NumPy: Exam 3 Reference Sheet

- **Array Creation**
  - np.array(*list*, dtype=*type*)
  - np.arange(*start*, *stop*, *step*)
  - np.linspace(*start*, *stop*, *count*)
  - np.zeros(*shape_tuple*)
  - np.ones(*shape_tuple*)
  - np.full(*shape_tuple*, *value*)
  - np.fromiter(*generator_expression*, dtype=*type*)
- **Array Attributes**:
  - .dtype, .shape, .shape[*index*], .ndim
- **Array Type & Shape Manipulation**:
  - .astype(*type*), .reshape(*tuple*), .copy()
- **Indexing & Slicing**
  - a[*i*], a[*i*, *j*]
  - a[*slice*], a[*slice*, *slice*]
  - a[*bool_array*]
- **Elementwise Operators**
  - Math: +, -, *, /
  - Boolean: &, |, ~
- **Aggregation**
  - .sum(axis=*0 or 1*, keepdims=*bool*)
  - Also .mean, .min, .max, .argmin, .argmax
- **Type Hints**
  - import numpy.typing as npt
  - npt.NDArray[np.int64]
  - npt.NDArray[np.float64]

Pandas and Numpy

- axis=1 means "collapse horizontally"
- axis=0 means "collapse vertically"

Pandas: Exam 3 Reference Sheet

- **Series Attributes**:
  - .shape, .index, .dtype
- **DataFrame Attributes**:
  - .shape, .columns, .index
- **Data Inspection**:
  - .head(*n*), .info(), .describe(include='all')
- **Construction**
  - pd.Series(*list*, index=*labels*)
  - pd.DataFrame(*dict*, index=*labels*)
  - pd.read_csv(*path*)
- **Accessing Elements**
  - s.iloc[*position_indexer*]
  - df.iloc[*row_position_indexer*, *col_position_indexer*]
  - s.loc[*label_indexer*]
  - df.loc[*row_label_indexer*, *col_label_indexer*]
  - An *indexer* can be one item, a list, or a slice.
  - For loc, a boolean Series can also be an *indexer*.
- **Elementwise Operators**
  - Math: +, -, *, /
  - Boolean: &, |, ~
- **Aggregation**
  - .sum(), .mean(), .std(), .min(), .max(), .median()
  - .idxmax(), .idxmin()
  - .mode(), .mode().iloc[0], .count(), .value_counts()
  - For DataFrame, can specify axis for the above
  - df.groupby(*category_col*)[*to_agg_col_or_list*].*aggregator*()
  - df.groupby(*category_col*)[*to_agg_col_or_list*].agg(*function*)
- **map and apply**
  - s.map(*dict*)
  - s.map(*func*)
  - df.apply(*func*, axis=*0 or 1*)