本講座でのテスト対象のプログラムと、解決したい課題

zodiac.py の概要

ユニットテストについての解説に入る前に、本講座での主たるテスト対象たる `zodiac.py` の仕様について説明します。
(`__doc__` でモジュールやクラス、関数等についての情報を docstring から取得できるので、これも活用してください)

誕生日から、その誕生日の星座の名前を返すプログラムを作りました。
ファイル `zodiac.py` です。

誕生日からその誕生日の星座を返すプログラムを書くのは案外大変です。
大変な理由は、期間が年をまたぐ星座があるからです。

今回採用した各星座の期間は以下のとおりです。(宗派によって違いはあります)
見て分かるとおり、山羊座は12月22日から1月19日までです。

星座名	開始日	最終日	年をまたぐか？
水瓶座	1月20日	2月18日	
魚座	2月19日	3月20日	
牡羊座	3月21日	4月19日	
牡牛座	4月20日	5月20日	
双子座	5月21日	6月20日	
蟹座	6月21日	7月22日	
獅子座	7月23日	8月22日	
乙女座	8月23日	9月22日	
天秤座	9月23日	10月22日	
蠍座	10月23日	11月21日	
射手座	11月22日	12月21日	
山羊座	12月22日	1月19日	年をまたぐ

とはいえ、いちおうきちんと動作しそうなプログラムを作ることができました。
以下の `get_zodiac_sign_name` です。

```
from zodiac import get_zodiac_sign_name

result = get_zodiac_sign_name(month=1, day=1)
print(result)  # 山羊座

result = get_zodiac_sign_name(month=1, day=25)
print(result)  # 水瓶座
```

今回ユニットテストを書く対象は、この関数 `get_zodiac_sign_name` と、これが内部で呼び出している関数など、関連のすべてのプログラムです。

実装

おおまかな方針

この関数 `get_zodiac_sign_name` の参照元として使えるデータとして、以下があります。
これは、星座の名前と、その星座の最終日を CSV 形式で記述したものです。

resouces/zodiac.csv

```
星座名,月,日
山羊座,1,19
水瓶座,2,18
魚座,3,20
牡羊座,4,19
牡牛座,5,20
双子座,6,20
蟹座,7,22
獅子座,8,22
乙女座,9,22
天秤座,10,22
蠍座,11,21
射手座,12,21
```

まずは、このCSVファイルを読みだし、その情報を元にして、まずは以下のような辞書を作ります。

```
zodiac_part_dict = {
    "山羊座": {"month": 1, "day": 19},
    "水瓶座": {"month": 2, "day": 18},
    "魚座": {"month": 3, "day": 20, },
    "牡羊座": {"month": 4, "day": 19, },
    "牡牛座": {"month": 5, "day": 20, },
    "双子座": {"month": 6, "day": 20, },
    "蟹座": {"month": 7, "day": 22, },
    "獅子座": {"month": 8, "day": 22, },
    "乙女座": {"month": 9, "day": 22, },
    "天秤座": {"month": 10, "day": 22, },
    "蠍座": {"month": 11, "day": 21, },
    "射手座": {"month": 12, "day": 21, },
}
```

そして、上記の辞書を元にして、以下のような辞書を作ります。

```
zodiac_full_dict = {
    '山羊座': {'from': {'month': 12, 'day': 22}, 'to': {'month': 1, 'day': 19}},
    '水瓶座': {'from': {'month': 1, 'day': 20}, 'to': {'month': 2, 'day': 18}},
    '魚座': {'from': {'month': 2, 'day': 19}, 'to': {'month': 3, 'day': 20}},
    '牡羊座': {'from': {'month': 3, 'day': 21}, 'to': {'month': 4, 'day': 19}},
    '牡牛座': {'from': {'month': 4, 'day': 20}, 'to': {'month': 5, 'day': 20}},
    '双子座': {'from': {'month': 5, 'day': 21}, 'to': {'month': 6, 'day': 20}},
    '蟹座': {'from': {'month': 6, 'day': 21}, 'to': {'month': 7, 'day': 22}},
    '獅子座': {'from': {'month': 7, 'day': 23}, 'to': {'month': 8, 'day': 22}},
    '乙女座': {'from': {'month': 8, 'day': 23}, 'to': {'month': 9, 'day': 22}},
    '天秤座': {'from': {'month': 9, 'day': 23}, 'to': {'month': 10, 'day': 22}},
    '蠍座': {'from': {'month': 10, 'day': 23}, 'to': {'month': 11, 'day': 21}},
    '射手座': {'from': {'month': 11, 'day': 22}, 'to': {'month': 12, 'day': 21}}
}
```

これで準備ができました。

あとは、誕生日の日付を受け取ると、変数 `zodiac_full_dict` を調べて星座を返す関数を作ります。

具体的な実装

上記の方針に基づいた実装を行いました。
`zodiac.py` は、以下のような関数群でを含むモジュールです。

項目	引数	概要
`get_zodiac_part_dict`		resources/zodiac.csv から、 データを取得する
`get_first_month_day_of_zodiac_sign`	`month` , `zodiac_part_dict`	各星座の最終日を含む月から、 その星座の最初の日を返す
`create_zodiac_full_dict`	`zodiac_part_dict`	各星座の期間を辞書型で返す
`get_zodiac_sign_name_dict`	`month` , `day` , `zodiac_full_dict`	日付と辞書を受け取り、 星座を示す文字列を返す
`get_zodiac_sign_name`	`month` , `day`	日付を受け取り、 星座を示す文字列を返す

`get_zodiac_part_dict` , `create_zodiac_full_dict` , `get_zodiac_sign_name_dict` は、 `get_zodiac_sign_name` から呼び出される関数です。
`get_first_month_day_of_zodiac_sign` は、 `create_zodiac_full_dict` から呼び出される関数です。

つまり、全体的な処理の流れは以下のようになります。

- `get_zodiac_sign_name` は、誕生日の月と日を受け取る。
  - `get_zodiac_part_dict` を実行して、各星座の最終日だけの辞書を取得する。
  - `create_zodiac_full_dict` を実行して、各星座の開始日と最終日の辞書を取得する。
    - `get_first_month_day_of_zodiac_sign` を実行して、各星座の開始日を取得する。
  - `get_zodiac_sign_name_dict` を実行して、誕生日に相当する星座を取得する。

解決したい課題 - ユニットテストを書きたい理由

ここまでに紹介したような実装のプログラムを書いて、最終的には以下のプログラムを実行すれば求める値を得られるようになったようです。

```
from zodiac import get_zodiac_sign_name

result = get_zodiac_sign_name(month=1, day=1)
print(result)  # 山羊座

result = get_zodiac_sign_name(month=1, day=25)
print(result)  # 水瓶座
```

上に示したとおり、この関数は、少なくとも1月1日と1月25日については正しく動作してるようです。

しかし、ほかの日付については、正しく動作しているかどうかはわかりません。

実際、「ほぼ間違いなく動作するだろう」という確信を持つには、365日すべてとは言わずとも、最低でも以下の類いの日付についてテストする必要があるでしょう。

テストの趣旨	日付の例	期待する結果
年明けの水瓶座初日以前について動作確認をする	1月1日	山羊座
任意の星座の初日について動作確認をする	1月20日	水瓶座
任意の星座の中日について動作確認をする	1月25日	水瓶座
任意の星座の最終日について動作確認をする	2月18日	水瓶座
年末の射手座最終日以降について動作確認をする	12月25日	山羊座

しかし、これらすべての日付について正しく動作するかどうかを手動で確認するのはなかなか面倒です。

また、一度動作確認したとしても、関数の内部仕様の変更等をした場合は、再度すべての日付について確認しなくてはなりません。

とはいえ、手直しの都度手動で動作確認するとなると、かなりの手間です。
どうしても、抜け漏れも発生してしてしまうでしょう。

ユニットテストは、このような手動での動作確認の手間を省き、抜け漏れを防ぐためのものです。

たとえば、 `get_zodiac_sign_name` やこれが呼び出している関数のそれぞれについて、正しく動作するかどうかを網羅的に検証するテストコードを書いておきます。

すると、関数の動作確認は、テストコードを実行するだけで簡単にできるようになります。