Data Type

Contents

먼저 데이터 타입에 대한 이해가 필요합니다. 주식 데이터 분석에서 활용할 데이터 타입은 숫자(Number), 문자열(String), 날짜(Date), 딕셔너리(Dictionary), 리스트(List), 시리즈(Series), 데이터프레임(DataFrame) 등이 있습니다. 각 타입의 형식은 아래와 같습니다.

```
# Number
n1 = 123
n2 = 234

# String
s1 = 'string'
s2 = 'I am Tom'

# Date
import datetime
d1 = datetime.datetime(2021, 1, 3, 0, 0)
yymmdd = '2021-01-03'
d2 = datetime.datetime.strptime(yymmdd, '%Y-%m-%d')

# Dictionary
dic1 = {'a':11, 'b':12, 'c':13}

# List
l1 = [1,2,3]
l2 =  ['a','b','c']

# Series
import pandas as pd
ss1 = [11,12,13,14,15]
ss2 = pd.Series(ss1)

# DataFrame
c_1 = [1,2,3]
c_2 =  ['a','b','c']
df1 = pd.DataFrame({'col1': c_1, 'col2': c_2})
```

문자열의 첫 글자부터 0, 1, 2 번째 문자 해당합니다. 예를 들어, ‘String’ 이란 문자열을 s1 이란 변수에서 저장한 경우, s1[0] 은 ‘s’ 에 해당하고, s1[1] 은 ‘t’ 에 해당합니다.

```
# String
s1 = 'string'
s2 = 'I am Tom'

print(s1)
print(s2[0], s2[5]) # I am Tom 의 첫번재[0], 6번째[5] 글자 반환. 대부분의 컴퓨터 언어는 0 부터 시작.
```

```
string
I T
```

리스트는 여러 개의 원소를 대괄호 [] 에 넣은 형태입니다. 리스트도 문자열과 동일하게 0 부터 시작합니다. 아래 l1 에서 0 번째 원소는 1 이고, l2 에서는 'a' 입니다. 그리고 [start_point:end_point] 를 형식으로 원소의 일부만 가져올 수 있습니다. 단, [start_point:end_point] 에서 원소는 end_point 전까지 가져오는 사실만 유의하시면 됩니다.

```
# List
l1 = [1,2,3]
l2 =  ['a','b','c','d']

print(l1[0])
print(l2[0])
print(l2[:2]) # 0 ~ 1 번째 문자를 가져옴. 2 번째 포함되지 않음.
print(l2[1:3]) # 1 ~ 2 번째 문자를 가져옴. 3 번째는 포함되지 않음
```

```
1
a
['a', 'b']
['b', 'c']
```

숫자형은 정수형, 소숫점형으로 나눌 수 있으나, 아래와 같이 곧바로 사칙연산이 가능합니다.

```
# Number
n1 = 123
n2 = 234
print(n1*n2)
```

```
28782
```

날짜형 데이터는 주식데이터 분석에 중요한 데이터형식입니다. 활용도가 아주 높습니다. 날짜 데이터를 활용하기 위해서는 먼저 datetime 패키지를 import 합니다. datetime 는 날짜 데이터를 다루기 위한 여러 메소드를 가지고 있습니다. 첫 번째 d1 변수에서 datetime.datetime 함수는 년, 월, 시 등의 숫자를 파이썬 날짜로 변형할 수 있게 해 줍니다. d2 변수는 문자열로 되어 있는 날짜를 파이썬 날짜로 변형하는 한 후 저장을 합니다. strptime 은 문자열을 파이썬 날짜로 변경해주는 메소드입니다. strptime 은 두 개의 인수가 필요한데요. 이 함수의 첫 번째 인자는 날짜 형태의 문자열, 두번 째 인자는 형식 포맷입니다. 문자열 날짜 포맷이 %Y-%m-%d 형태라는 것을 인수로 알려줍니다. d3 는 다른 날짜 형식으로 되어 있는 문자열을 파이썬 날짜로 변경하는 법을 보여줍니다. d4 는 반대로 파이썬 날짜를 다시 문자열로 변경하는 법을 보여주고 있습니다. 이 때 함수는 strpftime 입니다. 마지막으로 timedelta 를 알아보겠습니다. timedelta 는 일정한 시간을 뒤로 이동한 결과를 반환합니다. 예를 들어 d1 에 hours=5 를 추가하면 2021년 1월 3일 0 시에서 2021년 1월 3일 5시로 변경됩니다. 아래 예시는 d1 에서 5 시간을 더 했을 때, 2 일 더했을 때 결과를 보여줍니다.

```
# Date
import datetime

d1 = datetime.datetime(2021, 1, 3, 0, 0)

yymmdd = '2021-01-03'
d2 = datetime.datetime.strptime(yymmdd, '%Y-%m-%d')

time_point = '2021/01/03 19:15:32'
d3 =  datetime.datetime.strptime(time_point, '%Y/%m/%d %H:%M:%S')

d4 = datetime.datetime.strftime(d3, '%Y-%m-%d')

print(d1)
print(d2)
print(d3, type(d3))
print(d4, type(d4))

print(d1 + datetime.timedelta(hours=5))
print(d1 + datetime.timedelta(days=2))
```

```
2021-01-03 00:00:00
2021-01-03 00:00:00
2021-01-03 19:15:32 <class 'datetime.datetime'>
2021-01-03 <class 'str'>
2021-01-03 05:00:00
2021-01-05 00:00:00
```

디셔너리는 key 와 value 가 있는 짝으로 있는 형태입니다. key 를 이용하여 원하는 값을 찾을 수 있어서, 프로그램에서 참조 값을 저장해 둘 때 아주 유용합니다. value 대신에 List 나 DataFrame 형식의 데이터도 넣을 수 도 있습니다. 아래 예시에서는 dic1 에서 키값 'a' 에 해당하는 값은 11 입니다. 새로운 key 와 value 를 넣어서 기존의 Dictionary 에 추가할 수 있습니다. 아래 두 번째 예시는 'd'라는 key 에 value 14 를 추가하는 방법입니다. Dictionary 가 너무 커서 어떤 key 값들이 있는 지 알고 싶을 때는 key() 메소드를 사용합니다.

```
# Dictionary
dic1 = {'a':11, 'b':12, 'c':13}
print(dic1['a'])

dic1['d'] = 14
print(dic1)

print(dic1.keys())
```

```
11
{'a': 11, 'b': 12, 'c': 13, 'd': 14}
dict_keys(['a', 'b', 'c', 'd'])
```

Series 라는 데이터 타입을 이용하기 위해서는 Pandas 패키지를 이용합니다. Pandas 는 테이블 형태의 데이터를 다루는데 정말 강력한 패키지입니다. 먼저 ss1 이라는 리스트를 생성해 보겠습니다. ss1 이라는 리스트에 어떤 메소드를 사용할 수 있는 지 알기 위해서는 dir() 를 이용합니다. dir() 를 하면 Built-in 함수 전체를 알 수 있습니다. append 부터 sort 까지 총 11 개의 함수가 나옵니다. 이 11 개 함수가 List 에서 쓸 수 있는 메소드입니다. 이번에는 ss1 를 Series 로 변경한 후, ss2 에 저장하겠습니다. 그리고 dir() 함수로 호출해 보겠습니다. 많은 메소스가 나열됩니다. 예를 들어 List 값의 평균을 알고 싶은데, List 는 평균을 구하는 메소드가 없습니다. 하지만 Series 에는 mean() 으로 평균값을 구할 수 있습니다.

```
import pandas as pd

ss1 = [11,12,13,14,15]
ss2 = pd.Series(ss1)

print(dir(ss1)) # 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort' 등 사용가능
print('\n')

print(dir(ss2))
print('\n')

''' ss1.mean() --> 에러발생 '''
print(ss2.mean()) # 평균값 13 반환
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__',
'__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__',
'__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']


['T', '_AXIS_LEN', '_AXIS_ORDERS', '_AXIS_REVERSED', '_AXIS_TO_AXIS_NUMBER',
'_HANDLED_TYPES', '__abs__', '__add__', '__and__', '__annotations__', '__array__',
'__array_priority__', '__array_ufunc__', '__array_wrap__', '__bool__', '__class__',
'__contains__', '__copy__', '__deepcopy__', '__delattr__', '__delitem__', '__dict__',
'__dir__', '__divmod__', '__doc__', '__eq__', '__finalize__', '__float__',
'__floordiv__', '__format__', '__ge__', '__getattr__', '__getattribute__',
'__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__', '__iand__',
'__ifloordiv__', '__imod__', '__imul__', '__init__', '__init_subclass__', '__int__',
'__invert__', '__ior__', '__ipow__', '__isub__', '__iter__', '__itruediv__',
'__ixor__', '__le__', '__len__', '__long__', '__lt__', '__matmul__', '__mod__',
'__module__', '__mul__', '__ne__', '__neg__', '__new__', '__nonzero__', '__or__',
'__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__', '__reduce__',
'__reduce_ex__', '__repr__', '__rfloordiv__', '__rmatmul__', '__rmod__', '__rmul__',
'__ror__', '__round__', '__rpow__', '__rsub__', '__rtruediv__', '__rxor__',
'__setattr__', '__setitem__', '__setstate__', '__sizeof__', '__str__', '__sub__',
'__subclasshook__', '__truediv__', '__weakref__', '__xor__', '_accessors',
'_accum_func', '_add_numeric_operations', '_agg_by_level', '_agg_examples_doc',
'_agg_see_also_doc', '_align_frame', '_align_series', '_arith_method', '_as_manager',
'_attrs', '_binop', '_can_hold_na', '_check_inplace_and_allows_duplicate_labels',
'_check_inplace_setting', '_check_is_chained_assignment_possible',
'_check_label_or_level_ambiguity', '_check_setitem_copy', '_clear_item_cache',
'_clip_with_one_bound', '_clip_with_scalar', '_cmp_method', '_consolidate',
'_consolidate_inplace', '_construct_axes_dict', '_construct_axes_from_arguments',
'_construct_result', '_constructor', '_constructor_expanddim', '_convert',
'_convert_dtypes', '_data', '_dir_additions', '_dir_deletions', '_drop_axis',
'_drop_labels_or_levels', '_duplicated', '_find_valid_index', '_flags', '_from_mgr',
'_get_axis', '_get_axis_name', '_get_axis_number', '_get_axis_resolvers',
'_get_block_manager_axis', '_get_bool_data', '_get_cacher',
'_get_cleaned_column_resolvers', '_get_index_resolvers',
'_get_label_or_level_values', '_get_numeric_data', '_get_value', '_get_values',
'_get_values_tuple', '_get_with', '_gotitem', '_hidden_attrs', '_index',
'_indexed_same', '_info_axis', '_info_axis_name', '_info_axis_number', '_init_dict',
'_init_mgr', '_inplace_method', '_internal_names', '_internal_names_set',
'_is_cached', '_is_copy', '_is_label_or_level_reference', '_is_label_reference',
'_is_level_reference', '_is_mixed_type', '_is_view', '_item_cache', '_ixs',
'_logical_func', '_logical_method', '_map_values', '_maybe_update_cacher',
'_memory_usage', '_metadata', '_mgr', '_min_count_stat_function', '_name',
'_needs_reindex_multi', '_protect_consolidate', '_reduce', '_reindex_axes',
'_reindex_indexer', '_reindex_multi', '_reindex_with_indexers', '_replace_single',
'_repr_data_resource_', '_repr_latex_', '_reset_cache', '_reset_cacher',
'_set_as_cached', '_set_axis', '_set_axis_name', '_set_axis_nocheck', '_set_is_copy',
'_set_labels', '_set_name', '_set_value', '_set_values', '_set_with',
'_set_with_engine', '_slice', '_stat_axis', '_stat_axis_name', '_stat_axis_number',
'_stat_function', '_stat_function_ddof', '_take_with_is_copy', '_typ',
'_update_inplace', '_validate_dtype', '_values', '_where', 'abs', 'add',
'add_prefix', 'add_suffix', 'agg', 'aggregate', 'align', 'all', 'any', 'append',
'apply', 'argmax', 'argmin', 'argsort', 'array', 'asfreq', 'asof', 'astype', 'at',
'at_time', 'attrs', 'autocorr', 'axes', 'backfill', 'between', 'between_time',
'bfill', 'bool', 'clip', 'combine', 'combine_first', 'compare', 'convert_dtypes',
'copy', 'corr', 'count', 'cov', 'cummax', 'cummin', 'cumprod', 'cumsum', 'describe',
'diff', 'div', 'divide', 'divmod', 'dot', 'drop', 'drop_duplicates', 'droplevel',
'dropna', 'dtype', 'dtypes', 'duplicated', 'empty', 'eq', 'equals', 'ewm',
'expanding', 'explode', 'factorize', 'ffill', 'fillna', 'filter', 'first',
'first_valid_index', 'flags', 'floordiv', 'ge', 'get', 'groupby', 'gt', 'hasnans',
'head', 'hist', 'iat', 'idxmax', 'idxmin', 'iloc', 'index', 'infer_objects',
'interpolate', 'is_monotonic', 'is_monotonic_decreasing', 'is_monotonic_increasing',
'is_unique', 'isin', 'isna', 'isnull', 'item', 'items', 'iteritems', 'keys', 'kurt',
'kurtosis', 'last', 'last_valid_index', 'le', 'loc', 'lt', 'mad', 'map', 'mask',
'max', 'mean', 'median', 'memory_usage', 'min', 'mod', 'mode', 'mul', 'multiply',
'name', 'nbytes', 'ndim', 'ne', 'nlargest', 'notna', 'notnull', 'nsmallest',
'nunique', 'pad', 'pct_change', 'pipe', 'plot', 'pop', 'pow', 'prod', 'product',
'quantile', 'radd', 'rank', 'ravel', 'rdiv', 'rdivmod', 'reindex', 'reindex_like',
'rename', 'rename_axis', 'reorder_levels', 'repeat', 'replace', 'resample',
'reset_index', 'rfloordiv', 'rmod', 'rmul', 'rolling', 'round', 'rpow', 'rsub',
'rtruediv', 'sample', 'searchsorted', 'sem', 'set_axis', 'set_flags', 'shape',
'shift', 'size', 'skew', 'slice_shift', 'sort_index', 'sort_values', 'squeeze',
'std', 'sub', 'subtract', 'sum', 'swapaxes', 'swaplevel', 'tail', 'take',
'to_clipboard', 'to_csv', 'to_dict', 'to_excel', 'to_frame', 'to_hdf', 'to_json',
'to_latex', 'to_list', 'to_markdown', 'to_numpy', 'to_period', 'to_pickle', 'to_sql',
'to_string', 'to_timestamp', 'to_xarray', 'transform', 'transpose', 'truediv',
'truncate', 'tz_convert', 'tz_localize', 'unique', 'unstack', 'update',
'value_counts', 'values', 'var', 'view', 'where', 'xs']


13.0
```

DataFrame 은 Series의 확장으로, DataFrame 에서 한 개의 Series 는 하나의 Column 이 됩니다. DataFrame 은 여러 개 Column 이 모여 있는 테이블 형태의 데이터 형식입니다. 우선 Dictionary 로 DataFrame 을 만들어 보겠습니다. 두 개의 List - c1_list 와 c2_list 가 두 개의 key - 'c1', 'c2' 대응이 되는 Dictionary, dic_c12 를 생성합니다. 그 다음 pd.DataFrame(dic_c12) 와 같이 DataFrame 으로 데이터 타입을 변경합니다. 출력해보면 테이블 형태로 변경되었음을 알 수 있습니다. 그리고 이 DataFrame 을 df 라는 변수에 저장합니다. df 에서 한 컬럼만 자르면 다시 Series 로 변경됩니다. Series 보다 더 많은 메소드를 이용할 수 있습니다.

```
c1_list = [11,12,13,14,15]
c2_list = ['a','b','c','d','e']

dic_c12 = {'c1': c1_list, 'c2': c2_list}
df = pd.DataFrame(dic_c12) # DataFrame 으로 변경

print(df)
print('\n')

print(df['c1'], type(df['c1']))
```

```
   c1 c2
0  11  a
1  12  b
2  13  c
3  14  d
4  15  e


0    11
1    12
2    13
3    14
4    15
Name: c1, dtype: int64 <class 'pandas.core.series.Series'>
```