Mid-term review

1. Assert
2. Numpy Indexing
3. List Comprehension
4. Pandas
5. Series and DataFrames
6. Indexing
7. Boolean Indexing
8. Quantiles
9. Correlation
10. Visualization
11. Confidence Intervals
12. Permutation Test
13. P-Value
14. Normal and Standard Normal Distribution
15. A/B test
16. Multi-Armed Bandit

Assert

* We tend to use assert method to test anything we know should be true
  + It will raise Assertion Error when statement is false and return error message

A screenshot of a computer code

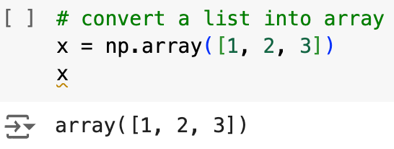
Description automatically generated

* + It returns nothing when statement is true.

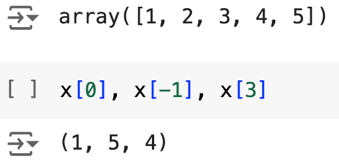


Numpy Indexing

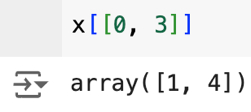
* Numpy aims to create an array of elements from a python list or tuples



* Array Indexing
  + Use .arrange(左闭右开) to create an array.
  + Just like list indexing, it starts from 0 to the array.
  + For **multiple individual searches**, it uses array[index] for each individual element search. It follows index rule



* + - Numpy Fancy indexing:
      * Array [[index 1, index 2, ….]] to extract a sub list of arrays based on given index. It doesn’t follow index rule.
      * Give elements whose index are 0 and 3



* + For **multiple continuous searches**, it uses array [start index: end index + 1]

A close up of a number

Description automatically generated

* + - Read [x: y] as from x to y + 1 (x and y is index)
    - Get the first 4 element 🡺 first refers to y + 1

A screenshot of a computer

Description automatically generated or A close-up of a number

Description automatically generated

* + - Get last 3 elements 🡺 last refers to x

A close-up of a computer code

Description automatically generated

* + - Get middle range to give element from 1 to 4

A screenshot of a computer

Description automatically generated

* + Negative indexing

A close-up of a piece of paper

Description automatically generated

* + - Get last 3 items

A close up of a number

Description automatically generated

* + - Get first 2 items

A close-up of a number

Description automatically generated

* + - Get range of items from element 1 to 3

A screenshot of a computer

Description automatically generated

* + Array indexing with step size
    - Array [x: y + 1: # of step to skip]
    - Get element from 1 to 4 with even numbers

A close-up of a number

Description automatically generated

* + Reverse array with -1 \* # of step size
    - Array [x: y + 1: - (# of step size)]
    - Get element in the items but each even number with reverse order

A close-up of a number

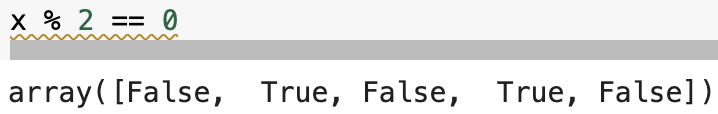
Description automatically generated

Boolean Indexing

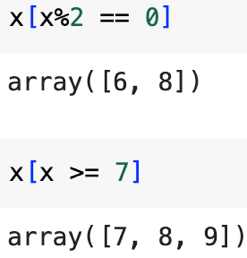
* It tends to select everything in array and apply to *the statement’s conditions*, returning the Boolean values to indicate who meets the condition and who doesn’t.



* Boolean mask: returned Boolean array with true or false based on the condition.
  + Give Boolean result of even numbers in the array
  + x % 2 == 0 is a conditional statement



* Boolean DataFrame: returned a value that only meets the conditional statement.
  + Array [ conditional statement regarding array] will give the value array that only meets the requirement



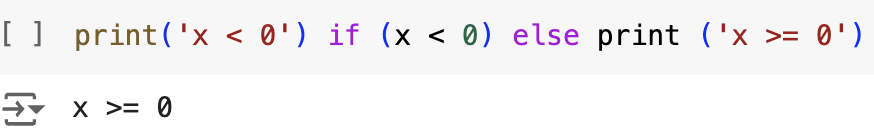
List Comprehension

* A process to transform the statement involving control flow statements.
* Control flow: if-else flow
  + if condition 1: statement 1
  + elif condition 2: statement 2
  + else: statement 3

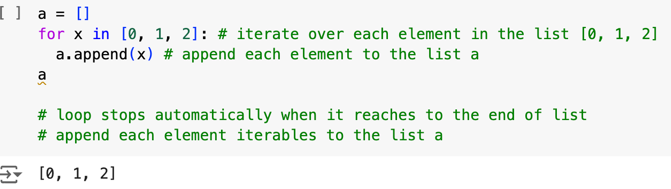
A screenshot of a computer

Description automatically generated

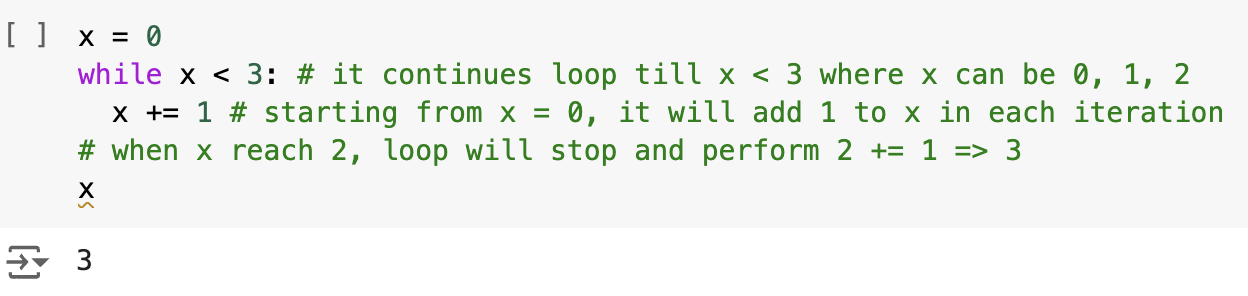
* + Convert into one line
    - Statement 1 if condition 1 else statement 2



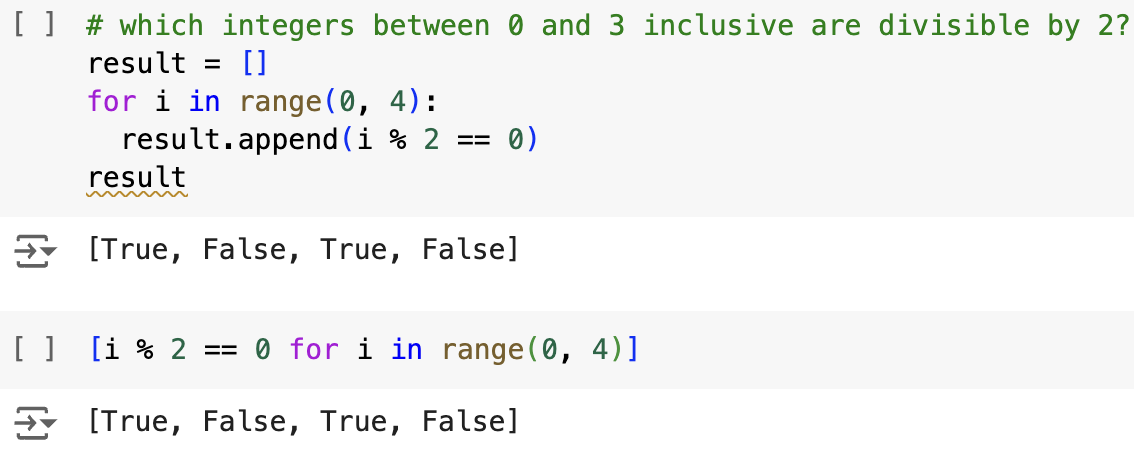
* Control flow: for loop
  + Use for loop to repeat instructions for a fixed number of times
  + Use for loop to iterate fixed sequence of values and reach to the end of list.



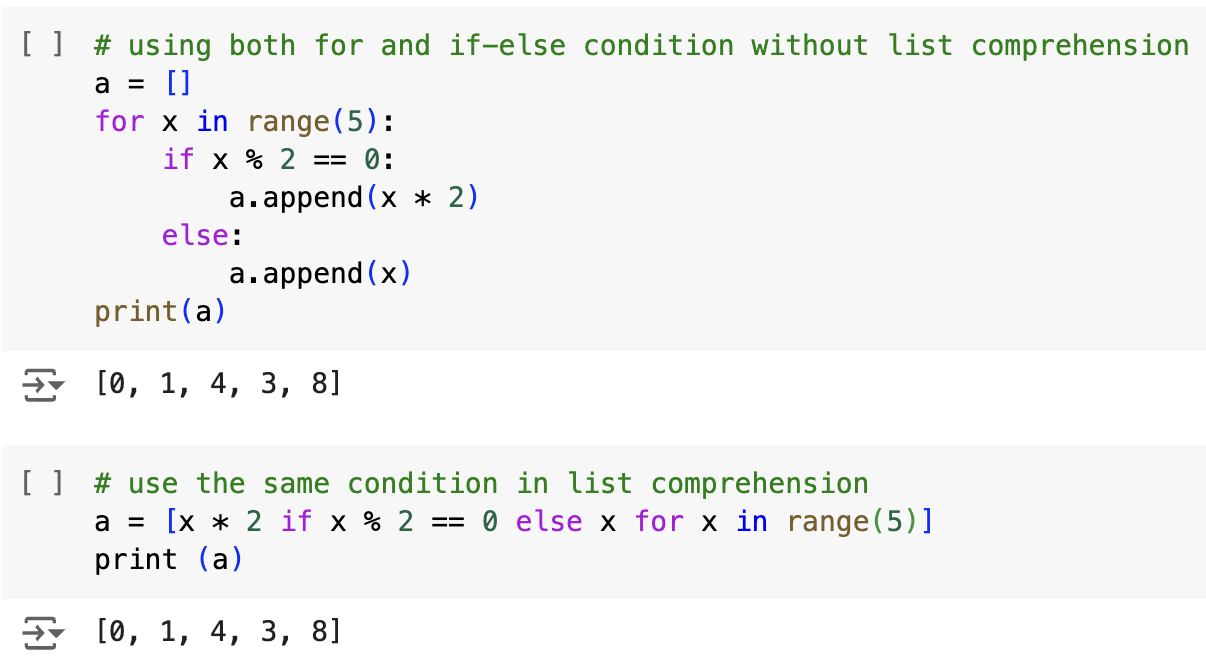
* Control flow: while loop
  + Use while loop to repeat instructions indefinitely until specified condition is true.
  + Use while loop when # of iterations are unknown, or loop should continue when a certain condition is met.



* Structure of list comprehension
* [instruction for item in iterables]



* [expression if condition else statement for item in iterables]



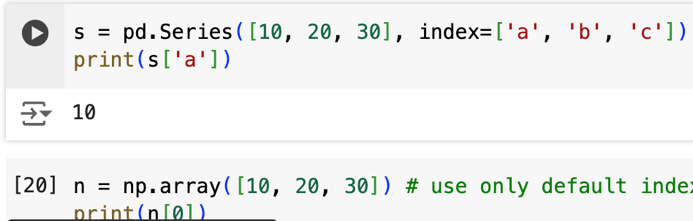
* If statement has no else statement, then we need to swap the position with for-loop

A screenshot of a computer

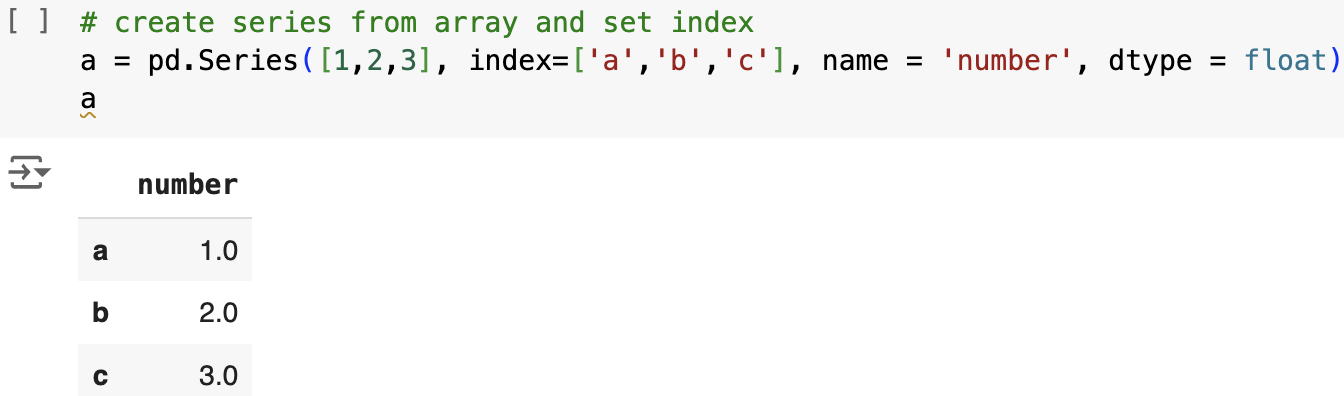
Description automatically generated

Pandas

* Pandas consists of 1-D series with customizable index or default as 0, 1, …
* Pandas consists of 2-D dataframe with index and column name.
  + Series and Array difference



* Pandas’ series
  + It can only translate array or list of elements into 1-D.
  + It can store any data types
  + Series index can be flexibly defined into anything hashable.
  + Create a series from list
    - pd.Series ([list], index = [‘a’, ‘b’, …], name = ‘name\_series’, dtype = any data type)
    - index is customizable and each list value will be granted with an index.

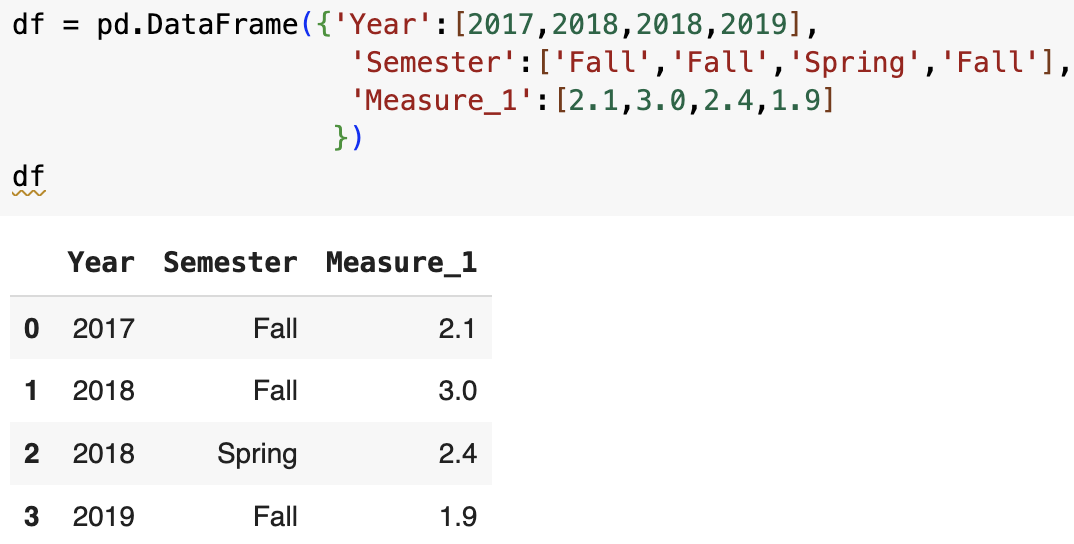


* + Create a series from dictionary
    - pd.Series ({‘key 1’: value 1, …}, name = ‘name\_series’, dtype = any data type)
    - index is the key name corresponding to each value.

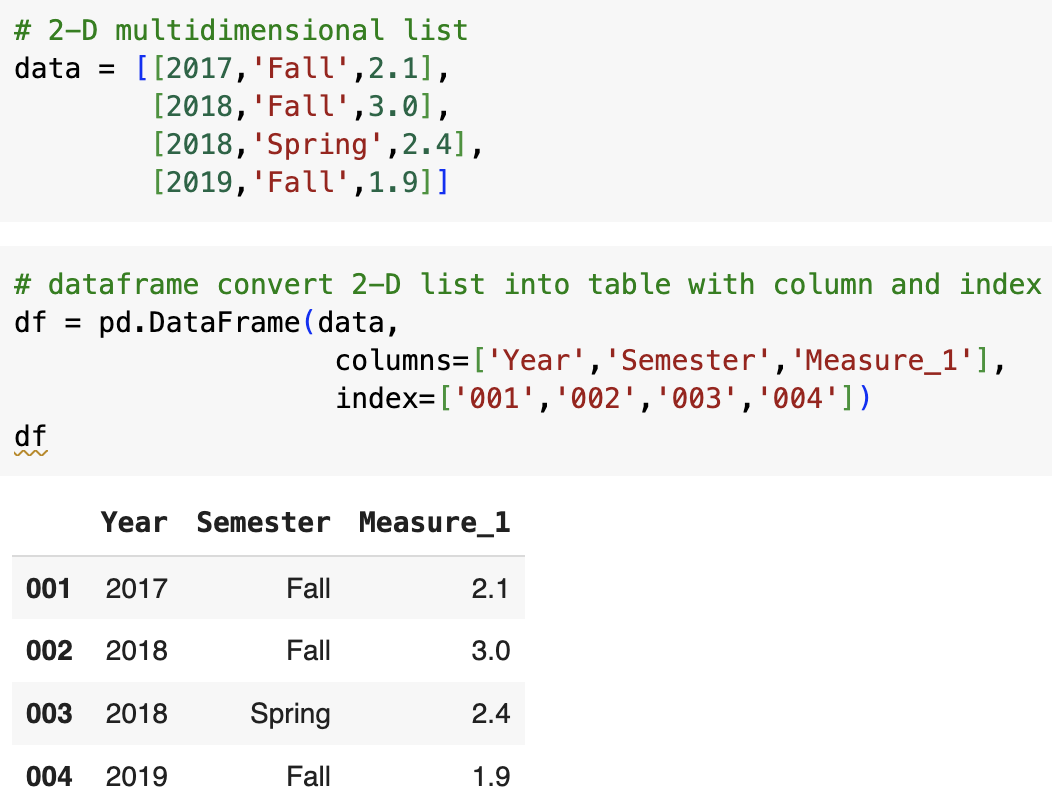
A screenshot of a computer

Description automatically generated

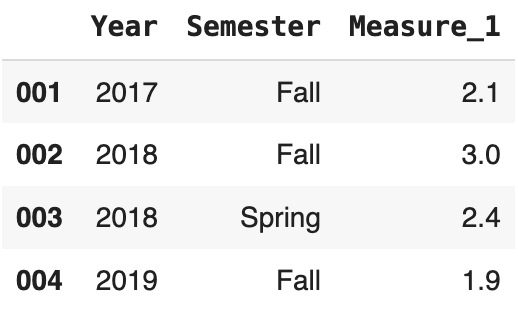
* Panda’s DataFrame
  + It can convert 2-D dictionary into tabular data.
  + Each column should have single and unique data type.
  + Contains both row and column index.
  + It can automatically generate row index from 0, 1, 2, …
  + Each column name is granted with 0, 1, 2, …
  + Create a dataframe from dictionary
    - pd.DataFrame( {‘key 1’: [value a, value b, …], …})



* + - each key will be the column with default granted index
    - each list of values corresponding to the key will be the row values with index.
  + Create a dataframe from 2-D list
    - pd.DataFrame ([2 -D list], columns = [customize index for columns], index = [customize index for rows])

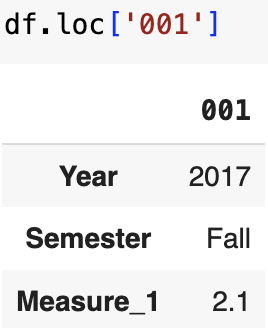


Pandas Indexing



1. .loc[ ]: query row and col by row and col names (label-based).

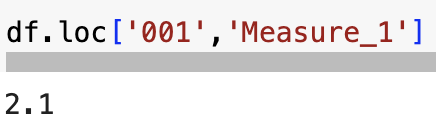
* Single row or col query by .loc[ ]
  1. .loc [‘x’] 🡺 single row query



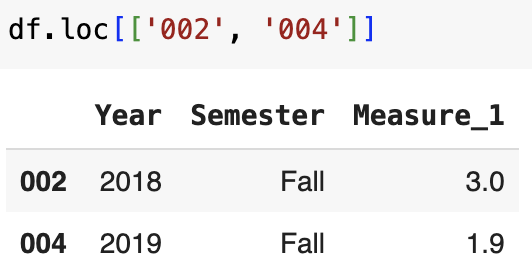
* 1. .loc [ : , ‘a’] 🡺 single col query



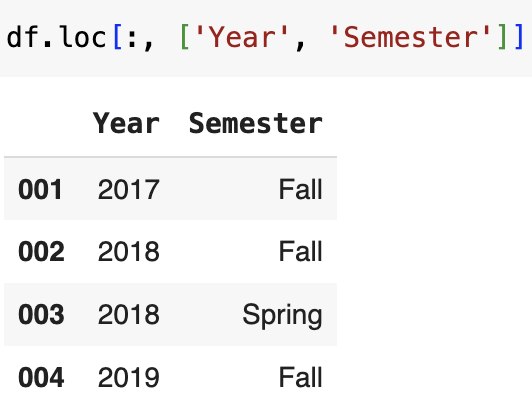
* 1. .loc [‘x’, ‘a’] 🡺 specific value in data frame.



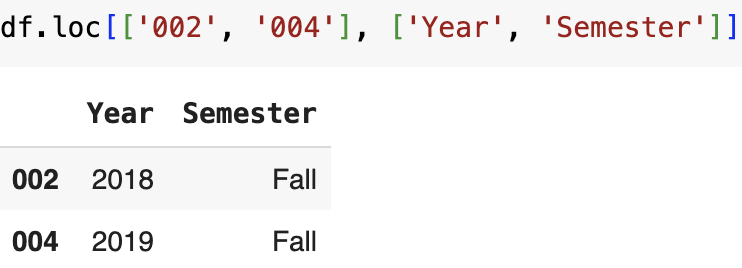
* Multiple individual row or col query by .loc[ ]
  1. .loc [ [ ‘x1’, ‘x2’, …] ] 🡺 multiple individual row query



* 1. .loc [ : , [‘a1’, ‘a2’, …] ] 🡺 multiple individual column query

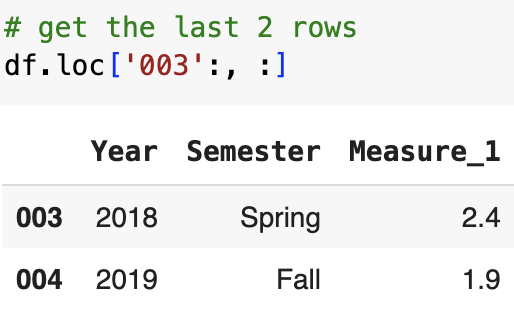


* 1. .loc [ [ ‘x1’, ‘x2’, …], [‘a1’, ‘a2’, …] ] 🡺 specific value in dataframe.

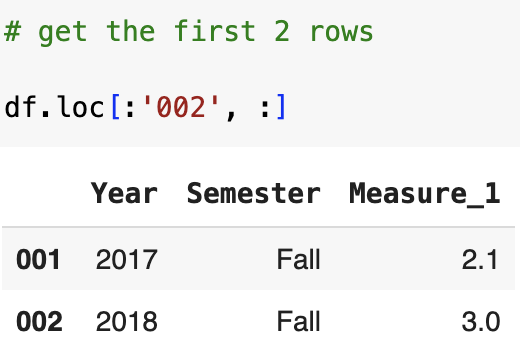


* Multiple consecutive row or col query by .loc [ ]

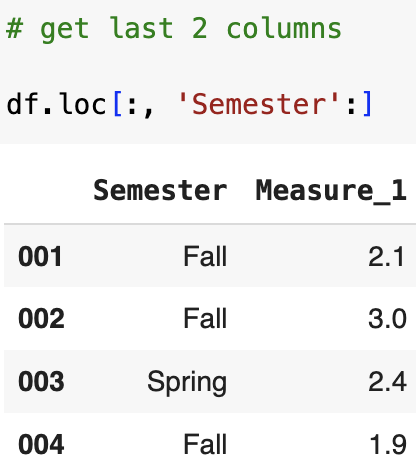
1. .loc [ ‘x’: ] 🡺 get rows from row x to the end



1. .loc [ : ‘y’ ] 🡺 gets rows from 1st row to row y



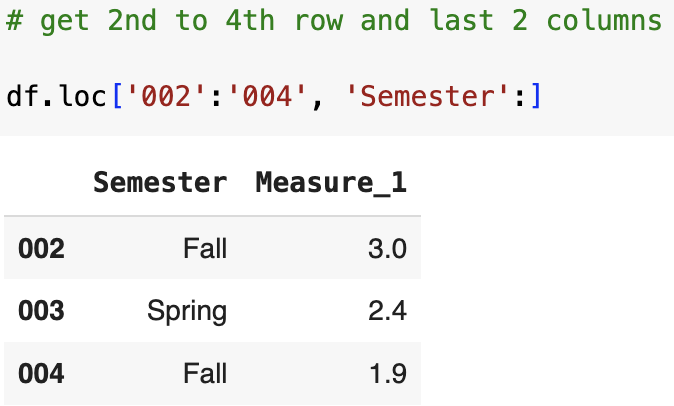
1. .loc [ : , ‘a’ : ] 🡺 gets columns from col a to the end



1. .loc [ : , : ‘b’] 🡺 gets columns from 1st col to col b

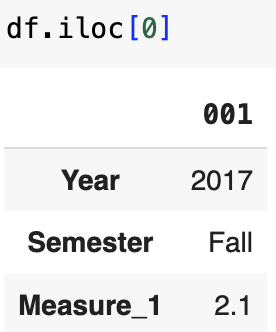


1. .loc [ ‘x’ : ‘y’ , ‘a’ : ‘b’ ] 🡺 get the rows from x to y and col from a to b

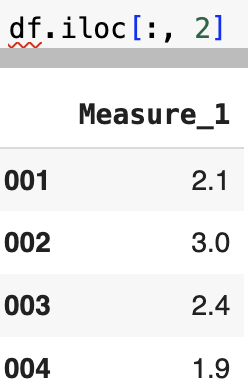


1. .iloc [ ]: query single or range of values based on row and column index (index-based with list indexing rule)

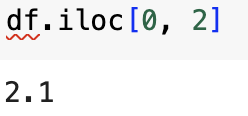
* single row and col query
  1. .iloc [x] 🡺 get particular row with index x



* 1. .iloc [ : , a ] 🡺 get the particular col with index a

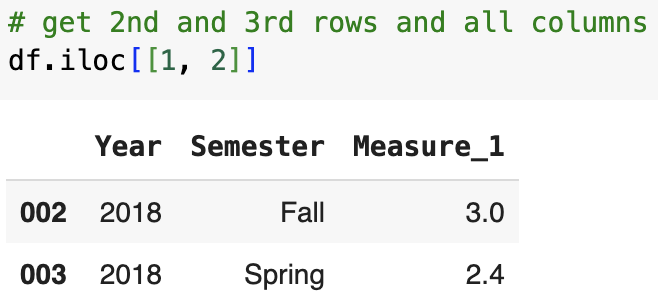


* 1. .iloc [ x , a ] 🡺 get the particular value in dataframe

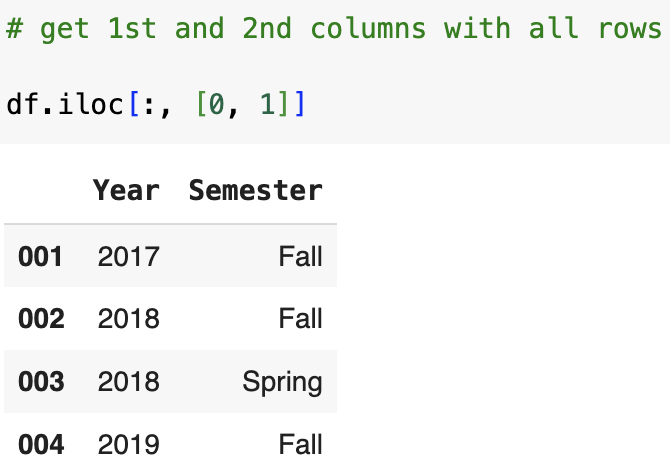


* Multiple individual row and column indexing

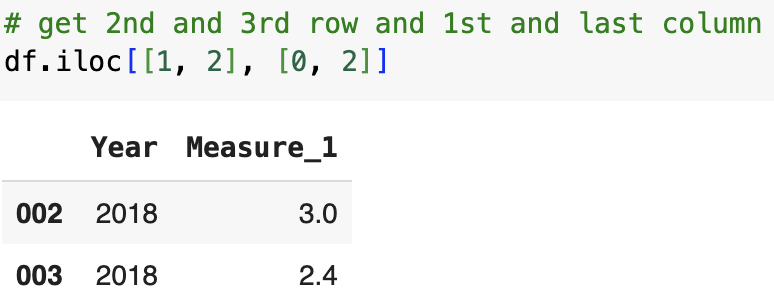
1. .iloc [ [x , y] ] 🡺 get row x + 1 and row y + 1.



1. .iloc [ : , [ a , b ] ] 🡺 get col a + 1 and col b + 1.

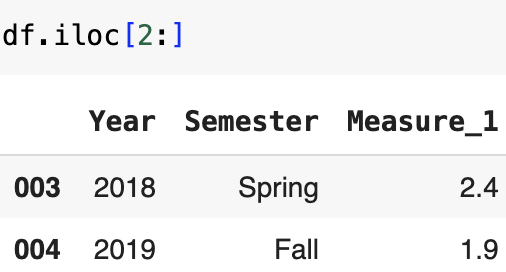


1. .iloc [ [ x , y ] , [ a , b ] ] 🡺 get row x + 1, y + 1, and col a + 1 and col b + 1.

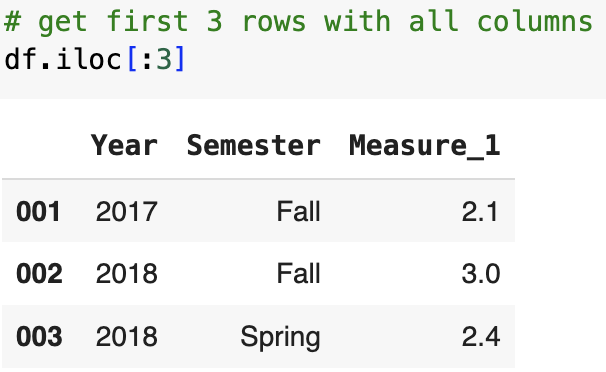


* Multiple consecutive row and cols indexing (左闭右开)

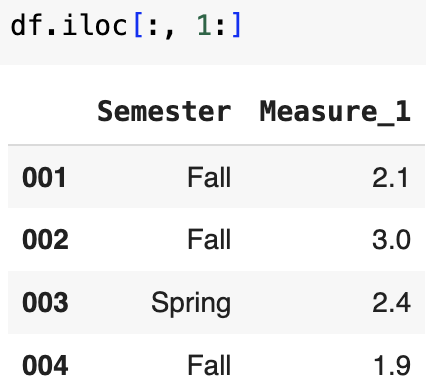
1. .iloc [ x : ] 🡺 get rows from rows with index x to the end



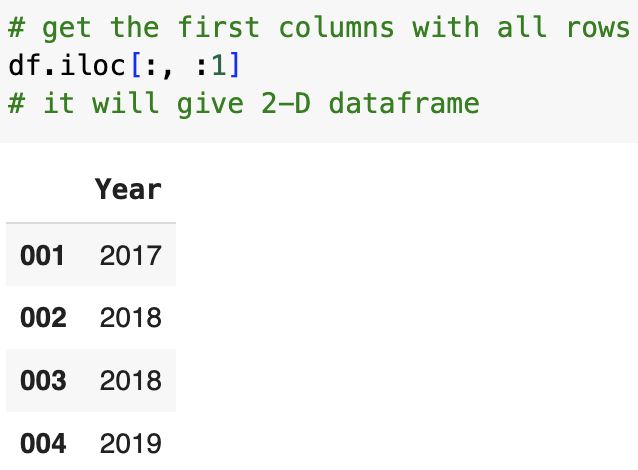
1. .iloc [ : y ] 🡺 get rows from 1st to row with index y + 1



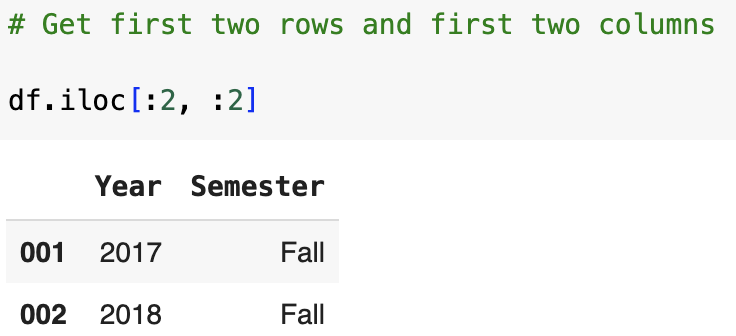
1. .iloc [ : , a : ] 🡺 get column from column with index a to the end



1. .iloc [ : , : b ] 🡺 get columns from 1st col to the col with index b + 1

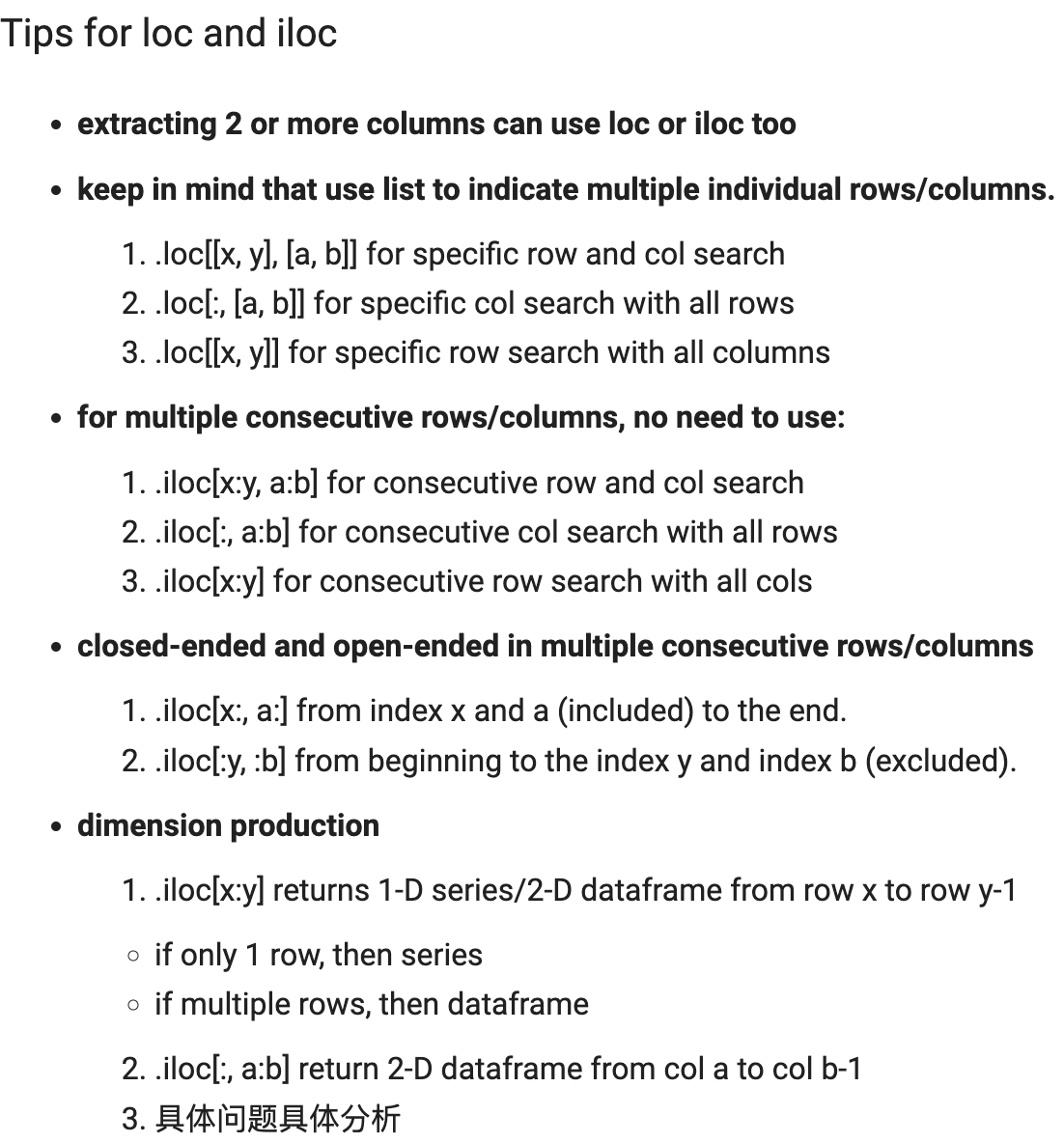


1. .iloc [ : y , : b ] 🡺 gets first y + 1 index rows and first b + 1 index columns

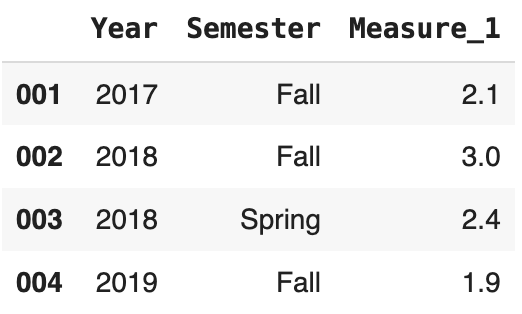


1. .iloc [ x : y , a : b] 🡺 get the rows from index x to index y (exclusive) and col from index a to index b (exclusive).

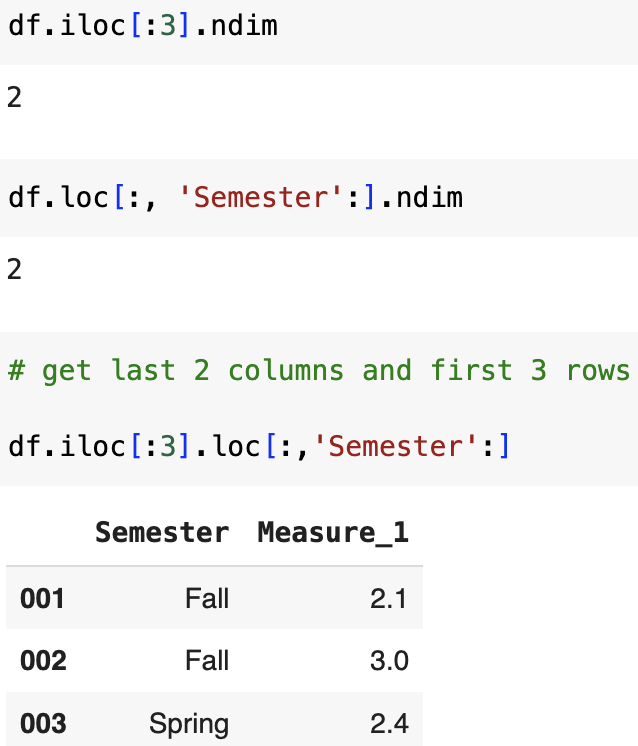
* Summary



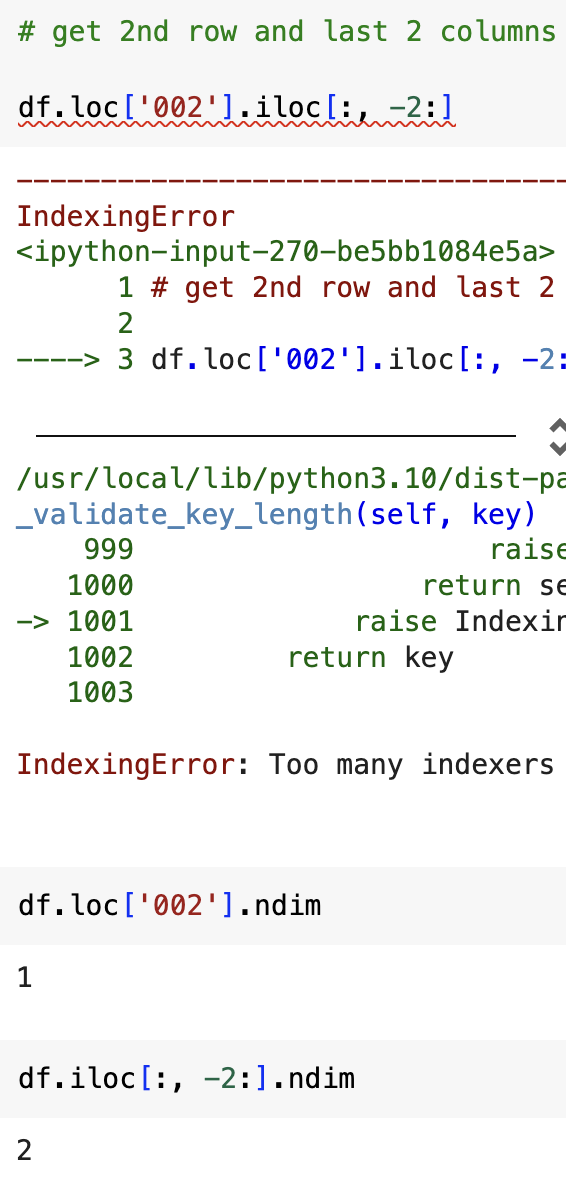
Pandas Selection Chaining



* loc 和 iloc 前只看前边的data的维度是否可以兼容后边的维度，如果前边是2维的dataframe，则代表它后可以跟1维或2维的；如果前边是1维的，则后边只能跟1维的。
* 2-D followed by 2-D



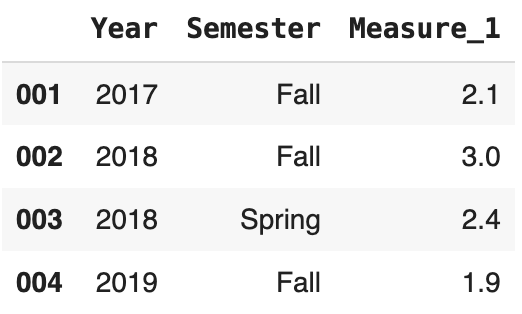
* 1-D followed by 2-D



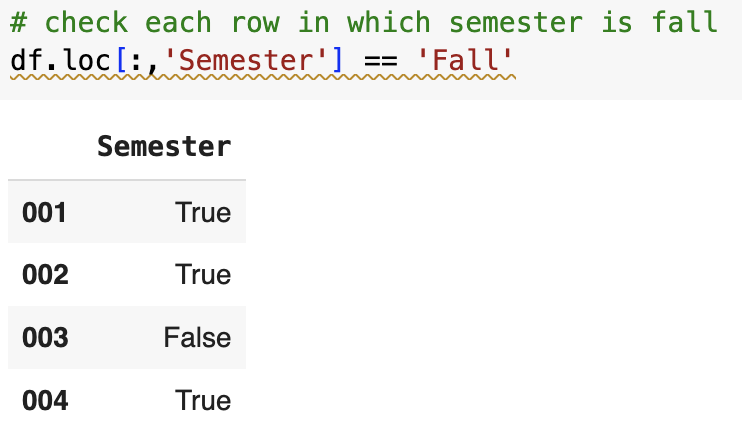
* 2维和1维的切换
  + 当.iloc [ : , 0 ] 时，选择多行单列，则返回是1维series
  + 当.iloc [ : , [0] ] 时，选择仍然是多行，但返回是2维dataframe，因为这里的[ 0 ] 代表潜在的multiple individual col索引
  + 当.iloc [ x : y , a : b] 时，要分情况看数据的样式，但大部分是都是返回的2维dataframe

Pandas Boolean Indexing

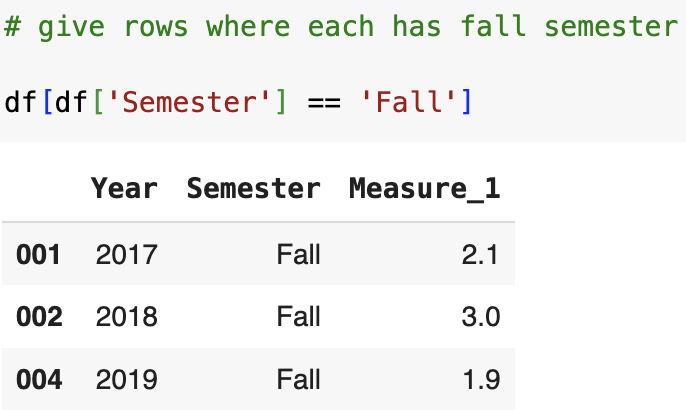
* .loc [ ] == ‘value’ to return Boolean mask of rows where value in specified columns matched specified value.



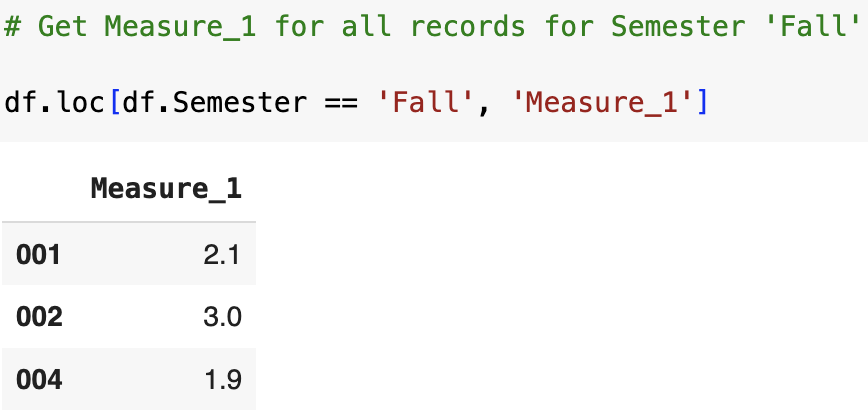
* The value is within the class under the specified columns, and we return rows with true satisfied with certain value or false dissatisfied with the value.



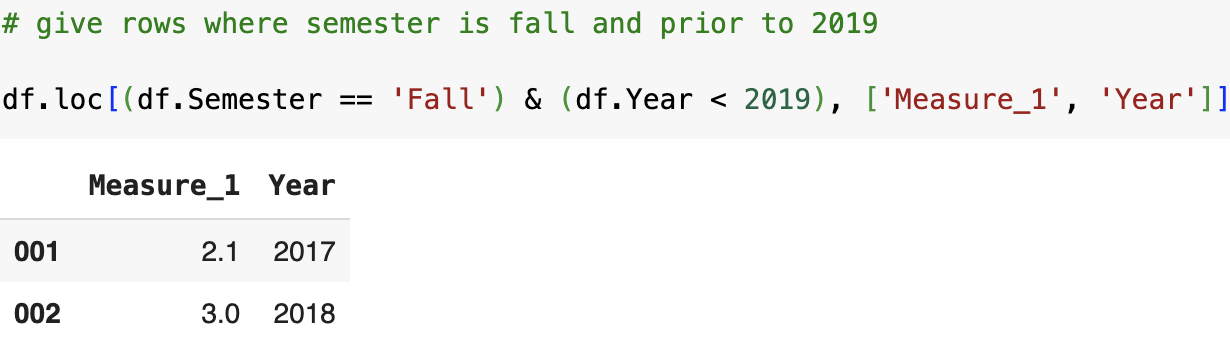
* Dataframe [ dataframe[ ‘col’ ] == ‘value’ ] 🡺 return rows where col with specified value



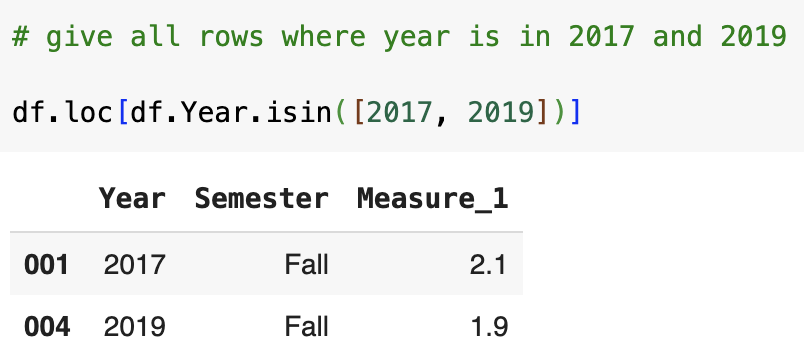
* Dataframe.loc [ dataframe.col\_1== ‘value’, ‘col\_2’ ] 🡺 returns col\_2 with rows where col\_1 have specified value.



* Need to use bitwise operator for value column filtering (&, |, ~)

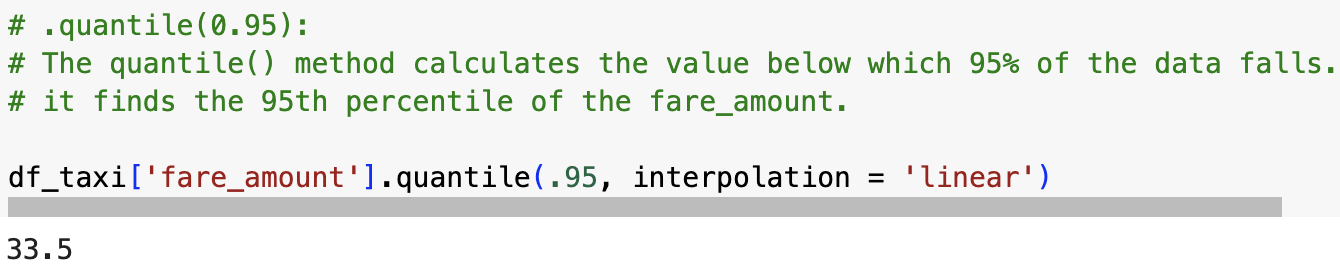


* 如果只想选在某列多个单独的value，则用col.isin ( [ value\_1, value\_2, …] ) 🡺 row filtering

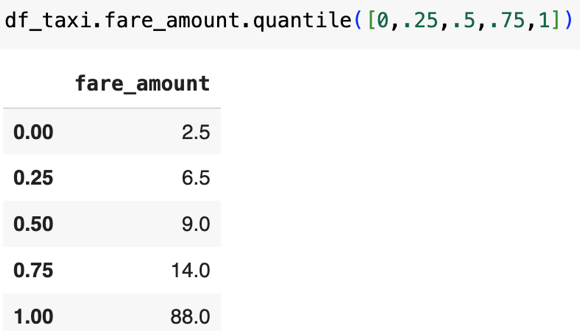


Quantiles and Percentiles

* Quantiles: it divides data into 4 equal parts where 25% of data falls each part.
  + It can describe how many data falls below the specified quantile (95%), corresponding to the returned value.



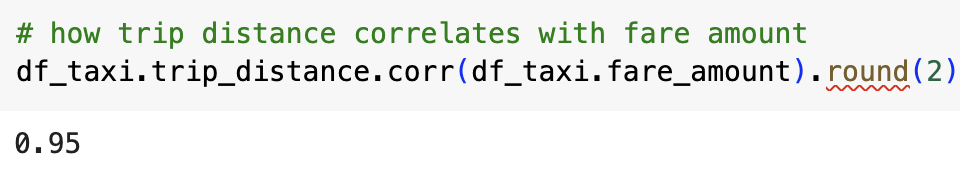
* + The returned value (33.5) indicates the threshold below which 95% of fare amounts falls.
  + It indicates 95% of fare are below 33.5 or less.
* For 4 parts



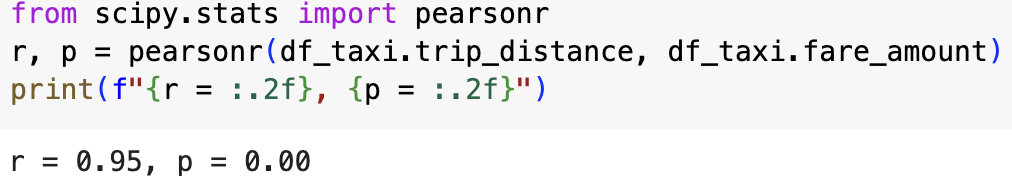
* + It describes how each 25% data falls under different returned value range where 25% of data fall between 2.5 to 6.5 of fare amount.

Correlation

* It describes degree where 2 variables are linearly related.
  + Sign (+ or - ) indicate the direction of correlation
  + Absolute value indicates the magnitude of correlation.
  + Col\_1. corr (Col\_2)



* Pearson correlation
  + Pearson correlation (r ) measures strength and direction of linear relationship.
  + r, p = pearsonr (Col\_1, Col\_2) will get coefficient ( r ) and p-value



Confidence Interval

* the range of values that true population parameter (mean) is likely to fall within the given certain level of confidence.
* Interpretation of CI
  + The range within which we can be 95% confident that true population mean falls between lowest point to highest point, based on sampled data.
  + If we were to construct confidence interval from infinite number of bootstrap samples, about 95% of those intervals would contain true parameter.
* Purpose of confidence interval
  + Use CI to estimate where true population parameter falls based on the given sample to infer to the population.
  + It quantifies uncertainty around sample estimate to infer population.
    - Narrower CI has more precision.
* Ways to build confidence interval
  + Generate bootstrap samples from original data as samples with replacement.
  + Get the sample mean.
  + Apply bootstrapping to resample data with replacement where each bootstrap’s mean is obtained.
  + The distribution of means from each bootstrap is used to build 95% CI.
* Find trim points to remove endpoint from CI
  + To create x% CI, we can determine the range of true population parameter falls within the CI in bootstrap distribution

A close up of text

Description automatically generated

Hypothesis Testing

* We want to determine whether there is enough evidence to reject null in favor of alternative hypothesis.
  + H0: the difference is happening due to luck or random chance.
  + H1: the different is happening not due to luck but to real difference between 2 groups.
  + Significance level: threshold of making Type I error where we incorrectly reject the H0 when H0 is actually true.
  + P-value: probability of obtaining test result as least extreme as observed result, under assumption H0 is true.
    - Test result: parameter from sample to compare observed result.
    - Observed result: actual measurement from sample.
* Type I and Type II error
  + Type I error: false positive
    - When H0 is true, but we incorrectly reject H0, where we mistakenly conclude there is effect or difference when, in reality, there isn’t.
    - Significance level links with type I error where p value is set as threshold (0.05), indicating there is always 5% risk of making Type I error.
    - Detect male patient with pregnancy where he is not in reality.
  + Type II error: false negative.
    - When H1 is true, but we incorrectly reject H1, where we mistakenly conclude that there isn’t effect or difference when, in reality, there is.
    - Beta links with Type II error where increase significance level will lower beta or vice versa.
    - Detect pregnant female without pregnancy where she is actually pregnant.

Permutation Test

* It is used to measure the significance of an observed effect by comparing observed data to distribution of outcomes obtained by randomly rearranging data labels.
* Goal: to test if observed data arrangement is not due to random chance compared to what might occur by chance.
* Non-parametric approach without relying on normality assumption. Instead, it generates a normal distribution based on the sample data itself.

Conceptual Steps

1. Compare groups together (assume H0 is true)
   1. Group all data together by pretending there is NO difference between groups by holding H0 assumption where any observed effect is due to random chance.
2. Permute (reorder) observations
   1. Randomly shuffle combined data to generate a new data arrangement, simulating what data arrangement looks like if H0 is true.
3. Create new groups to keep same sizes of original groups.
   1. Split permuted data into groups to match original group size, preserving the new randomized data structure matching with original one.
4. Choose and calculate metric:
   1. Compute metric by specified parameters (mean, difference, median, or etc.) for the new groups 🡺 test statistics
   2. This step is crucial since we will use such metric to represent effect for randomly permuted data to get randomized sampling distribution.
5. Repeat such step many times:
   1. Perform permutation and metric calculation steps multiple times (> 1,000).
   2. Each repetition produces a new test statistic for reordered groups and metrics.
   3. The distribution of possible outcomes under H0 assumption is created.
6. Locate original observation falls in the sampling permutation distribution.
   1. Compare where observed result is more extreme to the distribution of test statistics by permutations under H0 assumption.
   2. Permutation distribution = randomized data distribution under H0 assumption.
7. Determine significance of observed result.
   1. If observed result falls in the tail of distribution, it indicates that such outcome is rare under H0 assumption

🡺 there is a true difference / effect of such observed result.

🡺 the observed effect is highly unlikely occurred by random chance.

🡺 reject H0.

1. Normalization distribution by Z score
   1. Applying Z score to permutation test to standardize scale on test statistics.
      1. To normalize observed result and permuted test statistics to interpret extremeness of observed statistics on a common scale.
   2. Set mean as 0 and standard deviation of 1.
   3. Higher Z score will imply observed test statistics more extreme to permuted distribution.

A math equation with a white background

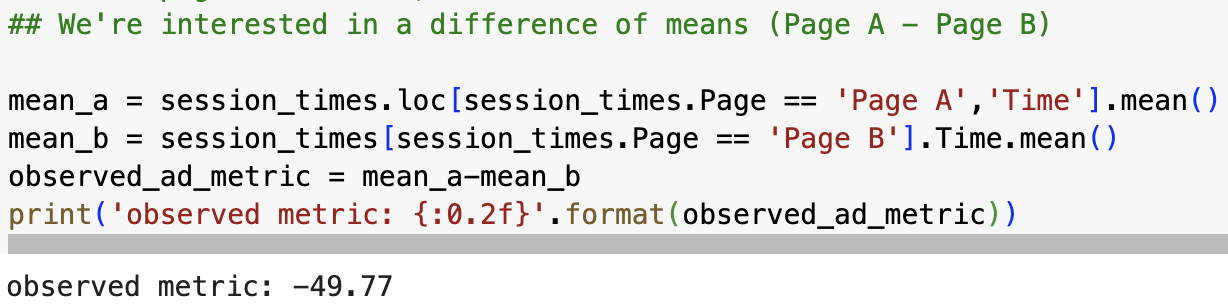
Description automatically generated with medium confidence

Technical steps

Q: we want to know which webpage leads to more sales by comparing time on page.

1. Observed result: we want to use mean difference between page A and page B as metric

* Get the observed metrics as observed result (mean difference between A and B)



1. Get group sizes one for page A and page B respectively

A computer screen shot of a math equation

Description automatically generated

1. Combine groups together under H0 assumption between page A and Page B

A screenshot of a computer

Description automatically generated

1. Permute observations

* Randomly shuffling original data (Time) without replacement by preserving the same sample size under H0 assumption

A screen shot of a computer

Description automatically generated

1. Create new groups to keep same size of original groups in page A and page B

A black and green text

Description automatically generated

1. Calculate metric from permuted samples.

* Get the mean difference between time in page A and page B

A close-up of a math equation

Description automatically generated

1. Repeat such steps many times on permuted observations and new mean difference.

A screen shot of a computer code

Description automatically generated

1. Compare observed mean difference falls to the permutation distribution

A graph with blue and green bars

Description automatically generated with medium confidence

1. Normalized distribution

* Normalize values on permuted distribution on metrics.

A close up of text

Description automatically generated

* Normalize values on observed result on mean difference.



* Locate the normalized observed result on the normalized permuted distribution

A graph with blue and green lines

Description automatically generated

Multi-Armed Bandit

It aims to choose between several options to maximize cumulative rewards.

* With unknown probability distribution of rewards in each option (arm), we need to maximize cumulative reward by choosing the best arm over time.
* Exploration: find which arms are more promising by trying different arms based on reward distribution.
* Exploitation: maximize reward by choosing arms with highest known average rewards.

Greedy Algorithm for MAB

1. Choose small Epsilon € to determine frequency of exploration
   1. Higher €, more exploration
   2. Lower €, more exploitation.
2. Generate random number between 0 and 1.
3. Decision based on €
   1. If random number < €, algo explores to gather potential cumulative rewards by different arms.
   2. If random number >= €, algo exploits to maximize immediate average reward based on current exploration.
4. Repeat the process: gradually improve the estimate of arm’s rewards.

Ground truth in MAB (unknown but true probability distribution of rewards of each arm)

* Exploration helps to gather info for reward distribution, help algo better estimate ground truth probabilities.
* Exploitation helps to select arm with highest known average reward based on current estimate.
  + If estimate is near ground truth, algo will consistently pick optimal arms
  + If estimate is away from ground truth, algo will try another optimal arms.
  + Repeatedly, algo will gather enough info to estimate reward distribution closer to ground truth.

Machine Learning Model tasks

1. Classification: predict categorical target
2. Regression: predict numeric target
3. Feature selection: detect feature importance when predicting hue
4. Interpretation: correlation / interaction between features and targets
5. Clustering: observation group feature space.

Supervised Learning framework

* Models is trained on labeled data
* Goals: to make prediction based on training data with performance evaluated by a known set of labels (outcomes).
  + Classification: where model predicts a categorical label.
  + Regression: where model predicts continuous value.

Unsupervised learning framework

* Model is trained on unlabeled data
* Goals: to discover hidden patterns, relationships, or groupings within data without prior knowledge of outcomes.
  + Clustering: groups data into clusters based on similarity
  + Dimensionality reduction: reduce number of features while preserving as much info as possible.

Prediction and Interpretation

1. Prediction: model ability to make accurate prediction based on training data.
   1. Use patterns to predict outcomes.
   2. Goal: to minimize error / maximize accuracy to predict on new and unseen data.
2. Interpretation: model’s decision-making process to explain relationships between features and predictions.
   1. Goals: to use model explain impact of each feature on the outcome.

Linear Regression Model

1. Correlation

* It reflects as coefficient in linear regression between each IVs and DV.
* High correlation between variables can improve predictive power but no causation conclusion can be made.
  + Correlation measure strength and direction of linear relationship between 2 variables.
  + Correlation doesn’t imply causation.

1. Causation

* Change in one variable will directly results in change in another.
* Regression model cannot imply causation but mostly correlation.
  + Causality measures one variable directly affect the other.
  + Establishing causality requires controlled experiments or careful consideration of confounding factors.

Simple Linear Regression Model

1. A black symbols with a plus and a black letter

   Description automatically generated with medium confidence
   1. W1: coefficient or slope.
   2. W0: coefficient, bias term, or intercept.
   3. €i: error (distance between actual and predicted data)
   4. Goal: to find a line that can best fit observations.
2. Ordinary Least Squares (OLS)
   1. Use OLS to fit the line by minimizing SSE between observed and predicted data from regression line.
   2. Since minimizing SSE required quadratic closed form, we can calculate W0 and W1 directly without iterative methods.

A math equations on a white background

Description automatically generated

1. Gradient Descent
   1. Iterative method to minimize SSE in regression by adjusting W0 and W1 in different attempts to reduce error.
   2. A gradient is a vector of partial derivatives, indicating direction of steepest increase or decrease.
   3. Goal: to minimize error by taking steps in opposite direction of the gradient.
      1. Maximization: it follows gradient’s direction to increase objective function.
      2. Minimization: it follows opposite direction of gradient to decrease objective function (minimizing squared error).
   4. Maximums and Minimum
      1. Global maximum /minimum: the absolute best solution over entire dataset.
      2. Local maximum / minimum: best solution within a specific neighborhood but not necessarily globally best.
2. Reshape (-1, 1)
   1. We need to convert input feature matrix into 2-D Dataframe.
   2. X needs to be shaped as column vector (only 1 column) with each row representing an observation and each column representing a feature.

A close-up of numbers

Description automatically generated 🡺 1-D array

In example

A close-up of a website

Description automatically generated

* 1. Use reshape ( -1, 1 )
     1. – 1 tells NumPy to infer from # of rows based on length of array.
     2. 1 enables there should be exactly 1 column, making it a column vector.

A close-up of numbers

Description automatically generated 🡺 2-D dataframe.

* 1. Many other ways to convert array into 2-D dataframe from example

A screenshot of a computer code

Description automatically generated

1. Coefficient interpretation

* On average, the amount of change in DV for a one-unit change in IV, holding all other variable constant.

A screenshot of a computer

Description automatically generated

Multiple Linear Regression

A white background with black text

Description automatically generated

1. Interpretation:
   1. on average, each coefficient represents change in DV for a one-unit change in corresponding IV, holding all other variables constant.
2. Adding a constant (intercept):
   1. Adding a term in regression equation that accounts for baseline level for DV in case all IVs are set to 0.
   2. Goals:
      1. Allows model to better fit the data by adjusting for baseline level of DV.
      2. It can avoid bias term by preventing model from being forced to pass through biased term.

A math equations on a white background

Description automatically generated

1. Adjusted R-sq

A screenshot of a computer

Description automatically generated

* R-sq: it measures proportion of variance in the DV explained by IV.
  + Ranging from 0 to 1, where higher value indicates a better fit.
* Adjusted R-sq:
  + It adjusts for potential inflation of r-sq when adding more variables.
  + Adding a constant lead to higher r-sq since the model can now capture variance around the mean of DV more effectively.

1. Collinearity in multiple linear regression.
   1. Linear independence: MLR assumption
      1. No feature can be expressed as a weighted sum of the others.
   2. Issue of linear dependence
      1. Model cannot uniquely attributes impact on target variable to any other feature clearly.
      2. It leads to unstable coefficient estimate, where a small change in training data will lead to large change in estimated coefficients.

Potential Mid-Term questions

1. Different data types
   1. Numeric
   2. Ordinal
   3. Categorical
2. Which numeric parameter is sensitive to outliers
   1. Mean: it leads to heavy tailed as skewness
   2. Variance / standard deviation
3. Which parameter is robust to outliers
   1. Median
   2. IQR
4. Array operation can ONLY apply with bitwise operator (&, ~, |) but not logical operator
   1. Boolean comparison operator

A green text on a white background

Description automatically generated

* 1. Numpy array comparison operator

A screenshot of a computer

Description automatically generated

1. Dataframe and series conversion
   1. Dataframe is collection series in pandas
2. Pandas indexing and chaining
   1. .loc [ ‘x’ : ‘y’ , ‘a’ : ‘b’ ]
   2. .iloc [ x : y , a : b ]
3. Importance of random sampling (with replacement) in CI
   1. We need random sampling to ensure subset of population is well representative to the overall population.
   2. Sampling with replacement (bootstrap) aims to create multiple samples from original data to estimate sampling distribution.
   3. In each bootstrap sample, the mean is obtained, and distribution of those different means is used to form a normal distribution and construct 95% confidence interval.
4. Central limit theorem
   1. If all samples are randomly drawn from sample population, then the distribution of sample mean is normally distributed (usually >= 30), regardless of population distribution.
5. How standard normal distribution differ than normal distribution.
   1. Normal distribution:
      1. Mean of sample: at the center of distribution.
      2. Standard deviation: measure spread of distribution away from mean. The larger, the wider curve.
   2. Standard normal distribution
      1. Use Z-score to transform value x from normal distribution (sample value – sample mean) / standard deviation.
6. Supervised / unsupervised learning framworks
7. Correlation for regression
8. Why use reshape (-1, 1)
9. Why need to add a constant
10. Adjusted R-sq
11. Hypothesis testing with 1 or 2 tails
12. Permutation test
13. Confidence interval
14. MAB