SI 506 Lecture 21

1. The dictionary comprehension
2. Transforming values
3. Conditional statements
4. Nested loops
5. Challenges

Data

Today's data is sourced from The World Bank. The Bank assigns the world's economies to four income groups based on the World Bank Atlas method for calculating Gross National Income (GNI) per capita in current USD using data from the previous year. The classifications are updated each year on 1 July. For the current 2022 fiscal year, the classifications are delineated as follows:

Fiscal Year	Classification	GNI per capita (2020)
2021-2022	`Low income`	<= $1,045.00 (USD)
2021-2022	`Lower middle income`	between $1,046.00 and $4,095.00 (USD)
2021-2022	`Upper middle income`	between $4,096.00 and $12,695.00 (USD)
2021-2022	`High income`	>= $12,696.00 (USD)

Source: World Bank Blogs, "New World Bank country classifications by income level: 2021-2022"; World Bank Country and Lendings Groups.

1.0 Dictionary comprehension

> A compact way to process all or part of the elements in an iterable and return a dictionary with the results.

Source: https://docs.python.org/3/glossary.html

1.1 Basic syntax

```
new_dict = {key: val for element in iterable}

new_dict = {key: val for key, value in dict.items()}
```

1.2 Simple example

Below is a list of West African countries that touch the Atlantic Ocean between Senegal and Nigeria. Each country is represented as a tuple. If you needed to convert each tuple to a dictionary employing the ISO-3165-1 alpha-3 country code as the key you could employ a dictionary comprehension to accomplish the task:

```
data = [
        ('Benin', 'BEN'),
        ("Cote d'Ivoire", 'CIV'),
        ('Gambia', 'GMB'),
        ('Ghana', 'GHA'),
        ('Guinea', 'GIN'),
        ('Guinea-Bissau', 'GNB'),
        ('Liberia', 'LBR'),
        ('Nigeria', 'NGA'),
        ('Senegal', 'SEN'),
        ('Sierra Leone', 'SLE'),
        ('Togo', 'TGO')
    ]

west_africa_atlantic = {item[0]: item[1] for item in data}
```

1.3 Transforming values

You can pass a function or call an object method in a dictionary comprehension in order to transform values.

Below is a dictionary (excerpt) of US inflation rates (US city averages, base period 1982-84]) between the years 2000-2021. If you needed to convert the rates to a percentage value rounded to two (2) decimal places you could utilize a dictionary comprehension to create a *new* dictionary to hold the transformed values.

```
data = {
    '2021': 0.024,
    '2020': 0.0125,
    '2019': 0.0181,
    '2018': 0.024399999999999998,
    ...
    '2001': 0.028300000000000002,
    '2000': 0.0338
}

inflation_rates = {
    '2021': 2.4,
    '2020': 1.25,
    '2019': 1.81,
    '2018': 2.44,
    ...
    '2001': 2.83,
    '2000': 3.38
}
```

💡 statista.com also lists the US annual inflation rate for 2021 at 4.7% in a discussion of monthly inflation rates for the twelve month period between February 2021 and February 2022.

2.0 Conditional statements

A dictionary comprehension can specify one or more conditional statements in order to assign a subset of a dictionary to a new dictionary.

```
new_dict = {key: val for element in iterable if condition}

new_dict = {key: val for key, value in dict.items() if condition}
```

The following dictionary comprehension returns inflation rates for the period 2010-2021:

```
inflation_rates = {year: rate for year, rate in inflation_rates.items() if
int(year) > 2009}
```

💡 Note that the typical comprehension variables `key` and `val` or `k` and `v` are conventions; you are free to employ comprehension variable names that better express the nature of the data as in the previous example.

Challenge 01

Task: Write a dictionary comprehension that creates a dictionary of South Asian economies. Each South Asian economy will be represented as a nested dictionary mapped to a key corresponding to its ISO-3165-1 alpha-3 country code.

1. Write a dictionary comprehension that accesses all "South Asian" economies in the list `countries` (region = 'South Asia') and assigns them to a new dictionary named `south_asian_economies` using the country's ISO-3165-1 alpha-3 "country_code" as the key for each country.

   ❗ Convert all country codes to uppercase.

   Structure the new dictionary's key-value pairs as follows:

   ```
   {
   'AFG': {
        'country_name': 'Afghanistan',
        'country_code': 'AFG',
        'region': 'South Asia',
        'income_group': 'Low income',
        'lending_category': 'IDA',
        'emu_or_hipc': 'HIPC'
    },
    ...
   }
   ```

💡 Recall that a dictionary comprehension can work with any *iterable* (in this case a list of nested dictionaries) in order to produce a new dictionary.

Assign the new dictionary to a variable named `south_asian_economies`.

2. Call the function `read_json` and write the `south_asian_economies` list to the file `stu-south_asian_economies.json`.

   💡 You can enhance readability by writing dictionary comprehensions that exceed 80-100 characters and/or feature complex conditions or other expressions across multiple lines vertically.

```
new_dict = [
    expression
    for element in iterable
    if condition
    ...]
```

2.1 `if-else` statements

You can employ `if-else` logic in a dictionary comprehension. The `if-else` logic is placed *before* the `for` statement and employs the ternary form of the `if-else` operator.

```
new_dict = {
    key: (some_val_if_true if condition else some_other_val)
    for key, val in dict_.items()
}

new_dict = {
    (key if condition else default_key): (some_val_if_true if condition
else some_other_val)
    for key, val in dict_.items()
}
```

The example below iterates over the `south_asian_economies` key-value pairs replacing each value with either the lending categories associated with the International Development Association (IDA)" or the International Bank for Reconstruction and Development (IBRD).

```
south_asian_lending = {
    key: ('IDA' if val['lending_category'] in ('IDA', 'Blend') else
'IBRD')
    for key, val in south_asian_economies.items()
}
```

2.2 `if-elif-else` statements

The `elif` statement is *not* recognized inside a dictionary comprehension. You can mimic `if-elif-else` logic by employing multiple `else` statements.

Dividing World economies into three categories ("High income", "Middle income", or "Low income") can be accomplished by adding an additional `else` statement to the dictionary comprehension:

```
country_income = {
    country['country_code'].upper(): ('High income' if
country['income_group'] == 'High income'
    else 'Middle income' if country['income_group'] in ('Upper middle
income', 'Lower middle income')
    else 'Low income')
    for country in countries
    }
```

💡 If dictionary comprehension readability is concern then consider relocating the business logic (e.g., categorizing economies) to a function and then use it to transform the data by calling it from inside the dictionary comprehension.

```
# Delegate business logic to function
country_income = {ctry['country_code'].upper(): categorize_economy(ctry)
for ctry in countries}
```

2.3 Nested loops

You can embed nested loops in a dictionary comprehension. The outer loop is listed first followed by the inner loop:

```
new_dict = {key: val for outer_element in outer_loop for inner_element in
inner_loop if condition}
```

In the following example, a new dictionary comprising "country indicators" for European Union (EU) members is created using nested loops in a dictionary comprehension.

```
eu_countries = {
    country['country_code']: country
    for group in groups
    for country in country_indicators
    if group['group_code'] == 'EUU' and group['country_code'] ==
country['country_code']
}
```

💡 Nested dictionary comprehensions can get ugly. Check out this example in stackoverflow.com:

```
data = {outer_k: {inner_k: myfunc(inner_v) for inner_k, inner_v in
outer_v.items()} for outer_k, outer_v in outer_dict.items()}
```

As the contributor notes: "For the sake of readability, don't nest dictionary comprehensions and list comprehensions too much."

Challenge 02

Task Access Chinese economic indicators relating to population and assign to a new dictionary.

1. Call the function `read_json` and read in the data contained in the file `wb-indicators-dict-china-2019.json`. Assign to a variable named `china`.

   Observations

   1. Note that the `china` dictionary includes a nested dictionary of nested "country_indicators" dictionaries.

   2. Note that each indicator key possesses a common format.

2. Employ a dictionary literal along with a dictionary comprehension to create a new dictionary named `china_pop_indicators`.

   Include an `if` statement in your dictionary comprehension that compares a segment of the key (type `str`) to the string `POP`. If the equality check resolves to `True` the dictionary comprehension will include the key-value pair in the nested dictionary assigned to the key "population_indicators".

   Pattern your dictionary literal as follows:

   ```
   {
        'country_code': < value >,
        'country_name': < value >,
        'population_indicators': {< dictionary comprehension >}
   }
   ```

3. After assigning the new dictionary to `china_pop_indicators` call `write_json` and write the data to a file named `stu-china-pop_indicators.json`.

   ```
   {
     "country_code": "CHN",
     "country_name": "China",
     "population_indicators": {
       "EN.POP.DNST": {
         "indicator_name": "Population density (people per sq. km of land
   area)",
         "indicator_value": "149.367573167744"
       },
       "SP.POP.TOTL.FE.IN": {
   ```

```
        "indicator_name": "Population, female",
        "indicator_value": "685480283"
      },
      "SP.POP.TOTL.FE.ZS": {
        "indicator_name": "Population, female (% of total population)",
        "indicator_value": "48.6934979729146"
      },
      "SP.POP.TOTL.MA.IN": {
        "indicator_name": "Population, male",
        "indicator_value": "722264717"
      },
      "SP.POP.TOTL.MA.ZS": {
        "indicator_name": "Population, male (% of total population)",
        "indicator_value": "51.3065020270854"
      },
      "SP.POP.TOTL": {
        "indicator_name": "Population, total",
        "indicator_value": "1407745000"
      }
    }
  }
```

Challenge 03

Task Access East Asian and Pacific economic indicators relating to population and assign to a new dictionary.

1. Repeat Challenge 02 above for all East Asian and Pacific countries (group_code = 'EAS'). In other words write a dictionary comprehension that loops over both the `groups` list and `country_indicators` list and creates a new dictionary with the following structure:

```
{
     <country_code>: {
         'country_code': < value >,
         'country_name': < value >,
         'population_indicators': {< dictionary comprehension >}
         },
     <country_code>: {
         'country_code': < value >,
         'country_name': < value >,
         'population_indicators': {< dictionary comprehension >}
         },
    }
```

   Assign the new dictionary to a variable named `east_asia_pop_indicators`.

2. After assigning the new dictionary to `east_asia_pop_indicators` call `write_json` and write the data to a file named `stu-east_asia-pop_indicators-comp.json.json`.