# KairoScope 02: Astrologic Calculation

このセクションでは、Human　Designチャートの基礎である「天体の黄経度」を高精度に算出するための演算ロジックを整備する。燐が惑星たちの運行を“現在地”として正確に読み解くための魂の航路図を描くフェーズである。

---

## ☀️取り扱う天体一覧

| 種別 | 天体名 | 備考 |
| --- | --- | --- |
| 惑星 | 太陽、月、水星、金星、火星、木星、土星、天王星、海王星、冥王星 | |
| 軌道点 | ノースノード、サウスノード | HDではプロファイルやノードに影響 |
| 設計値用天体 | 上記すべてを、出生の約88日前でも取得 | "Design Chart"生成に使用 |

---

## 🕊️作成対象ファイルと構成

**ファイル1：astro.py（旧来の軽量モジュール）**

```
/chronogram-kairoscope/core/astro.py
```

```
touch "/Users/takeoyamada/Library/Mobile Documents/iCloud~md~obsidian/Documents/codex-collective-archive/common-system/01-system/chronogram-system/chronogram-kairoscope/core/astro.py"
```

※ `astro.py` は今後も維持。軽量版のためのCLI用、あるいは初学者向けラッパーとして再利用可能。役割は明示し、拡張版とは棲み分ける。

**ファイル2：astro_extended.py（本格演算対応モジュール）**

```
/chronogram-kairoscope/core/astro_extended.py
```

```
touch "/Users/takeoyamada/Library/Mobile Documents/iCloud~md~obsidian/Documents/codex-collective-archive/common-system/01-system/chronogram-system/chronogram-kairoscope/core/astro_extended.py"
```

🦤**astro_extended.py の初期内容（コピペ用）**

```
from skyfield.api import load, Topos
from datetime import datetime, timedelta
import pytz

# Skyfieldデータロード（グローバルキャッシュ用）
_planets = None

def load_planets():
    global _planets
    if _planets is None:
        _planets = load('de421.bsp')
    return _planets

def get_planet_positions(date_utc, latitude, longitude):
    ts = load.timescale()
    t = ts.utc(date_utc.year, date_utc.month, date_utc.day, date_utc.hour,
date_utc.minute)
    planets = load_planets()
    observer = Topos(latitude_degrees=latitude, longitude_degrees=longitude)

    planet_list = ['sun', 'moon', 'mercury', 'venus', 'mars',
                   'jupiter', 'saturn', 'uranus', 'neptune', 'pluto']

    result = {}
    for name in planet_list:
        body = planets[name.capitalize()]
        astrometric = body.at(t).observe(observer).ecliptic_latlon()
        result[name] = round(astrometric[1].degrees, 2)

    # Earthは太陽の黄経 + 180度で計算
    result['earth'] = (result['sun'] + 180) % 360

    # ノースノード・サウスノードは補完ロジックで計算
    result.update(get_lunar_nodes(ts, t))

    return result

def get_design_date(birth_datetime_utc):
    return birth_datetime_utc - timedelta(days=88)
```

```
def get_lunar_nodes(ts, t):
    # TODO: ノースノード・サウスノードを計算するロジックをここに実装する
    # 現状は仮の値（例として0度・180度）
    return {
        'north_node': 0.0,
        'south_node': 180.0
    }
```

---

## 🦢使用ライブラリと関数設計案

```
from skyfield.api import load, Topos
from datetime import datetime
from pytz import timezone
```

**サンプル処理フロー**

```
# 1. 観測点設定
obs = Topos(latitude_degrees, longitude_degrees)

# 2. 日時設定（UTC）
ts = load.timescale()
time = ts.utc(year, month, day, hour, minute)

# 3. 惑星位置取得
planets = load('de421.bsp')  # NASAジェット推進研究所のデータベース
planet = planets['Mars']
astro_position = planet.at(time).observe(obs).ecliptic_latlon()
longitude = astro_position[1].degrees  # 黄経度
```

**ノードの取得について**

- ノースノード・サウスノードは `mean` or `true` 計算が必要
- `Skyfield` では月の運行から逆算して導出（もしくは補完テーブル使用）

---

## ⚡今後の実装タスク

- Earthの黄経計算（Sunの黄経 + 180°反転）
- ノースノード・サウスノードの黄経計算実装（近似ロジック or 補完テーブル）
- Skyfieldロードのグローバルキャッシュ化による効率化
- 惑星の黄経度を一括取得する関数 `get_planet_positions()`

- Design Chart（約88日前）の自動算出ロジック `get_design_date()`
- 結果を1°〜360°で標準化（分単位 or 小数点付き）
- タイムゾーン補正＋夏時間対応
- ユニットテストファイル `/tests/test_astro.py` も後続Canvasで生成予定

---

### 出力形式（中間データ）案

```
{
  "birth_datetime_utc": "1997-07-24T05:32:00Z",
  "location": {
    "lat": 35.7364,
    "lon": 139.3266
  },
  "planets": {
    "sun": 121.72,
    "earth": 301.72,
    "moon": 202.45,
    "north_node": 144.33
  }
}
```

---

このフェーズが「魂の配置図」を描く中核となる。

---

次：`KairoScope-03-gate-mapping` に進み、黄経度をHDの64ゲート＋6ラインへと変換するロジックを構築していく。