546  
 547  
 548  
 549  
 550  
 551  
 552  
 553  
 554  
 555  
 556  
 557  
 558  
 559  
 560  
 561  
 562  
 563  
 564  
 565  
 566  
 567  
 568  
 569  
 570  
 571  
 572  
 573  
 574  
 575  
 576  
 577  
 578  
 579  
 580  
 581  
 582  
 583  
 584  
 585  
 586  
 587  
 588  
 589  
 590  
 591  
 592  
 593  
 594  
 595  
 596  
 597  
 598  
 599  
 600  
 601  
 602  
 603  
 604  
 605  
 606  
 607  
 608  
 609  
 610  
 611  
 612  
 613  
 614  
 615  
 616  
 617  
 618  
 619  
 620  
 621  
 622  
 623  
 624  
 625  
 626  
 627  
 628  
 629  
 630  
 631  
 632  
 633  
 634  
 635  
 636  
 637  
 638  
 639  
 640  
 641  
 642  
 643  
 644  
 645  
 646  
 647  
 648  
 649  
 650  
 651  
 652  
 653  
 654  
 655  
 656  
 657  
 658  
 659  
 660  
 661  
 662  
 663  
 664  
 665  
 666  
 667  
 668  
 669  
 670  
 671  
 672  
 673  
 674  
 675  
 676  
 677  
 678  
 679  
 680  
 681  
 682  
 683  
 684  
 685  
 686  
 687  
 688  
 689  
 690  
 691  
 692  
 693  
 694  
 695  
 696  
 697  
 698  
 699  
 700  
 701  
 702  
 703  
 704  
 705  
 706  
 707  
 708  
 709  
 710  
 711  
 712  
 713  
 714  
 715  
 716  
 717  
 718  
 719  
 720  
 721  
 722  
 723  
 724  
 725  
 726  
 727  
 728  
 729  
 730  
 731  
 732  
 733  
 734  
 735  
 736  
 737  
 738  
 739  
 740  
 741  
 742  
 743  
 744  
 745  
 746  
 747  
 748  
 749  
 750  
 751  
 752  
 753  
 754  
 755  
 756  
 757  
 758  
 759  
 760  
 761  
 762  
 763  
 764  
 765  
 766  
 767  
 768  
 769  
 770  
 771  
 772  
 773  
 774  
 775  
 776  
 777  
 778  
 779  
 780  
 781  
 782  
 783  
 784  
 785  
 786  
 787  
 788  
 789  
 790  
 791  
 792  
 793  
 794  
 795  
 796  
 797  
 798  
 799  
 800  
 801  
 802  
 803  
 804  
 805  
 806  
 807  
 808  
 809  
 810  
 811  
 812  
 813  
 814  
 815  
 816  
 817  
 818  
 819  
 820  
 821  
 822  
 823  
 824  
 825  
 826  
 827  
 828  
 829  
 830  
 831  
 832  
 833  
 834  
 835  
 836  
 837  
 838  
 839  
 840  
 841  
 842  
 843  
 844  
 845  
 846  
 847  
 848  
 849  
 850  
 851  
 852  
 853  
 854  
 855  
 856  
 857  
 858  
 859  
 860  
 861  
 862  
 863  
 864  
 865  
 866  
 867  
 868  
 869  
 870  
 871  
 872  
 873  
 874  
 875  
 876  
 877  
 878  
 879  
 880  
 881  
 882  
 883  
 884  
 885  
 886  
 887  
 888  
 889  
 890  
 891  
 892  
 893  
 894  
 895  
 896  
 897  
 898  
 899  
 900  
 901  
 902  
 903  
 904  
 905  
 906  
 907  
 908  
 909  
 910  
 911  
 912  
 913  
 914  
 915  
 916  
 917  
 918  
 919  
 920  
 921  
 922  
 923  
 924  
 925  
 926  
 927  
 928  
 929  
 930  
 931  
 932  
 933  
 934  
 935  
 936  
 937  
 938  
 939  
 940  
 941  
 942  
 943  
 944  
 945  
 946  
 947  
 948  
 949  
 950  
 951  
 952  
 953  
 954  
 955  
 956  
 957  
 958  
 959  
 960  
 961  
 962  
 963  
 964  
 965  
 966  
 967  
 968  
 969  
 970  
 971  
 972  
 973  
 974  
 975  
 976  
 977  
 978  
 979  
 980  
 981  
 982  
 983  
 984  
 985  
 986  
 987  
 988  
 989  
 990  
 991  
 992  
 993  
 994  
 995  
 996  
 997  
 998  
 999  
 1000  
 1001  
 1002  
 1003  
 1004  
 1005  
 1006  
 1007  
 1008  
 1009  
 1010  
 1011  
 1012  
 1013  
 1014  
 1015  
 1016  
 1017  
 1018  
 1019  
 1020  
 1021  
 1022  
 1023  
 1024  
 1025  
 1026  
 1027  
 1028  
 1029  
 1030  
 1031  
 1032  
 1033  
 1034  
 1035  
 1036  
 1037  
 1038  
 1039  
 1040  
 1041  
 1042  
 1043  
 1044  
 1045  
 1046  
 1047  
 1048  
 1049  
 1050  
 1051  
 1052  
 1053  
 1054  
 1055  
 1056  
 1057  
 1058  
 1059  
 1060  
 1061  
 1062  
 1063  
 1064  
 1065  
 1066  
 1067  
 1068  
 1069  
 1070  
 1071  
 1072  
 1073  
 1074  
 1075  
 1076  
 1077  
 1078  
 1079  
 1080  
 1081  
 1082  
 1083  
 1084  
 1085  
 1086  
 1087  
 1088  
 1089  
 1090  
 1091  
 1092  
 1093  
 1094  
 1095  
 1096  
 1097  
 1098  
 1099  
 1100  
 1101  
 1102  
 1103  
 1104  
 1105  
 1106  
 1107  
 1108  
 1109  
 1110  
 1111  
 1112  
 1113  
 1114  
 1115  
 1116  
 1117  
 1118  
 1119  
 1120  
 1121  
 1122  
 1123  
 1124  
 1125  
 1126  
 1127  
 1128  
 1129  
 1130  
 1131  
 1132  
 1133  
 1134  
 1135  
 1136  
 1137  
 1138  
 1139  
 1140  
 1141  
 1142  
 1143  
 1144  
 1145  
 1146  
 1147  
 1148  
 1149  
 1150  
 1151  
 1152  
 1153  
 1154  
 1155  
 1156  
 1157  
 1158  
 1159  
 1160  
 1161  
 1162  
 1163  
 1164  
 1165  
 1166  
 1167  
 1168  
 1169  
 1170  
 1171  
 1172  
 1173  
 1174  
 1175  
 1176  
 1177  
 1178  
 1179  
 1180  
 1181  
 1182  
 1183  
 1184  
 1185  
 1186  
 1187  
 1188  
 1189  
 1190  
 1191  
 1192  
 1193  
 1194  
 1195  
 1196  
 1197  
 1198  
 1199  
 1200  
 1201  
 1202  
 1203  
 1204  
 1205  
 1206  
 1207  
 1208  
 1209  
 1210  
 1211  
 1212  
 1213  
 1214  
 1215  
 1216  
 1217  
 1218  
 1219  
 1220  
 1221  
 1222  
 1223  
 1224  
 1225  
 1226  
 1227  
 1228  
 1229  
 1230  
 1231  
 1232  
 1233  
 1234  
 1235  
 1236  
 1237  
 1238  
 1239  
 1240  
 1241  
 1242  
 1243  
 1244  
 1245  
 1246  
 1247  
 1248  
 1249  
 1250  
 1251  
 1252  
 1253  
 1254  
 1255  
 1256  
 1257  
 1258  
 1259  
 1260  
 1261  
 1262  
 1263  
 1264  
 1265  
 1266  
 1267  
 1268  
 1269  
 1270  
 1271  
 1272  
 1273  
 1274  
 1275  
 1276  
 1277  
 1278  
 1279  
 1280  
 1281  
 1282  
 1283  
 1284  
 1285  
 1286  
 1287  
 1288  
 1289  
 1290  
 1291  
 1292  
 1293  
 1294  
 1295  
 1296  
 1297  
 1298  
 1299  
 1300  
 1301  
 1302  
 1303  
 1304  
 1305  
 1306  
 1307  
 1308  
 1309  
 1310  
 1311  
 1312  
 1313  
 1314  
 1315  
 1316  
 1317  
 1318  
 1319  
 1320  
 1321  
 1322  
 1323  
 1324  
 1325  
 1326  
 1327  
 1328  
 1329  
 1330  
 1331  
 1332  
 1333  
 1334  
 1335  
 1336  
 1337  
 1338  
 1339  
 1340  
 1341  
 1342  
 1343  
 1344  
 1345  
 1346  
 1347  
 1348  
 1349  
 1350  
 1351  
 1352  
 1353  
 1354  
 1355  
 1356  
 1357  
 1358  
 1359  
 1360  
 1361  
 1362  
 1363  
 1364  
 1365  
 1366  
 1367  
 1368  
 1369  
 1370  
 1371  
 1372  
 1373  
 1374  
 1375  
 1376  
 1377  
 1378  
 1379  
 1380  
 1381  
 1382  
 1383  
 1384  
 1385  
 1386  
 1387  
 1388  
 1389  
 1390  
 1391  
 1392  
 1393  
 1394  
 1395  
 1396  
 1397  
 1398  
 1399  
 1400  
 1401  
 1402  
 1403  
 1404  
 1405  
 1406  
 1407  
 1408  
 1409  
 1410  
 1411  
 1412  
 1413  
 1414  
 1415  
 1416  
 1417  
 1418  
 1419  
 1420  
 1421  
 1422  
 1423  
 1424  
 1425  
 1426  
 1427  
 1428  
 1429  
 1430  
 1431  
 1432  
 1433  
 1434  
 1435  
 1436  
 1437  
 1438  
 1439  
 1440  
 1441  
 1442  
 1443  
 1444  
 1445  
 1446  
 1447  
 1448  
 1449  
 1450  
 1451  
 1452  
 1453  
 1454  
 1455  
 1456  
 1457  
 1458  
 1459  
 1460  
 1461  
 1462  
 1463  
 1464  
 1465  
 1466  
 1467  
 1468  
 1469  
 1470  
 1471  
 1472  
 1473  
 1474  
 1475  
 1476  
 1477  
 1478  
 1479  
 1480  
 1481  
 1482  
 1483  
 1484  
 1485  
 1486  
 1487  
 1488  
 1489  
 1490  
 1491  
 1492  
 1493  
 1494  
 1495  
 1496  
 1497  
 1498  
 1499  
 1500  
 1501  
 1502  
 1503  
 1504  
 1505  
 1506  
 1507  
 1508  
 1509  
 1510  
 1511  
 1512  
 1513  
 1514  
 1515  
 1516  
 1517  
 1518  
 1519  
 1520  
 1521  
 1522  
 1523  
 1524  
 1525  
 1526  
 1527  
 1528  
 1529  
 1530  
 1531  
 1532  
 1533  
 1534  
 1535  
 1536  
 1537  
 1538  
 1539  
 1540  
 1541  
 1542  
 1543  
 1544  
 1545  
 1546  
 1547  
 1548  
 1549  
 1550  
 1551  
 1552  
 1553  
 1554  
 1555  
 1556  
 1557  
 1558  
 1559  
 1560  
 1561  
 1562  
 1563  
 1564  
 1565  
 1566  
 1567  
 1568  
 1569  
 1570  
 1571  
 1572  
 1573  
 1574  
 1575  
 1576  
 1577  
 1578  
 1579  
 1580  
 1581  
 1582  
 1583  
 1584  
 1585  
 1586  
 1587  
 1588  
 1589  
 1590  
 1591  
 1592  
 1593  
 1594  
 1595  
 1596  
 1597  
 1598  
 1599  
 1600  
 1601  
 1602  
 1603  
 1604  
 1605  
 1606  
 1607  
 1608  
 1609  
 1610  
 1611  
 1612  
 1613  
 1614  
 1615  
 1616  
 1617  
 1618  
 1619  
 1620  
 1621  
 1622  
 1623  
 1624  
 1625  
 1626  
 1627  
 1628  
 1629  
 1630  
 1631  
 1632  
 1633  
 1634  
 1635  
 1636  
 1637  
 1638  
 1639  
 1640  
 1641  
 1642  
 1643  
 1644  
 1645  
 1646  
 1647  
 1648  
 1649  
 1650  
 1651  
 1652  
 1653  
 1654  
 1655  
 1656  
 1657  
 1658  
 1659  
 1660  
 1661  
 1662  
 1663  
 1664  
 1665  
 1666  
 1667  
 1668  
 1669  
 1670  
 1671  
 1672  
 1673  
 1674  
 1675  
 1676  
 1677  
 1678  
 1679  
 1680  
 1681  
 1682  
 1683  
 1684  
 1685  
 1686  
 1687  
 1688  
 1689  
 1690  
 1691  
 1692  
 1693  
 1694  
 1695  
 1696  
 1697  
 1698  
 1699  
 1700  
 1701  
 1702  
 1703  
 1704  
 1705  
 1706  
 1707  
 1708  
 1709  
 1710  
 1711  
 1712  
 1713  
 1714  
 1715  
 1716  
 1717  
 1718  
 1719  
 1720  
 1721  
 1722  
 1723  
 1724  
 1725  
 1726  
 1727  
 1728  
 1729  
 1730  
 1731  
 1732  
 1733  
 1734  
 1735  
 1736  
 1737  
 1738  
 1739  
 1740  
 1741  
 1742  
 1743  
 1744  
 1745  
 1746  
 1747  
 1748  
 1749  
 1750  
 1751  
 1752  
 1753  
 1754  
 1755  
 1756  
 1757  
 1758  
 1759  
 1760  
 1761  
 1762  
 1763  
 1764  
 1765  
 1766  
 1767  
 1768  
 1769  
 1770  
 1771  
 1772  
 1773  
 1774  
 1775  
 1776  
 1777  
 1778  
 1779  
 1780  
 1781  
 1782  
 1783  
 1784  
 1785  
 1786  
 1787  
 1788  
 1789  
 1790  
 1791  
 1792  
 1793  
 1794  
 1795  
 1796  
 1797  
 1798  
 1799  
 1800  
 1801  
 1802  
 1803  
 1804  
 1805  
 1806  
 1807  
 1808  
 1809  
 1810  
 1811  
 1812  
 1813  
 1814  
 1815  
 1816  
 1817  
 1818  
 1819  
 1820  
 1821  
 1822  
 1823  
 1824  
 1825  
 1826  
 1827  
 1828  
 1829  
 1830  
 1831  
 1832  
 1833  
 1834  
 1835  
 1836  
 1837  
 1838  
 1839  
 1840  
 1841  
 1842  
 1843  
 1844  
 1845  
 1846  
 1847  
 1848  
 1849  
 1850  
 1851  
 1852  
 1853  
 1854  
 1855  
 1856  
 1857  
 1858  
 1859  
 1860  
 1861  
 1862  
 1863  
 1864  
 1865  
 1866  
 1867  
 1868  
 1869  
 1870  
 1871  
 1872  
 1873  
 1874  
 1875  
 1876  
 1877  
 1878  
 1879  
 1880  
 1881  
 1882  
 1883  
 1884  
 1885  
 1886  
 1887
```

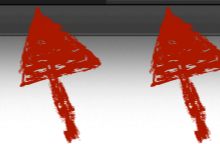
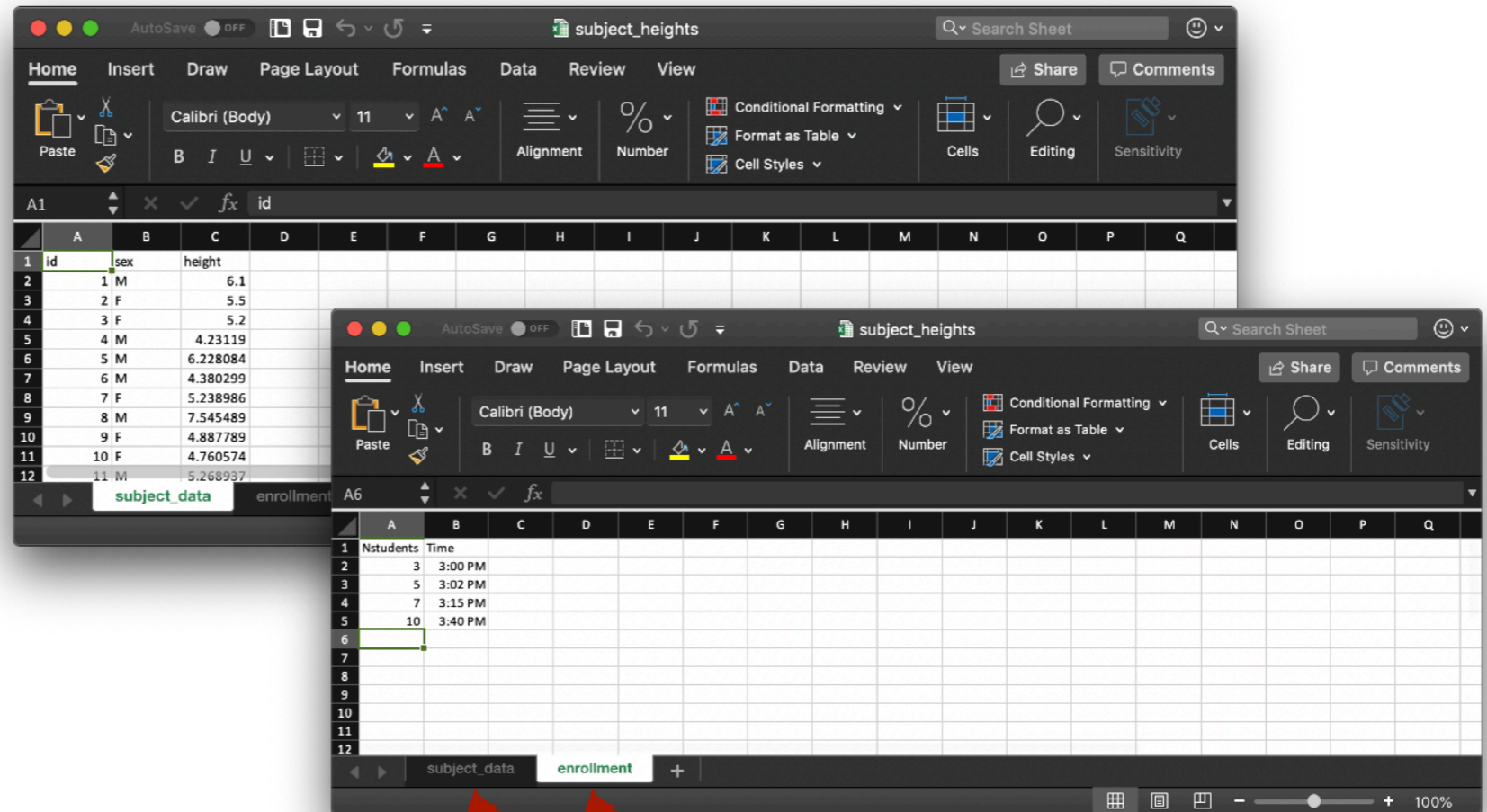
Excel Terminology

... a workbook may contain one or more worksheets ...



Workbook

subject_heights.xlsx



Worksheets

Reading an Excel Spreadsheet

... importing an XLSX file ...



```
# Parse all worksheet  
df = pandas.read_excel("subject_heights.xlsx", None);
```

```
# See worksheets available as dictionaries  
df.keys()  
# [u'subject_data', u'enrollment']
```

```
# Parse-per worksheet  
xl_overview = pd.ExcelFile('subject_heights.xlsx')
```

```
# See all worksheets  
xl_overview.sheet_names
```

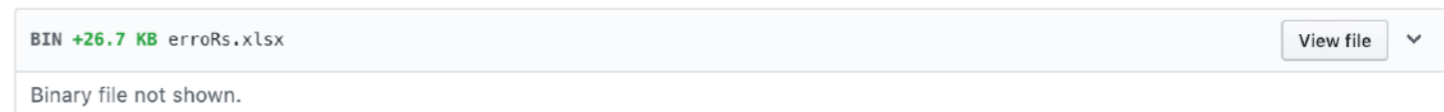
```
# Obtain a specific worksheet  
xl_overview.parse(sheet_name)
```

Pros vs. Cons

... when to employ each file type ...

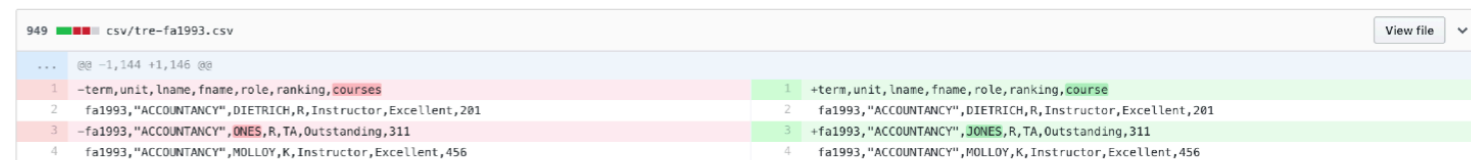
Does storage size matter?

> **Binary Files**



Do you need to view diffs?

> **Flat Files**



Missing Data

What happens if
a data point is missing?

Definition:

Missingness indicates that no data has been recorded or was omitted. In *Python*, we denote this by ***np.nan***, which stands for "***N***ot ***a***vailable ***N***umber."

id	sex	height
1	M	6.1
2	F	5.5
3	F	5.2
...
55	M	5.9

Complete Cases

id	sex	height
1	M	6.1
2	F	<i>NaN</i>
3	<i>NaN</i>	5.2
...
55	<i>NaN</i>	<i>NaN</i>

Incomplete Cases



Missingness *is*

Contagious & Propagates

Operations with missingness (NaN) yield more missingness (NaN)...

```
np.nan + 2
```

```
# [1] nan
```

```
np.nan == 2
```

```
# [1] nan
```

```
12 - 2 + np.nan*5
```

```
# [1] nan
```

```
np.nan == np.nan *
```

```
# [1] False
```

“An ***NaN*** is the presence of an absence. Don't forget that some missing values are the absence of a presence.”

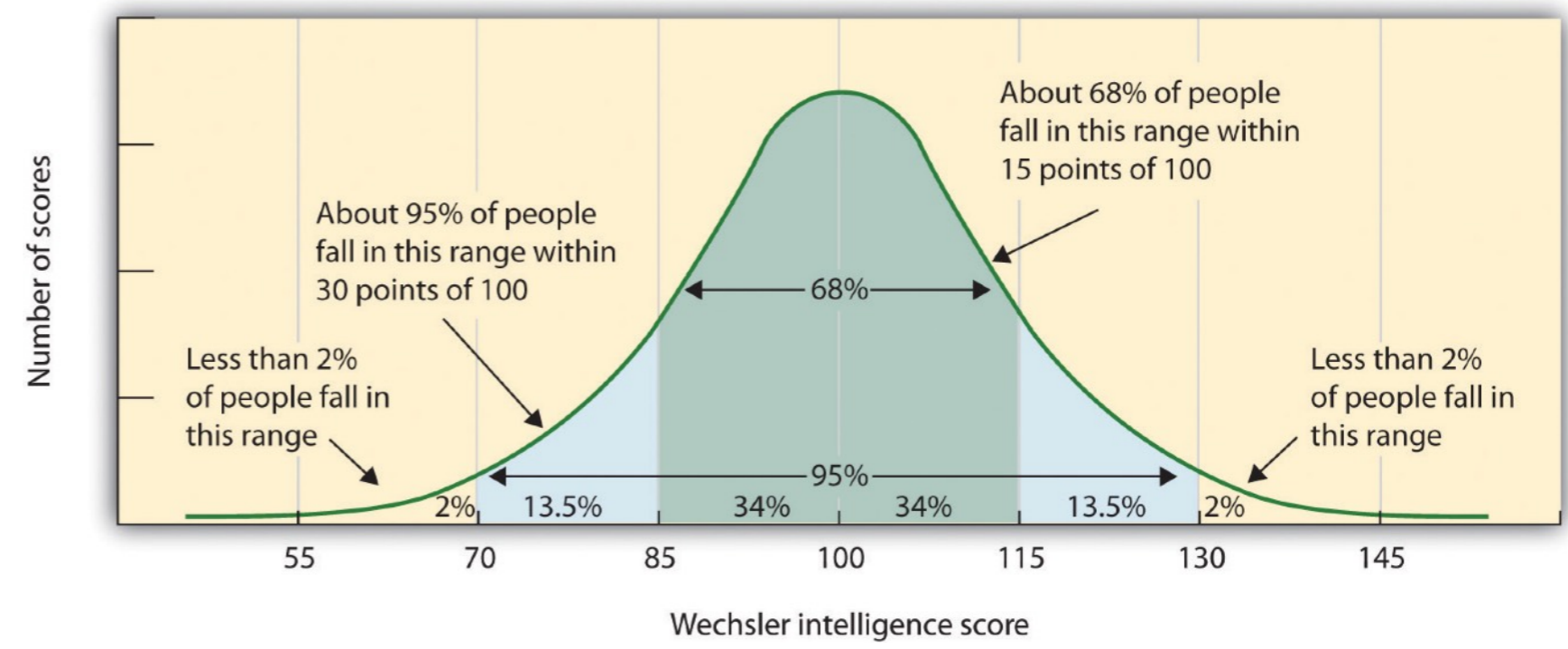
– Hadley Wickham on [Twitter](#)

Aside

IQ Example

... dealing with test data ...

IQ Range	IQ Classification
130 and above	Very superior
120–129	Superior
110–119	High average
90–109	Average
80–89	Low average
70–79	Borderline
69 and below	Extremely low



Source

Types of Missingness

... got data ???

Original

Age	IQ
18	112
19	108
19	94
22	87
25	132
28	79
30	103

Missing Completely At Random

Age	IQ
18	NA
19	108
19	94
22	87
25	NA
28	79
30	NA

????

Missing At Random

Age	IQ
18	NA
19	NA
19	NA
22	87
25	132
28	79
30	103

Non-response

Missing Not At Random

Age	IQ
18	112
19	108
19	NA
22	NA
25	132
28	NA
30	103

Low IQ

* **MCAR** indicates that no relationship exists between missing values and any observed values

** **MAR** indicates a relationship exists between missing values and recorded values that can be inferred.

*** **MNAR** indicates a relationship exists between the value of the missing data.

Missing Values

Note: The choice of using `NaN` internally to denote missing data was largely for simplicity and performance reasons. It differs from the `MaskedArray` approach of, for example, `scikits.timeseries`. We are hopeful that NumPy will soon be able to provide a native NA type solution (similar to R) performant enough to be used in pandas.

[Source](#)

```
# Check for missing values
uci_adult_df.isnull().values.any()
```

```
import numpy as np
```

```
# Create a copy
uci_adult_df_na = uci_adult_df.copy()
```

```
# Infect with a missing value
uci_adult_df_na.loc[1, ('Age')] = np.NaN
```

```
# Check for missing values in new DataFrame
uci_adult_df_na.isnull().values.any()
```

```
# Compute the median
age_median = uci_adult_df_na['Age'].median()
```

```
# Replace NA values with the median value
# for Age.
```

```
uci_adult_df_na['Age'] =
    uci_adult_df_na['Age'].fillna(age_median)
```

```
# Verify it's gone again
uci_adult_df_na.isnull().values.any()
```

Acknowledgements

Summary

- Data can be stored in a compressed or human readable format.
- Data storage formats may vary unexpectedly.
- Missing values are problematic and imputation should be used for them.

Acknowledgements

- [Pandas documentation](#) and the [pandas cheatsheet](#) by Irv Lustig
- Style of the RStudio Cheatsheet for Data Transformations

This work is licensed under the
Creative Commons
Attribution-NonCommercial-
ShareAlike 4.0 International
License

