CTMP Python techniques

* print(\_\_file\_\_) 🡺 prints the path to the current .py file
* print(os.path.abspath(\_\_file\_\_)) 🡺 prints absolute path to the current .py file
* os.path.dirname(\_\_file\_\_) 🡺 directory that current .py file resides at
* os.chdir(os.path.dirname(\_\_file\_\_))) 🡺 change current directory to the directory that current .py file resides at
* df[“column”].tolist() 🡺 convert dataframe column into a list
* os.path.exists(“file.txt”) 🡺 check whether this file exists in current directory
* os.remove(“file.txt”) 🡺 remove file
* set(mylist) 🡺 removes the duplicates from list by converting it into a set. (set doesn’t contain duplicates)
* “-“.join(mylist) 🡺 join list values together by putting ‘-‘ inbetween. Also, “”.join(mylist) will join them without without separator
* mystr.lower() 🡺 make string characters all lower
* mystr.upper() 🡺 make string characters all upper
* mystr.replace(“a”, ”b”) 🡺 replace a with b inside string
* mystr.strip() 🡺 remove leading and tailing spaces. can remove other chars too. heck docu
* mystr.lstrip() 🡺 remove from left side (leading)
* mystr.rstrip() 🡺 remove from right side (tailing)
* [w for w in mylist if w not in yourlist] 🡺 subselect from mylist depending on filter
* print(“my name is {}”.format(mystr)) 🡺 printing style
* {v\_term: v\_index for v\_term, v\_index in zip(terms, indices)} 🡺 dictionary filling style
* \_, value = str.split(“:”) 🡺 split the string by separator ‘:’ then assign them.
* df.to\_pickle(directory) 🡺 pickle save the dataframe into mentioned directory
* df[“column”] = df[“column”].apply(lambda x: 1 if x > 4 else 0) 🡺 apply function to dataframe column
* df[“column”].drop\_duplicates() 🡺 drops duplicates from column
* df[“column”].nunique() 🡺 gives number of unique values in column
* np.arange(100) 🡺 give range of 0-99 numbers in a list
* mynumpyarray.flatten() 🡺 flattens the 2D into 1D for example
* np.vectorize + lambda x: np.where(smthg=x) 🡺 check tables.py
* sys.float\_info.max 🡺 maximum float. check docu
* sys.float\_info.min 🡺 minimum float. check docu
* sys.maxsize 🡺 check docu
* sys.maxint 🡺 check docu
* np.argmax() 🡺 return index with max value
* np.argmin() 🡺 return index with min value
* np.negative() 🡺 numerical negative, element-wise
* diff = first == second🡺 compare first and second np arrays elementwise and assign True/False to diff
* np.count\_nonzero(myarr) 🡺 counts number of nonzero elements in array
* dir(object) 🡺 returns all properties and methods of given object.
* getattr(object, x) 🡺 equivalent to object.x
* numpy.argsort(arr) 🡺 return indices that would sort an array
* numpy.setdiff1d(arr1, arr2) 🡺 find the set difference of two arrays
* np.array(myarr) + 1e-10 🡺 to avoid zeros
* np.in1d(arr1, arr2) 🡺 test whether each element of 1D array is also present in second array
* arr[:, np.newaxis] 🡺 np.newaxis is used to increase the dimension. 1D 🡪 2D
* arr = arr.astype(“float64”) 🡺 change type of numpy array
* map(lambda x: x+1, mylist)🡺 map/apply the function to each element in the least