PyEphem

Krzysztof Katarzyński

Centrum Astronomii UMK

Proponuję opodatkować wszystkich cudzoziemców, którzy nie mieszkają w naszym kraju!

Latający Cyrk Monty Pythona

PyEphem

Moduł **PyEphem** (rhodesmill.org/pyephem/) pozwala wykonywać wiele profesjonalnych obliczeń astronomicznych. Można przy jego pomocy:

- wyliczać pozycje Słońca, Księżyca, planet oraz ich księżyców,
- ustalać aktualne położenie asteroid, komet oraz sztucznych satelitów (należy podać elementy orbitalne),
- wyznaczać wschody, zachody różnych obiektów w zależności od pozycji obserwatora na powierzchni Ziemi,
- odnajdywać w jakim gwiazdozbiorze znajduje się dany obiekt,
- wyliczać pozycję gwiazd, galaktyk itd.

PyEphem bazuje na procedurach napisanych dla programu **XEphem** (www.clearskyinstitute.com/xephem/), który od wielu lat dostępny jest dla użytkowników systemów Unix-owych.

Jak zainstalować

Moduł **PyEphem** w wersji źródłowej lub dla Windows (32 bit) można pobrać ze strony głównej programu: rhodesmill.org/pyephem/

W systemach Linux opartych o archiwa debianowe można do instalacji modułu użyć programu **pip**

```
sudo apt-get install pip # jeżeli pip nie jest zainstalowany
sudo pip install pyephem
```

Instalator dla Windows 32 i 64 bit można znaleźć również na stronie: www.lfd.uci.edu/ gohlke/pythonlibs/#pyephem

Importowanie modułu oraz pozycja obserwatora

Moduł **PyEphem** dostępny jest pod nazwą **ephem**, którą w trakcie importowania warto zmienić na **ep** aby skrócić programowanie

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# IMPORTOWANIE
from pylab import *
# dla wygody zmieniamy nazwę ephem na ep
import ephem as ep

# OBSERWATOR
# tworzymy strukturę, w której przechowywane
# będą dane o pozycji obserwatora
obs = ep.Observer()
obs.lon = "18.56406" # długość geograficzna
obs.lat = "53.09546" # szerokość geograficzna
obs.elevation = 133.61 # wysokość n.p.m
```

Funkcje oraz struktury modułu **ephem** wywołujemy pisząc **ep.funkcja()**. W pierwszej kolejności definiujemy położenie obserwatora (instrumentu obserwacyjnego). W tym celu tworzymy odpowiednią strukturę (**struktura = ep.Observer()**) i wypełniamy jej pola (**struktura.pole = wartość**).

1 Obiekty oraz ich położenie

Wszystkie ważne obiekty astronomiczne takie jak Słońce, Księżyc, planety oraz ich księżyce można tworzyć funkcją **struktura = ep.NazwaObiektu()**. Obowiązują nazwy anglojęzyczne.

```
# OBIEKT
# tworzymy strukturę, w której przechowywane będą dane
# o pozycji obiektu
ks = ep.Moon() # Księżyc
```

Po utworzeniu obiektu możemy wyliczyć jego aktualne położenie podając informacje na temat obserwatora, które znajdują się utworzonej wcześniej strukturze **obs**.

```
# OBLICZENIA
# obliczmy aktualną pozycję obiektu dla utworzonego
# wcześniej obserwatora
ks.compute(obs)
```

Jak widać można zdefiniować kilku różnych obserwatorów i policzyć współrzędne wybranego obiektu dla różnych miejsc na Ziemi. Może to być przydatne przy planowaniu kampanii obserwacyjnych prowadzonych przez różne obserwatoria.

1 Odczyt współrzędnych

Obliczone współrzędne odczytujemy z pól struktury obiektu

```
# WYLICZONE WSPÓŁRZĘDNE
print "Aktualna pozycja Księżyca"
print "-------------------------"
# wypisujemy rektascensję i deklinację
print "RA:",  ks.ra
print "Dec:", ks.dec
# oraz współrzędne azymutalne
print "-------------------------"
print "Az:", ks.az
print "El:", ks.alt
```

Wyliczone wartości podawane są w formacie godziny:minuty:sekundy lub stopnie:minuty:sekundy łuku dla aktualnego czasu UT na epokę 2000.

```
Aktualna pozycja Księżyca
-------------------------
RA: 23:42:46.99
Dec: 0:31:15.1
-------------------------
Az: 251:35:22.6
El: 14:02:16.7
```

Współrzędne w stopniach

Wszystkie współrzędne wyliczane przez procedury modułu PyEphem podawane są w **radianach**. Jednak w momencie gdy chcemy np. wypisać ich wartość, radiany automatycznie zamieniane są do odpowiedniego formatu (godziny, minuty, sekundy w przypadku rektascensji lub stopnie, minuty, sekundy łuku dla innych współrzędnych).
Jeżeli wyliczone współrzędne chcemy wykorzystać na wykresie to warto przeliczyć je na stopnie używając funkcji **degrees**.

```
# współrzędne azymutalne w stopniach jako liczba rzeczywista
print "-------------------------"
print "Az:", degrees(ks.az)
print "El:", degrees(ks.alt)
```

Azymut i elewacja w stopniach:

```
Az: 251.589608266
El: 14.0379640039
```

Miasta oraz dowolny czas

Jeżeli nie znamy dokładnego położenia obserwatora i równocześnie potrzebujemy jedynie szacunkowe wartości współrzędnych, aby np. ocenić czy dany obiekt jest akurat widoczny, to możemy wykorzystać jedno ze 122 miast, których pozycje zostały zapisane w module PyEphem.

```
# OBSERWATOR
obs = ep.city("London")
print "długość geograficzna:  ", obs.lon
print "szerokość geograficzna:", obs.lat
```

Każdemu obserwatorowi możemy przypisać dowolną datę i czas:

```
# WŁASNA DATA I CZAS UT
obs.date = "2014/05/24 12:00:00"
```

należy jedynie pamiętać aby cyfry daty były rozdzielone znakiem / a czas dwukropkiem.

3 Ruch dzienny Słońca – początek programu

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# IMPORTOWANIE
from pylab import *
import ephem as ep

# OBSERWATOR
obs = ep.city("Warsaw")

# OBIEKT
sun = ep.Sun()

# GODZINA
tm = linspace(0, 24, 25)

# BIEGUNOWY UKŁAD WSPÓŁRZĘDNYCH
pt = subplot(111, polar=True)
```

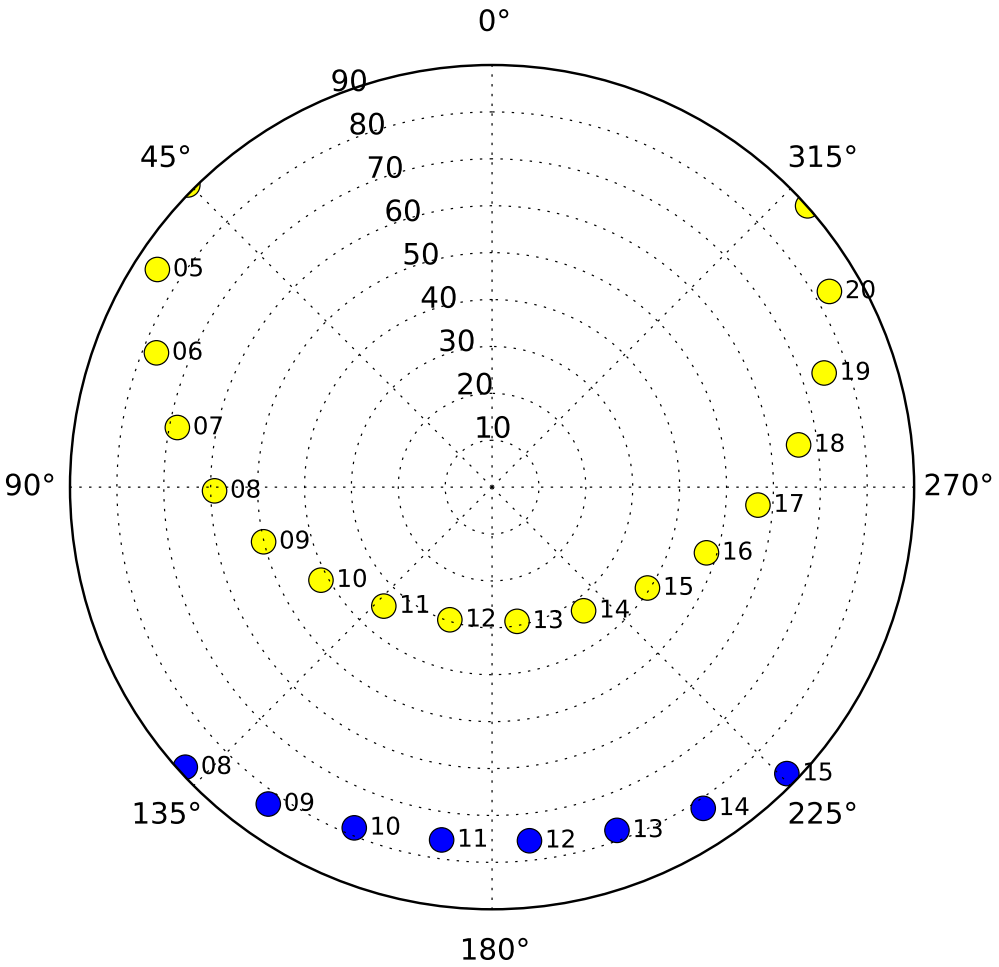
Ruch dzienny Słońca – przesilenie letnie

```
# GŁÓWNA PĘTLA
for t in tm:
    # PRZESILENIE LETNIE
    # zmieniamy godzinę
    obs.date = "2014/06/21 %02d:00:00" % t
    # obliczamy współrzędne
    sun.compute(obs)
    # pobieramy współrzędne azymutalne - azymut w radianach
    az = float(repr(sun.az))
    el = degrees(float(repr(sun.alt)))
    # rysowanie - elewację zamieniamy na odległość zenitalną
    pt.plot([az], [90-el], ls="", marker="o", c="yellow", \
    markersize=10)
    # czas lokalny UT +2 godziny w lecie
    if el > 0:
      pt.text(az, 90-el, "  %02d" % (t+2), fontsize=10, \
      ha='left',  va='center')
```

Ruch dzienny Słońca – przesilenie zimowe

```
    # PRZESILENIE ZIMOWE - powtarzamy obliczenia "w grudniu"
    obs.date = "2014/12/22 %02d:00:00" % t
    sun.compute(obs)
    az = float(repr(sun.az))
    el = degrees(float(repr(sun.alt)))
    pt.plot([az], [90-el], ls="", marker="o", c="blue", \
    markersize=10)
    # czas lokalny UT +1 godzina w zimie
    if el > 0:
      pt.text(az, 90-el, "  %02d" % (t+1), fontsize=10, \
      ha='left',  va='center')

# ograniczamy odległość zenitalną do 90 stopni - horyzont
pt.set_rmax(90.0)
# ustawiamy północ na górze wykresu
pt.set_theta_zero_location("N")
savefig("RuchSlonca.pdf")
show()
```

Ruch dzienny Słońca – rysunek



W programie warto też zadać inne miasto np. `obs = ep.city("Cairo")`

4 Ruch wsteczny Marsa – początek programu

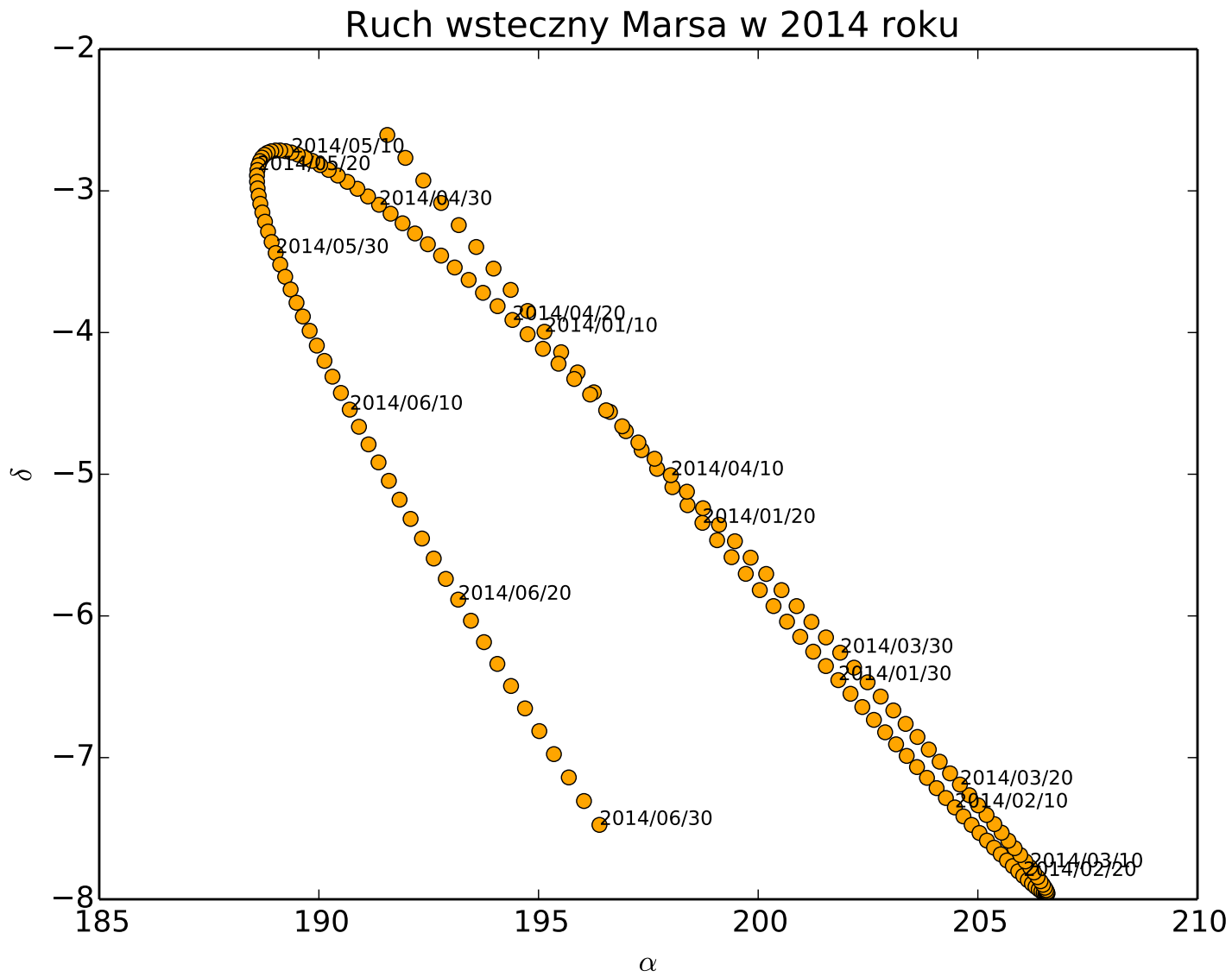
Już w starożytności zauważono, że ruch niektórych obiektów na niebie ma odmienny charakter od ruchu większości tzw. gwiazd stałych. Obiekty takie nazywano gwiazdami błądzącymi lub wędrowcami (gr. planetami). Dziś wiemy, że „błądzenie” planet to efekt złożenia ich ruchu z ruchem Ziemi wokół Słońca. Możemy prześledzić to zjawisko na przykładzie Marsa.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# IMPORTOWANIE
from pylab import *
import ephem as ep

# potrzebne będą dwie dodatkowe funkcje z modułu datetime
from datetime import datetime, timedelta

# OBSERWATOR
obs = ep.city("Warsaw")
# MARS
mr = ep.Mars()
```

4 Ruch wsteczny Marsa – główna pętla

```
for i in range(0, 181):
    # zmieniamy datę co jeden dzień przez pół roku
    dt = datetime(2014, 1, 1) + timedelta(i)
    ds = "%d/%02d/%02d" % (dt.year, dt.month, dt.day)
    print "dzien roku:", i+1, ds
    # ustawiamy datę obserwatora i obliczamy współrzędne
    obs.date = ds
    mr.compute(obs)
    ra = degrees(float(repr(mr.ra)))
    de = degrees(float(repr(mr.dec)))
    # rysujemy pozycje
    plot([ra], [de], c="orange", marker="o")
    # dodajemy opis daty średnio co 10 dni
    if (dt.day%10) == 0: text(ra, de, ds, fontsize=8)

# opis rysunku
xlabel("$\\alpha$")
ylabel("$\\delta$")
title("Ruch wsteczny Marsa w 2014 roku")
savefig("RuchMarsa.pdf")
show()
```

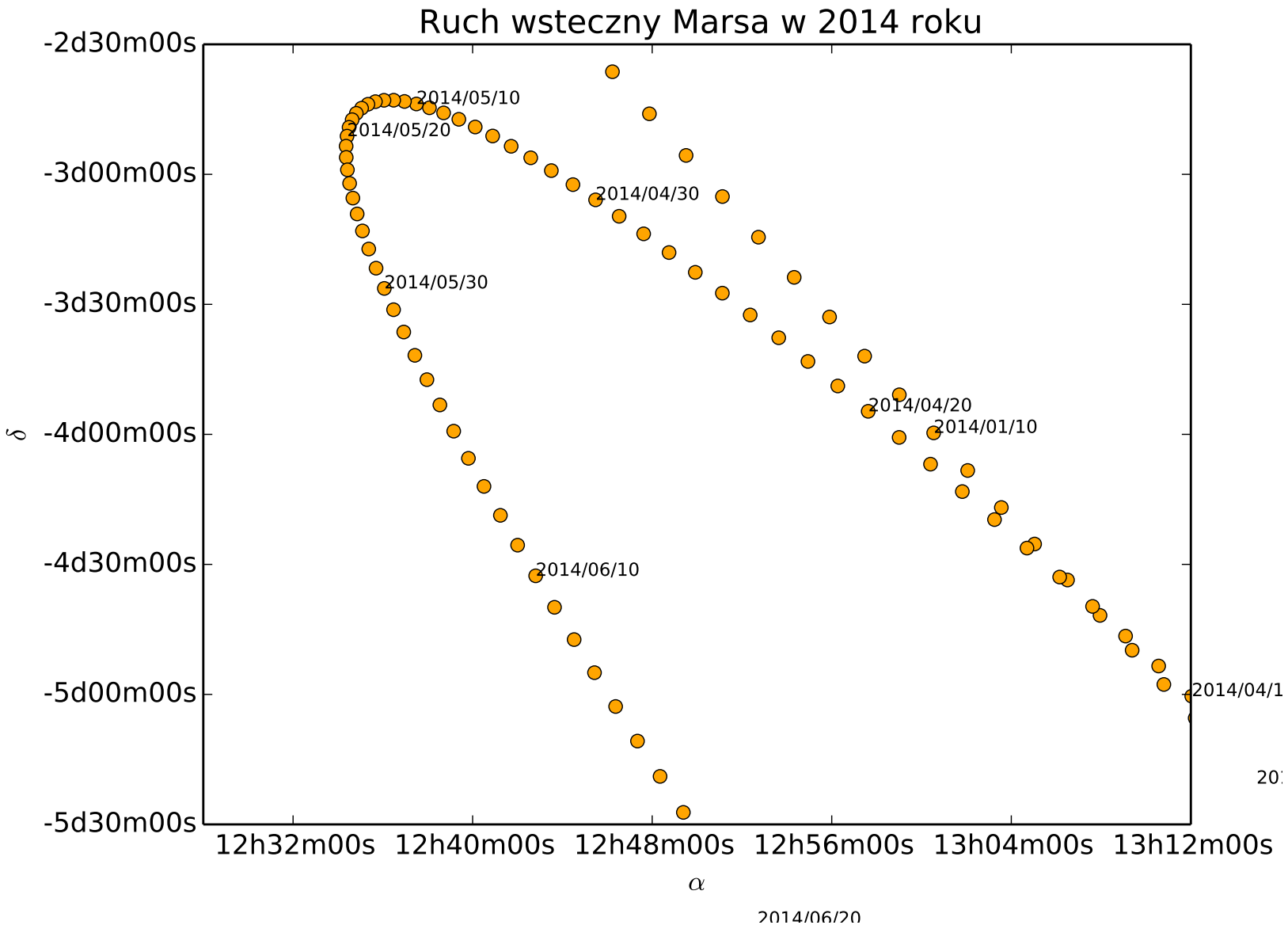
Ruch wsteczny Marsa – rysunek



Ruch wsteczny Marsa – zmiana opisu osi

```
# zamiana rektascensji podanej w stopniach
# na format godzina, minuta, sekunda
def RAd2hms(x, loc):
    h = x//15
    m = int(((x-h*15.0)/15.0)*60.0)
    s = ((x-h*15-m/4.0)/15.0)*3600.0
    return "%02dh%02dm%02ds" % (h, m, s)

# zamiana deklinacji podanej w stopniach
# na format stopień, minuta, sekunda łuku
def DEd2dms(x, loc):
    d = int(fabs(x))
    m = int((fabs(x)-d)*60)
    s = (fabs(x)-d-m/60.0)*3600.0
    if x<0: d=-1*d
    return "%02dd%02dm%02ds" % (d, m, s)

# opis rysunku
xlabel("$\\alpha$") # poniżej wstawiamy naszą funkcję
gca().xaxis.set_major_formatter(FuncFormatter(RAd2hms))
ylabel("$\\delta$") # poniżej wstawiamy naszą funkcję
gca().yaxis.set_major_formatter(FuncFormatter(DEd2dms))
```

Zmiana opisu osi – rysunek



6 Ruch księżyców Jowisza – początek programu

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# IMPORTOWANIE
from pylab import *
import ephem as ep

# TWORZYMY OBIEKTY
Io = ep.Io()
Eu = ep.Europa()
Ga = ep.Ganymede()
Ca = ep.Callisto()

# PUSTE TABLICE DO KTÓRYCH BĘDZIEMY
# DOPISYWAĆ KOLEJNE WSPÓŁRZĘDNE
y    = [] # wspólna wsp. y
xIo  = [] # współrzędna x dla
xEu  = [] # każdego księżyca
xGa  = [] # osobno
xCa  = []
```

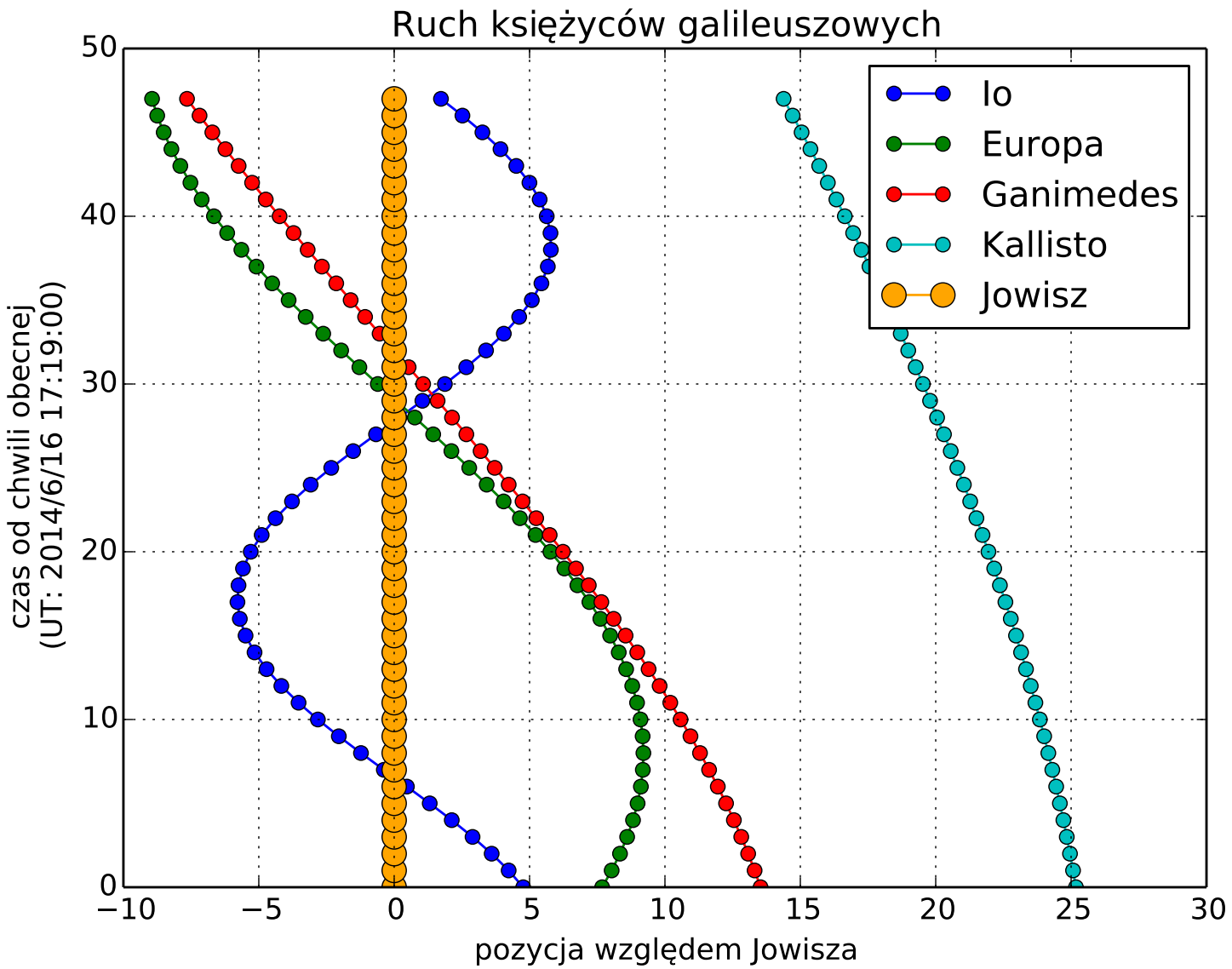
6 Ruch księżyców Jowisza – główna pętla

```
# krok czasowy - godzina
dt = ep.hour
# czas początkowy
ts = ep.now()
# czas aktualny
tm = ts

# GŁÓWNA PĘTLA OBLICZENIOWA
for i in range(2*24):
    # obliczamy wsp. y-ową
    y.append((tm-ts)*24.0)
    # obliczamy wsp. x-owe
    Io.compute(tm)
    Eu.compute(tm)
    Ga.compute(tm)
    Ca.compute(tm)
    # dodajemy wyliczenia do tablic
    xIo.append(Io.x)
    xEu.append(Eu.x)
    xGa.append(Ga.x)
    xCa.append(Ca.x)
    # zwiększamy czas o godzinę
    tm += dt
```

Ruch księżyców Jowisza – wykres

```
# rysujemy wynik
plot(xIo, y, marker="o", label="Io")
plot(xEu, y, marker="o", label="Europa")
plot(xGa, y, marker="o", label="Ganimedes")
plot(xCa, y, marker="o", label="Kallisto")
plot(zeros(len(y)), y, marker="o", markersize=10,
label="Jowisz", color="orange")

# opis rysunku
xlabel(u"pozycja względem Jowisza")
ylabel("czas od chwili obecnej\n(UT: %s)"%ts)
title(u"Ruch księżyców galileuszowych")
grid()
legend(loc=1)
savefig("KsiezyceJowisza.pdf")
show()
```

Ruch księżyców Jowisza – rysunek



Wyznaczanie zaćmienia Słońca – początek programu

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# IMPORTOWANIE
from pylab import *
import ephem as ep

# OBSERWATOR
obs = ep.city("Warsaw")

# TWORZYMY OBIEKTY
Slonce  = ep.Sun()
Ksiezyc = ep.Moon()

# krok czasowy - godzina
dt = ep.hour
# czas początkowy
ts = ep.now()
# czas aktualny
tm = ts
```

Wyznaczanie zaćmienia Słońca – główna pętla

Sprawdzamy separację Słońca i Księżyca co godzinę przez następne 10 lat.

```
for i in range(365*24*10):
    # ustawiamy aktualny czas
    obs.date = tm
    # obliczamy współrzędne
    Slonce.compute(obs)
    Ksiezyc.compute(obs)
    # promienie
    rs = Slonce.radius
    rk = Ksiezyc.radius
    # obliczamy odległość kątową
    sp = ep.separation(Slonce, Ksiezyc)
    # sprawdzamy czy suma promieni będzie
    # mniejsza od wyliczonej separacji
    if sp < rs+rk:
      # sprawdzamy czy Słońce będzie nad horyzontem
      if Slonce.alt > 0.0:
        print "UT: ", ep.Date(tm), "separacja:", sp
    # zwiększamy czas o godzinę
    tm += dt
```

Wyznaczanie zaćmienia Słońca – wyniki

Daty przewidzianego zaćmienia Słońca widocznego z Warszawy w ciągu następnych 10 lat. Należy pamiętać, że czas UT jest zapóźniony w stosunku do czasu lokalnego o godzinę w zimie i o dwie godziny w lecie.

```
UT:  2015/3/20 09:35:55 separacja: 0:13:26.1
UT:  2015/3/20 10:35:55 separacja: 0:20:03.9
UT:  2021/6/10 10:35:55 separacja: 0:25:13.0
UT:  2021/6/10 11:35:55 separacja: 0:27:46.1
UT:  2022/10/25 09:35:55 separacja: 0:24:30.4
UT:  2022/10/25 10:35:55 separacja: 0:16:09.3
```

Przebieg zaćmienia – początek programu

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# IMPORTOWANIE
from pylab import *
import ephem as ep

# OBSERWATOR
obs = ep.city("Warsaw")

# TWORZYMY OBIEKTY
Slonce  = ep.Sun()
Ksiezyc = ep.Moon()

# krok czasowy - 20 minut
dt = ep.hour/3.0
# czas początkowy
ts = ep.Date("2015/3/20 09:00:00")
#ts = ep.Date("2021/6/10 10:30:00")
#ts = ep.Date("2022/10/25 09:30:00")
obs.date = ts
```

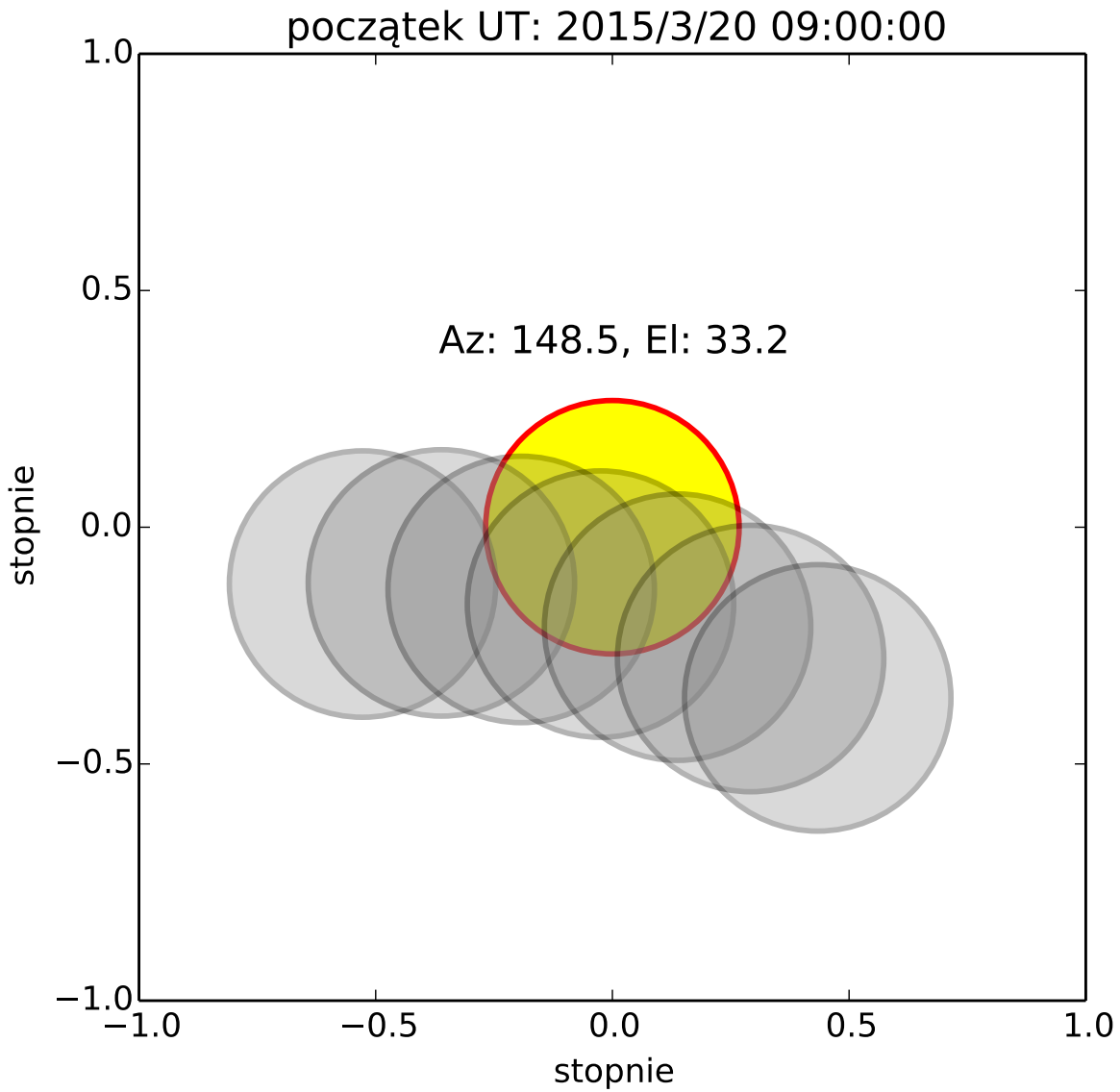
Przebieg zaćmienia – początkowe wyliczenia

```
# obliczamy współrzędne
Slonce.compute(obs)
Ksiezyc.compute(obs)
rs = degrees(Slonce.radius)
rk = degrees(Ksiezyc.radius)

# tworzymy wykres przypisując go do zmiennej pl
pl = subplot(111, aspect="equal")
# tytuł wykresu z datą i czasem
pl.set_title(u"początek UT: "+str(ep.Date(ts)))
# rysujemy Słońce w centrum
sc = Circle((0,0), rs, facecolor="yellow",
edgecolor="red", lw=2)
pl.add_artist(sc)
# współrzędne Słońca
pl.text(0, rs+0.1, "Az: %.1f, El: %.1f" %
(degrees(Slonce.az), degrees(Slonce.alt)),
ha='center', fontsize=14)
```

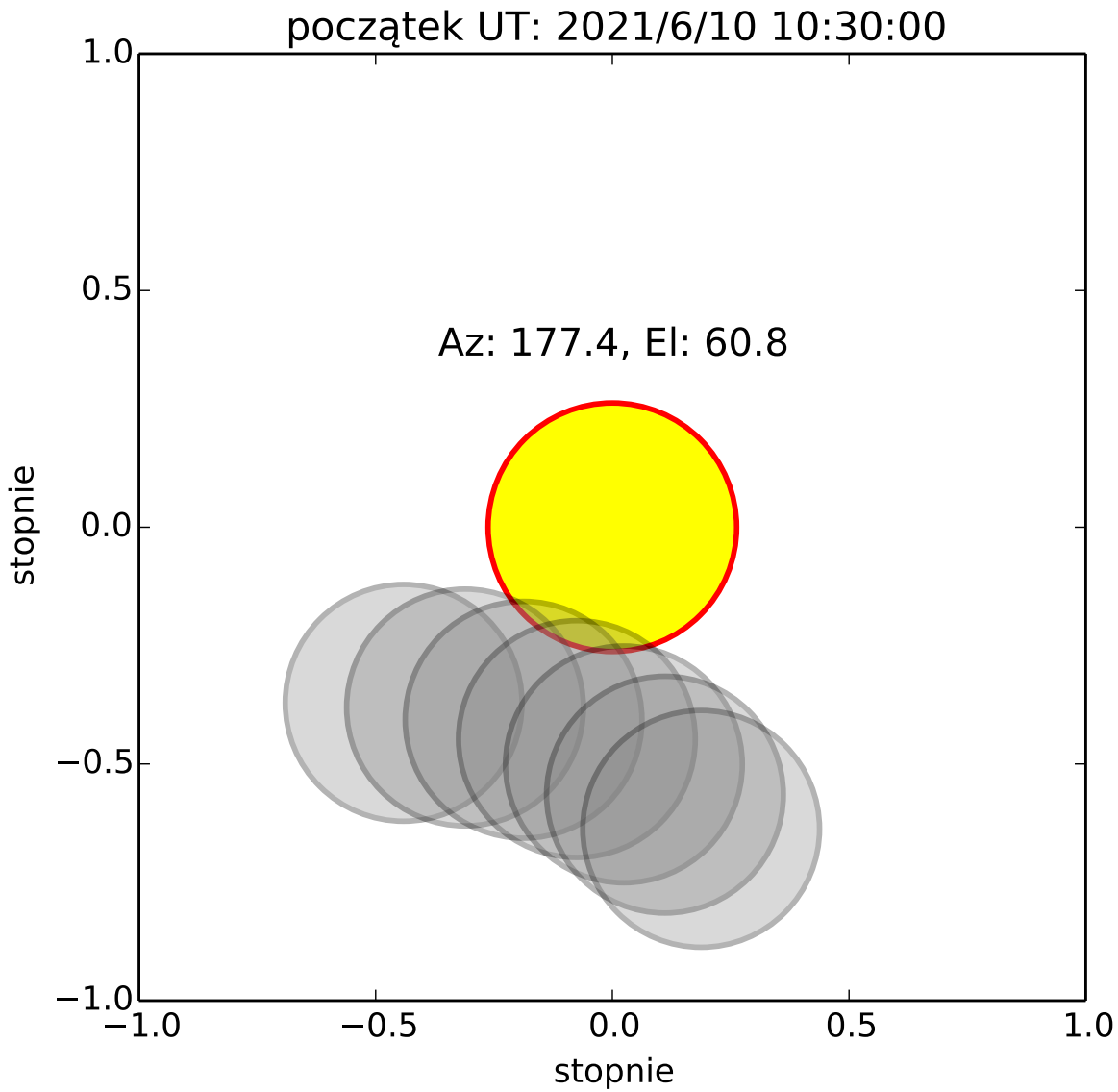
Przebieg zaćmienia – główna pętla

```
# rysujemy kolejne pozycje Księżyca
for i in range(7):
    print "czas UT: ", ep.Date(ts)
    obs.date = ts
    # obliczamy współrzędne
    Slonce.compute(obs)
    Ksiezyc.compute(obs)
    # obliczamy różnicę w pozycji
    az = degrees(Slonce.az - Ksiezyc.az)
    el = degrees(Slonce.alt - Ksiezyc.alt)
    # rysujemy aktualną pozycję Księżyca względem Słońca
    kc = Circle((az, el), rk, facecolor="gray",
    edgecolor="black", lw=2, alpha=0.3)
    pl.add_artist(kc)
    # zwiększamy czas o 20 minut
    ts += dt

pl.set_xlim(-1.0, 1.0)
pl.set_ylim(-1.0, 1.0)
pl.set_xlabel("stopnie")
pl.set_ylabel("stopnie")
savefig("Zacmienie.pdf")
show()
```

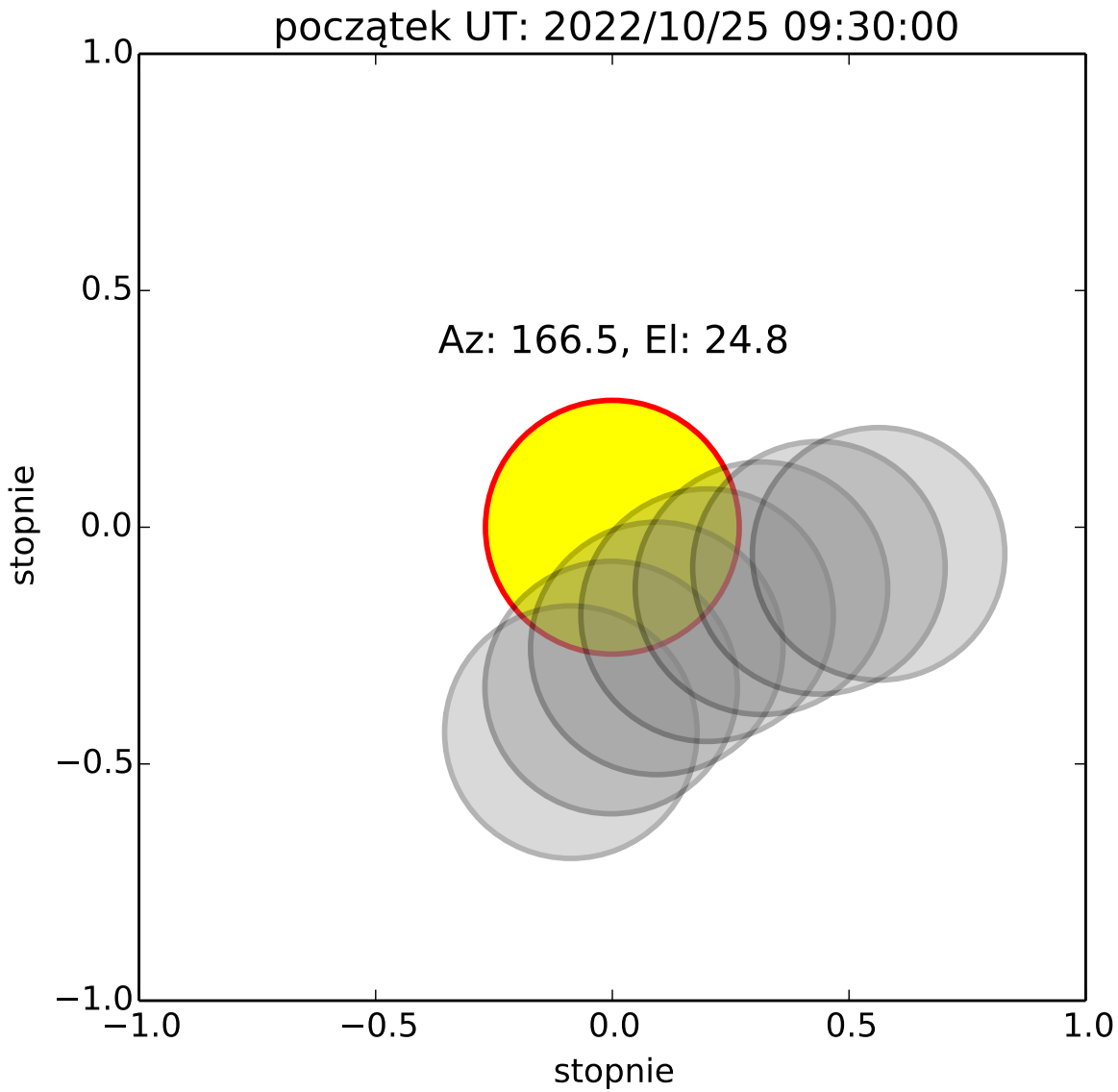
Przebieg zaćmienia – rysunek



Zaćmienie w czerwcu 2021 roku



Zaćmienie w październiku 2022 roku



Najbliższe całkowite zaćmienie

W programie `pyephem7.py` podajemy Oslo zamiast Warszawy

```
# OBSERWATOR
obs = ep.city("Oslo")
```

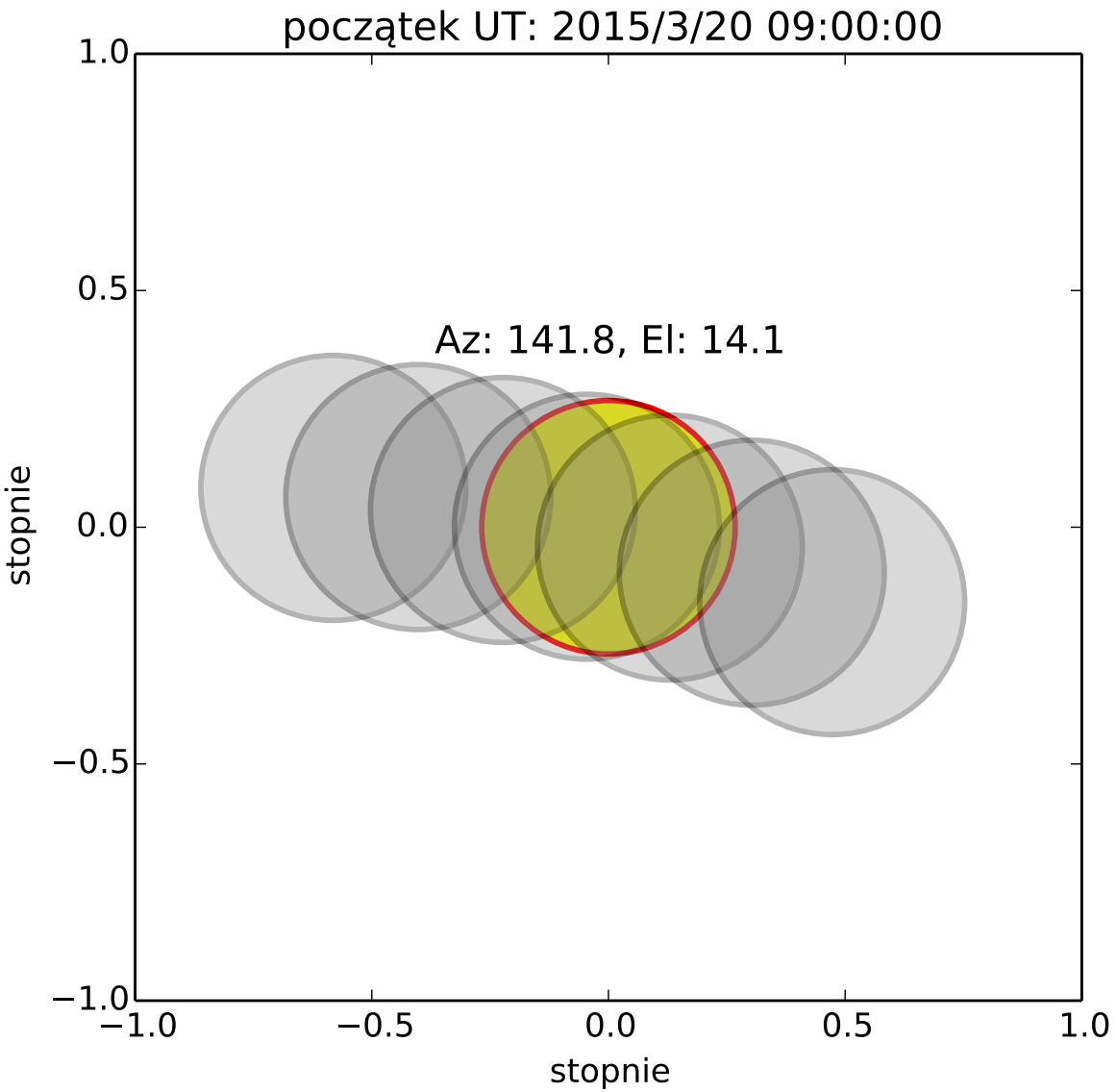
albo „wybieramy” się na jeszcze lepszą pozycję, na Morze Północne

```
obs = ep.Observer()
obs.lon = "10.0"
obs.lat = "72.0"
```

podobnie zmieniamy pozycję obserwatora w programie `pyephem8.py`, w którym ustawiamy też nową datę

```
# czas początkowy
ts = ep.Date("2015/3/20 09:00:00")
```

Całkowite zaćmienie widoczne z Morza Północnego



Koniunkcja Księżyca i planet

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# IMPORTOWANIE
from pylab import *
import ephem as ep

# OBSERWATOR
obs = ep.city("Warsaw")

# TWORZYMY OBIEKTY
Slonce  = ep.Sun() # sprawdzamy czy jest pod horyzontem
Ksiezyc = ep.Moon()
Wenus   = ep.Venus()
Mars    = ep.Mars()
Jowisz  = ep.Jupiter()

# krok czasowy - godzina
dt = ep.hour
# czas początkowy
ts = ep.now()
# czas aktualny
tm = ts
```

9 Koniunkcja Księżyca i planet – główna pętla

```
# GŁÓWNA PĘTLA PROGRAMU
for i in range(365*24*2):
    # ustawiamy aktualny czas
    obs.date = tm
    # obliczamy współrzędne
    Slonce.compute(obs)
    Ksiezyc.compute(obs)
    Wenus.compute(obs)
    Mars.compute(obs)
    Jowisz.compute(obs)
    # obliczamy separację
    s1 = ep.separation(Wenus, Ksiezyc)
    s2 = ep.separation(Mars, Ksiezyc)
    s3 = ep.separation(Jowisz, Ksiezyc)
    # separacja ma byc mniejsza jak 5 stopni
    if degrees(s1) < 5:
      # sprawdzamy czy Księżyc będzie nad
      # horyzontem a Słońce pod horyzontem
      if degrees(Ksiezyc.alt) > 5.0 and degrees(Slonce.alt) < -5.0:
        print "-----------------------------------------------------"
        print u"poprzedni nów, UT:", ep.previous_new_moon(ep.Date(tm))
        print u"Wenus-Księżyc, UT:", ep.Date(tm), "separacja:", s1
        print u"poz. Księżyca, Az:", Ksiezyc.az, "El:", Ksiezyc.alt
```

9 Koniunkcja Księżyca i planet – główna pętla, cd.

```
    if degrees(s2) < 5:
      if degrees(Ksiezyc.alt) > 5.0 and degrees(Slonce.alt) < -5.0:
        print "-----------------------------------------------------"
        print u"poprzedni nów, UT:", ep.previous_new_moon(ep.Date(tm))
        print u"Mars-Księżyc,  UT:", ep.Date(tm), "separacja:", s2
        print u"poz. Księżyca, Az:", Ksiezyc.az, "El:", Ksiezyc.alt
    if degrees(s3) < 5:
      if degrees(Ksiezyc.alt) > 5.0 and degrees(Slonce.alt) < -5.0:
        print "-----------------------------------------------------"
        print u"poprzedni nów, UT:", ep.previous_new_moon(ep.Date(tm))
        print u"Jowisz-Księżyc,UT:", ep.Date(tm), "separacja:", s3
        print u"poz. Księżyca, Az:", Ksiezyc.az, "El:", Ksiezyc.alt
    # zwiększamy czas o godzinę
    tm += dt
```

Przykładowe rezultaty wyliczeń:

```
-----------------------------------------------------
poprzedni nów, UT: 2015/9/13 06:41:15
Jowisz-Księżyc,UT: 2015/10/10 03:32:01 separacja: 4:39:14.6
poz. Księżyca, Az: 100:15:36.6 El: 12:31:04.8
-----------------------------------------------------
poprzedni nów, UT: 2015/10/13 00:05:43
Wenus-Księżyc, UT: 2015/11/7 02:32:01 separacja: 4:19:58.5
poz. Księżyca, Az: 105:40:27.4 El: 13:24:08.9
-----------------------------------------------------
poprzedni nów, UT: 2015/10/13 00:05:43
Mars-Księżyc,  UT: 2015/11/7 02:32:01 separacja: 3:08:06.4
poz. Księżyca, Az: 105:40:27.4 El: 13:24:08.9
-----------------------------------------------------
```

10 Mapa nieba – początek programu

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# IMPORTOWANIE
from pylab import *
import ephem as ep
from ephem import stars # KATALOG JASNYCH GWIAZD
# OBSERWATOR
obs = ep.city("Warsaw")
# KOLORY PLANET
ko = ["yellow", "grey", "brown", "cyan", "red",
"orange", "orange", "cyan", "blue"]
# LISTA OBIEKTÓW
ob = []
ob.append(ep.Sun())
ob.append(ep.Moon())
ob.append(ep.Mercury())
ob.append(ep.Venus())
ob.append(ep.Mars())
ob.append(ep.Jupiter())
ob.append(ep.Saturn())
ob.append(ep.Uranus())
ob.append(ep.Neptune())
```

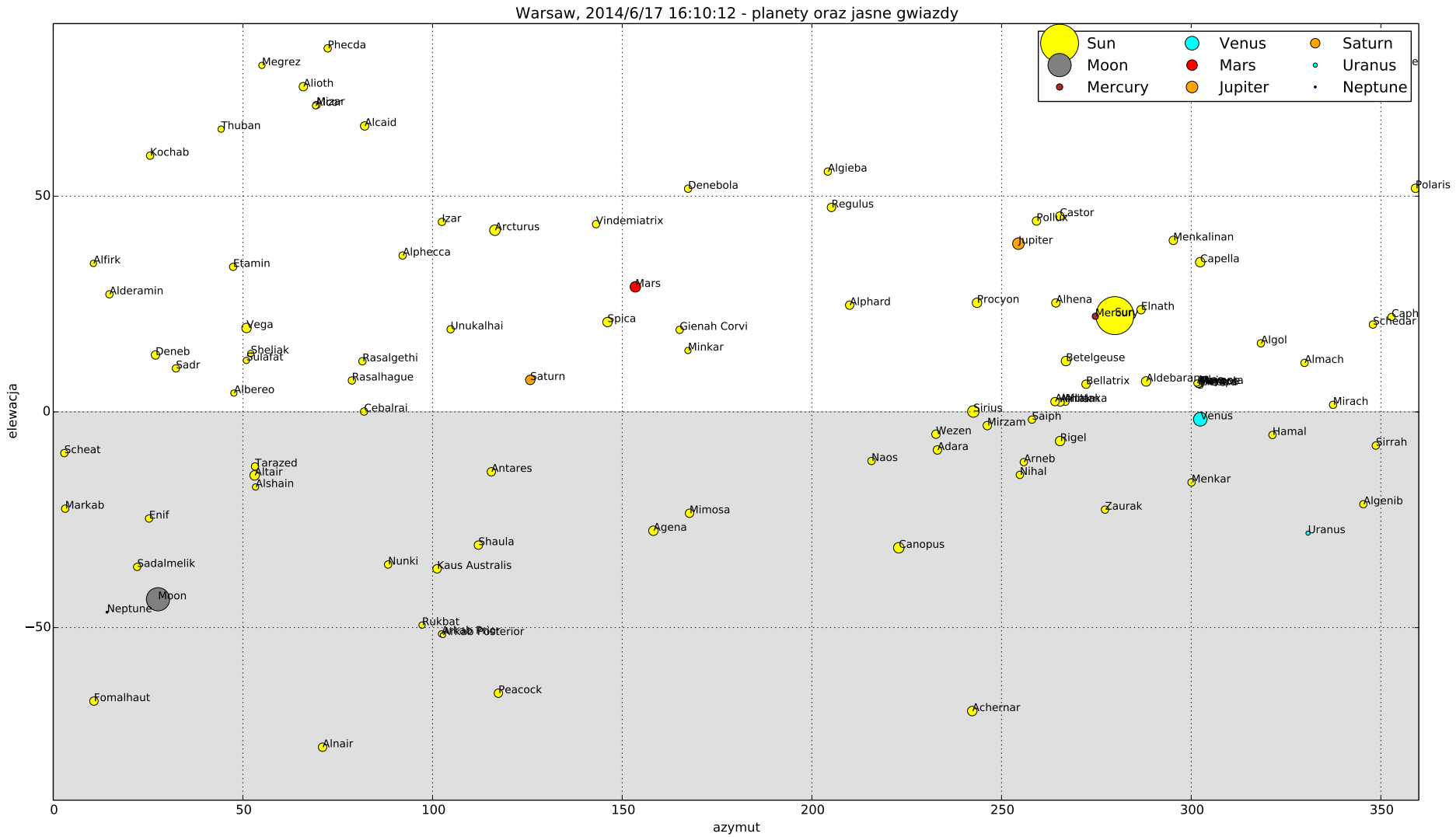
10 Mapa nieba – główna pętla

```
# DODAJEMY KATALOG JASNYCH GWIAZD
for x in stars.db.split("\n"):
    if x!="": ob.append(ep.readdb(x))

# OBLICZANIE WSPÓŁRZĘDNYCH
for x in ob: x.compute(obs)

# GŁÓWNA PĘTLA PROGRAMU
i = 0
for x in ob:
    # obliczamy wsp. azymutalne
    az = degrees(x.az)
    el = degrees(x.alt)
    # kolory dla planet
    if i <= 8:
      plot([az], [el], markersize=int(-x.mag+10),
      ls="", marker="o",
      color=ko[i], label=x.name)
    else:
      plot([az], [el], markersize=int(-x.mag+10), ls="",
      marker="o", color="yellow")
    # opisy obiektów
    text(az, el, x.name, fontsize=10)
    i += 1
```

Mapa nieba – parametry wykresu

```
# szary cień pod horyzontem
xd = [0, 360]
y1 = [-90, -90]
y2 = [0, 0]
fill_between(xd, y1, y2, color="black",
facecolor="grey", alpha=0.25)

# opis rysunku
xlabel("azymut")
ylabel("elewacja")
title("%s, %s - planety oraz jasne gwiazdy" %
      (obs.name, ep.now()))
legend(ncol=3, numpoints=1)
xlim(0, 360)
ylim(-90, 90)
grid()
show()
```

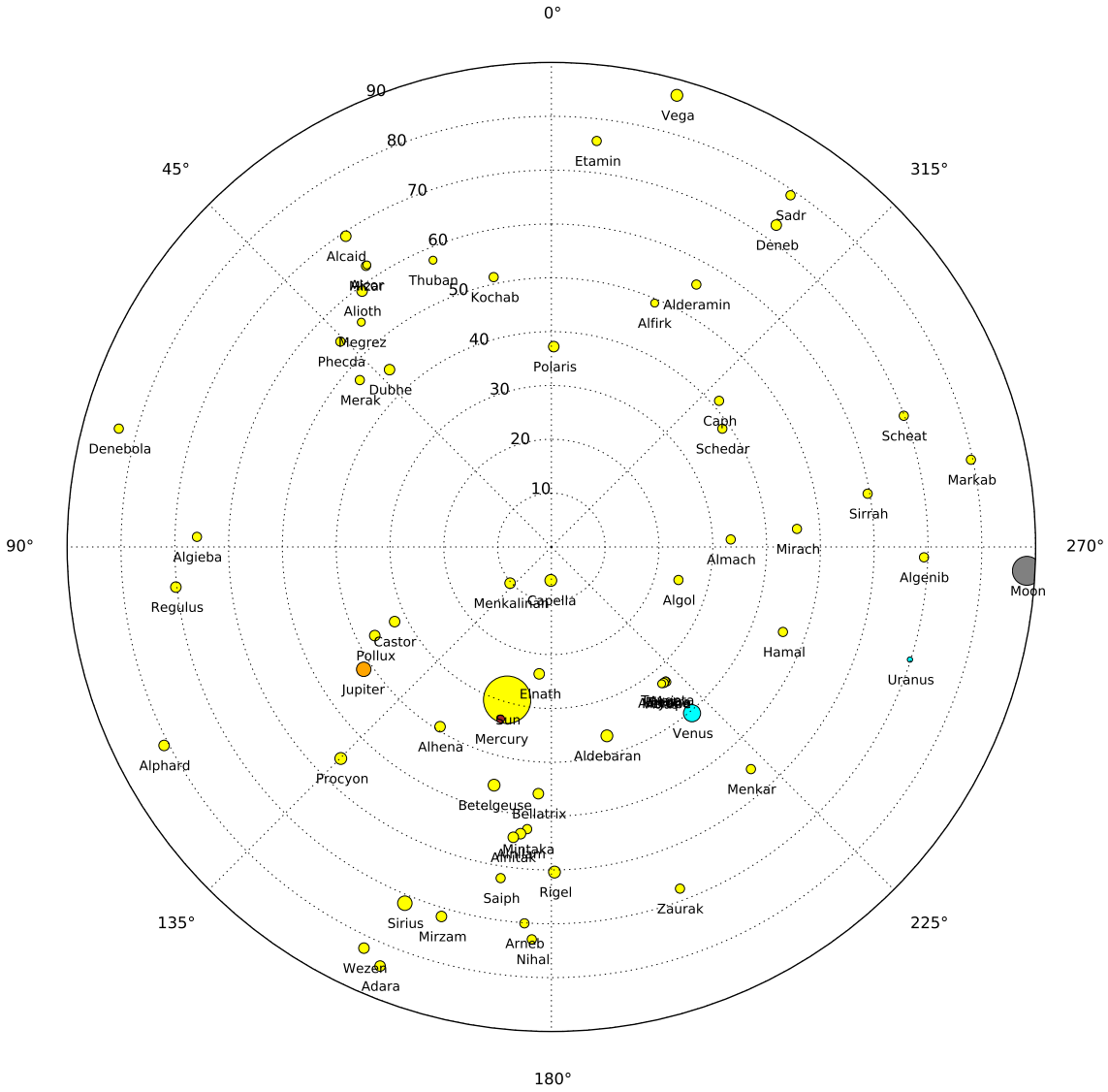
Mapa nieba


Warsaw, 2014/6/17 16:10:12 - planety oraz jasne gwiazdy
Sun
Moon
Mercury
Venus
Mars
Jupiter
Saturn
Uranus
Neptune
Phecda
Megrez
Alioth
Mizar
Alcaid
Thuban
Kochab
Algieba
Denebola
Regulus
Polaris
Izar
Arcturus
Vindemiatrix
Castor
Pollux
Jupiter
Menkalinan
Capella
Alphecca
Alfirk
Etamin
Alderamin
Mars
Alphard
Procyon
Alhena
Elnath
Mercury
Sun
Caph
Schedar
Vega
Unukalhai
Spica
Gienah Corvi
Minkar
Algol
Deneb
Sheliak
Sulafat
Sadr
Rasalgethi
Rasalhague
Saturn
Betelgeuse
Almach
Bellatrix
Aldebaran
Albereo
Cebalrai
Sirius
Mirach
Saiph
Mirzam
Venus
Wezen
Hamal
Scheat
Rigel
Adara
Sirrah
Tarazed
Altair
Alshain
Antares
Naos
Arneb
Nihal
Menkar
Markab
Enif
Mimosa
Agena
Zaurak
Algenib
Shaula
Canopus
Uranus
Sadalmelik
Nunki
Kaus Australis
Moon
Neptune
Rukbat
Peacock
Fomalhaut
Achernar
Alnair
elewacja
azymut
50
0
−50
0
50
100
150
200
250
300
350

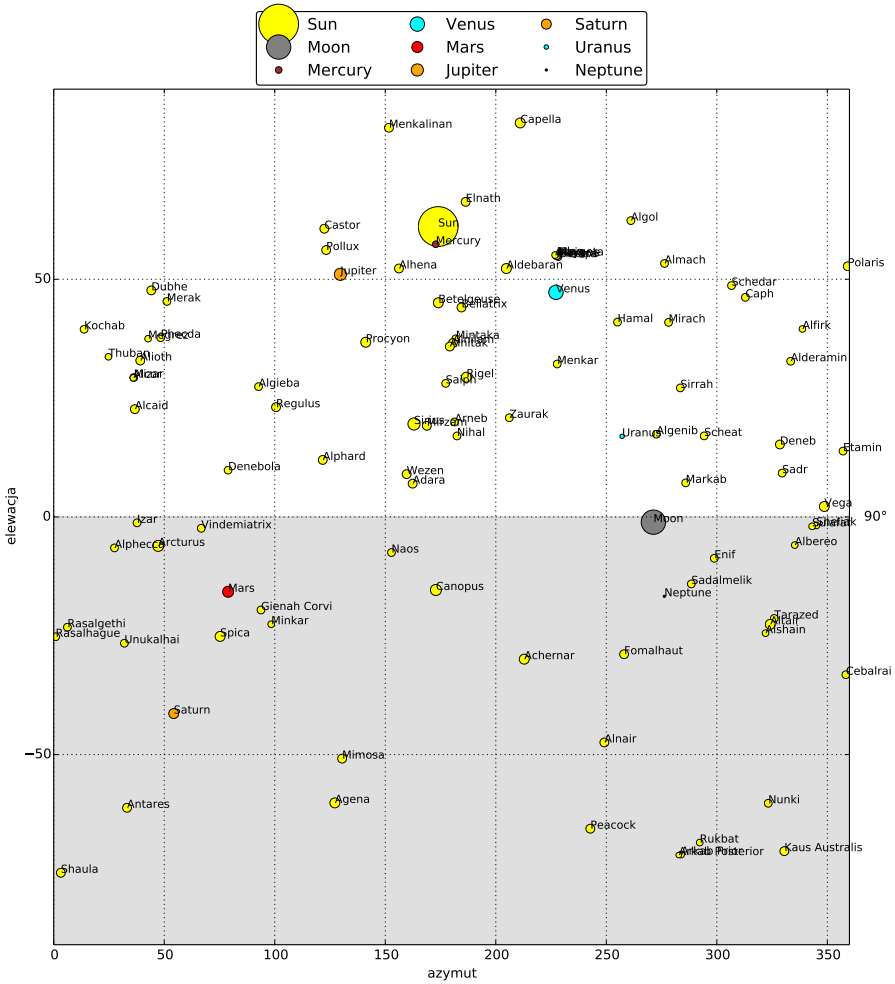
Mapa nieba – współrzędne biegunowe

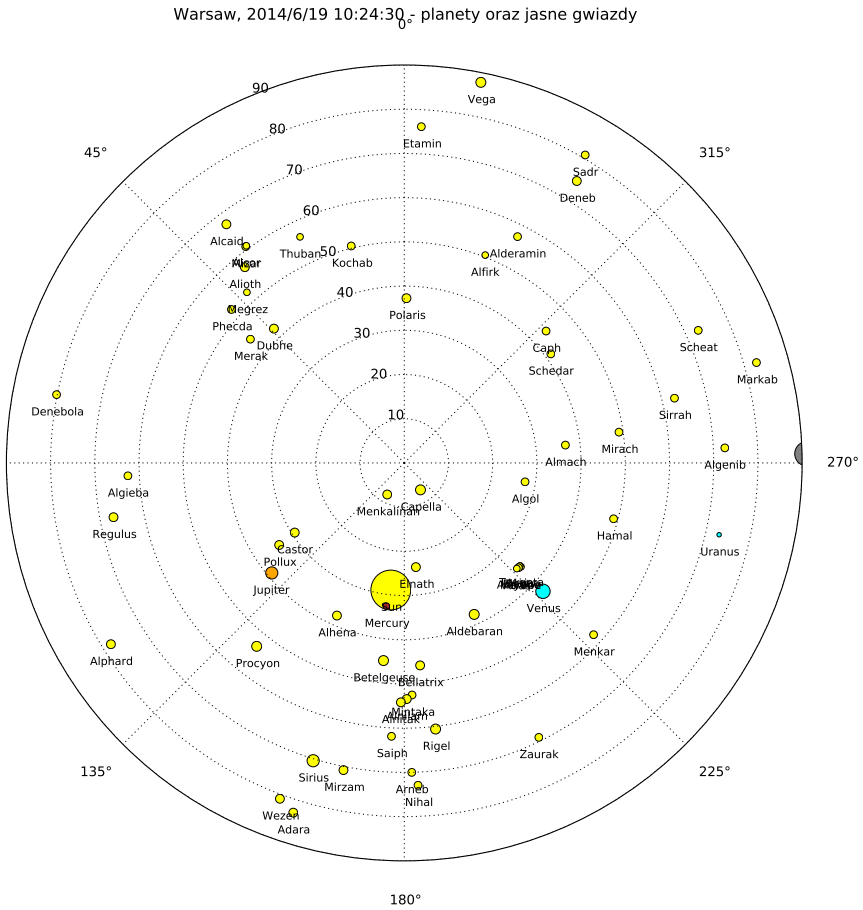
```
# WSPÓŁRZĘDNE BIEGUNOWE
subplot(111, polar=True)
# GŁÓWNA PĘTLA PROGRAMU
i = 0
for x in ob:
    # obliczamy wsp. azymutalne
    az = x.az # AZYMUT PODAJEMY W RADIANACH!
    el = degrees(x.alt)
    # kolory dla planet
    if i <= 8:
      plot([az], [90-el], markersize=int(-x.mag+10),
      ls="", marker="o", color=ko[i])
    else:
      plot([az], [90-el], markersize=int(-x.mag+10), ls="",
      marker="o", color="yellow")
    # opisy obiektów
    if el > 0:
      text(az, 90-el, "\n"+x.name, fontsize=10,
      ha='center',  va='top')
    i += 1
# ograniczamy odległość zenitalną do 90 stopni
gca().set_rmax(90.0)
# ustawiamy północ na górze wykresu
gca().set_theta_zero_location("N")
```

Mapa nieba – współrzędne biegunowe



Dwie mapy nieba razem





Wczytywanie katalogów

Moduł PyEphem pozwala wczytywać katalogi rożnych obiektów sporządzone jako pliki tekstowe, w formacie przyjętym dla programu Xephem (www.clearskyinstitute.com/xephem/help/xephem.html)

```
M42|NGC 1976|LBN 974|Orion nebula,f|N|EN,5:35:17.1,-5:23:25,4,2000,3900
M43|NGC 1982|CED 55G,f|N|EN,5:35:31.3,-5:16:3,9,2000,1200
M44|NGC 2632|OCL 507|Praesepe,f|O|T2,8:39:57.2,19:40:21,3.1,2000,4200
M45,f|U,03:47:0,24:07,1.6,2000,6600
```

Jedna linia katalogu opisuje jeden obiekt dla którego podana jest nazwa lub kilka nazw, typ, rektascensja, deklinacja, jasność widoma, epoka oraz dodatkowy parametr zależny od typu obiektu, np. rozmiar w milisekundach łuku.

Źródła programu Xephem rozpowszechniane są z dwoma katalogami **Messier.edb** oraz **SKY2k65.edb**. Pierwszy z plików zawiera katalog obiektów Messier'a, natomiast drugi jest okrojoną (obiekty do 6.5 mag) wersją katalogu SKY2000 (Myers J.R. et al. 2002).

Wczytanie katalogu

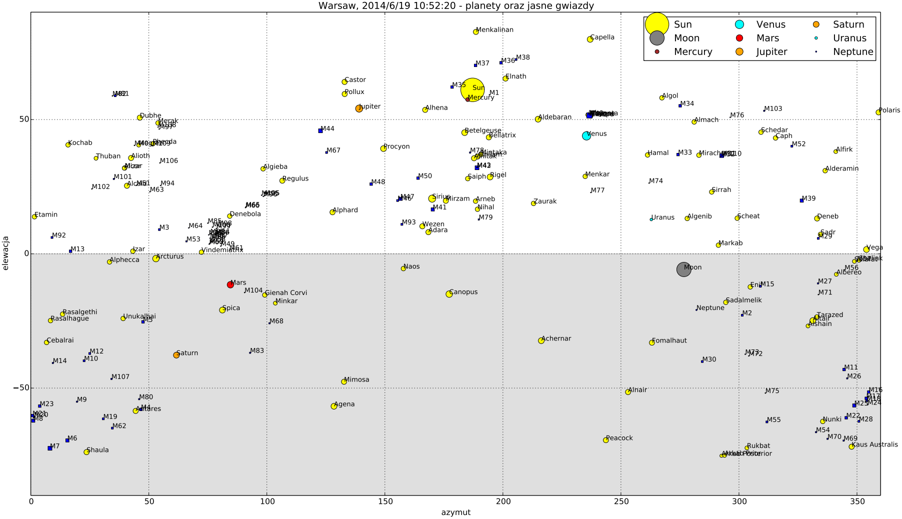
Wczytujemy katalog z pliku tekstowego **Messier.edb** i linia po linii dodajemy obiekty do naszej listy obiektów **ob**.

```
# WCZYTUJEMY KATALOG MESSIERA
plik = open("Messier.edb", "r")
for linia in plik:
    # ignorujemy linie komentarza
    # zaczynające się znakiem #
    if linia[0]!="#": ob.append(ep.readdb(linia))
plik.close()
```

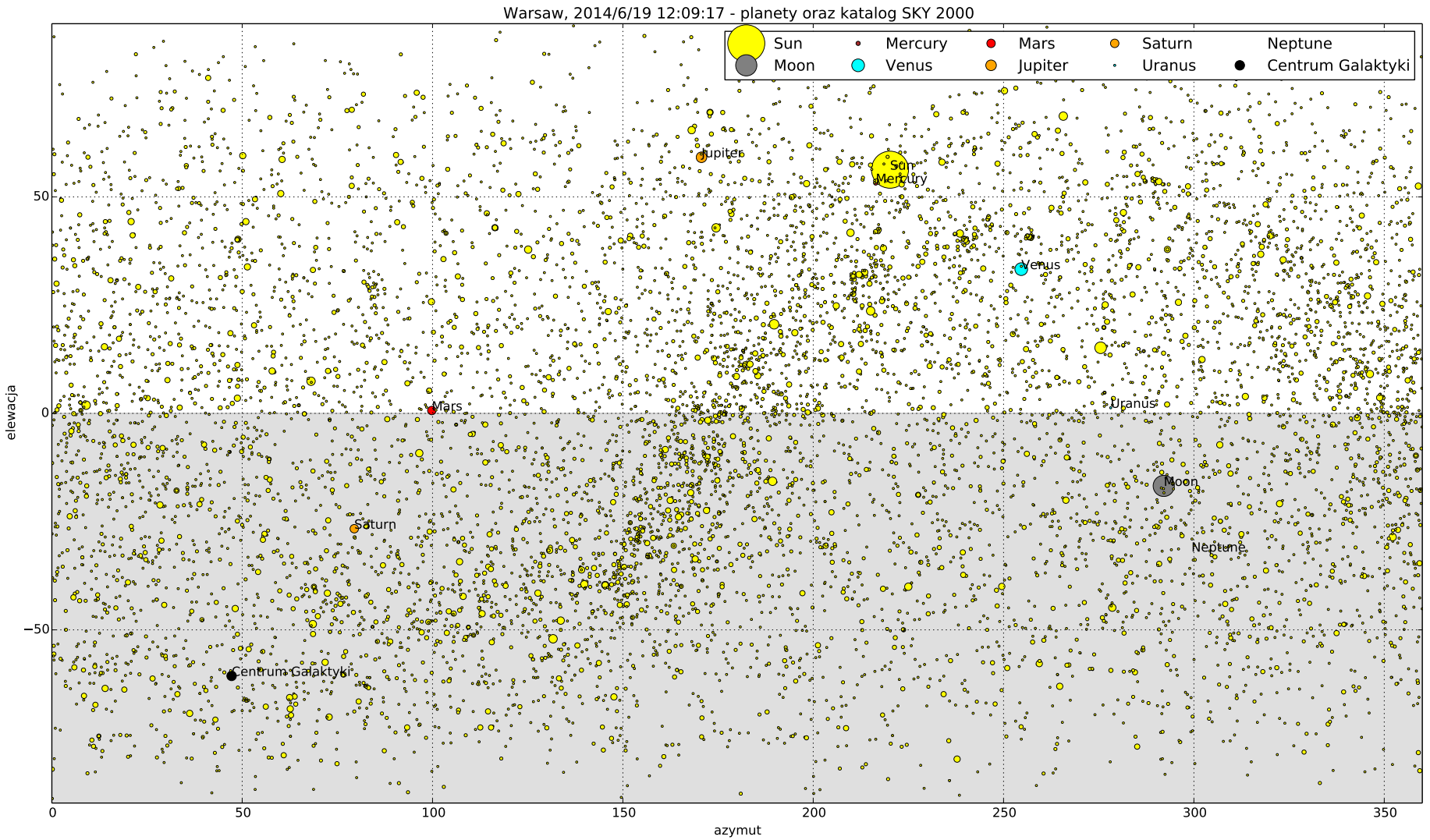
Możemy też dodać własny obiekt do listy obiektów.

```
# DODAJEMY WŁASNY OBIEKT
ob.append(ep.readdb("Centrum Galaktyki,f,17:45:40.04,-29:00:28.1,-1,2000,1200"))
```

13 Mapa nieba z obiektami Messier’a



Katalog SKY 2000



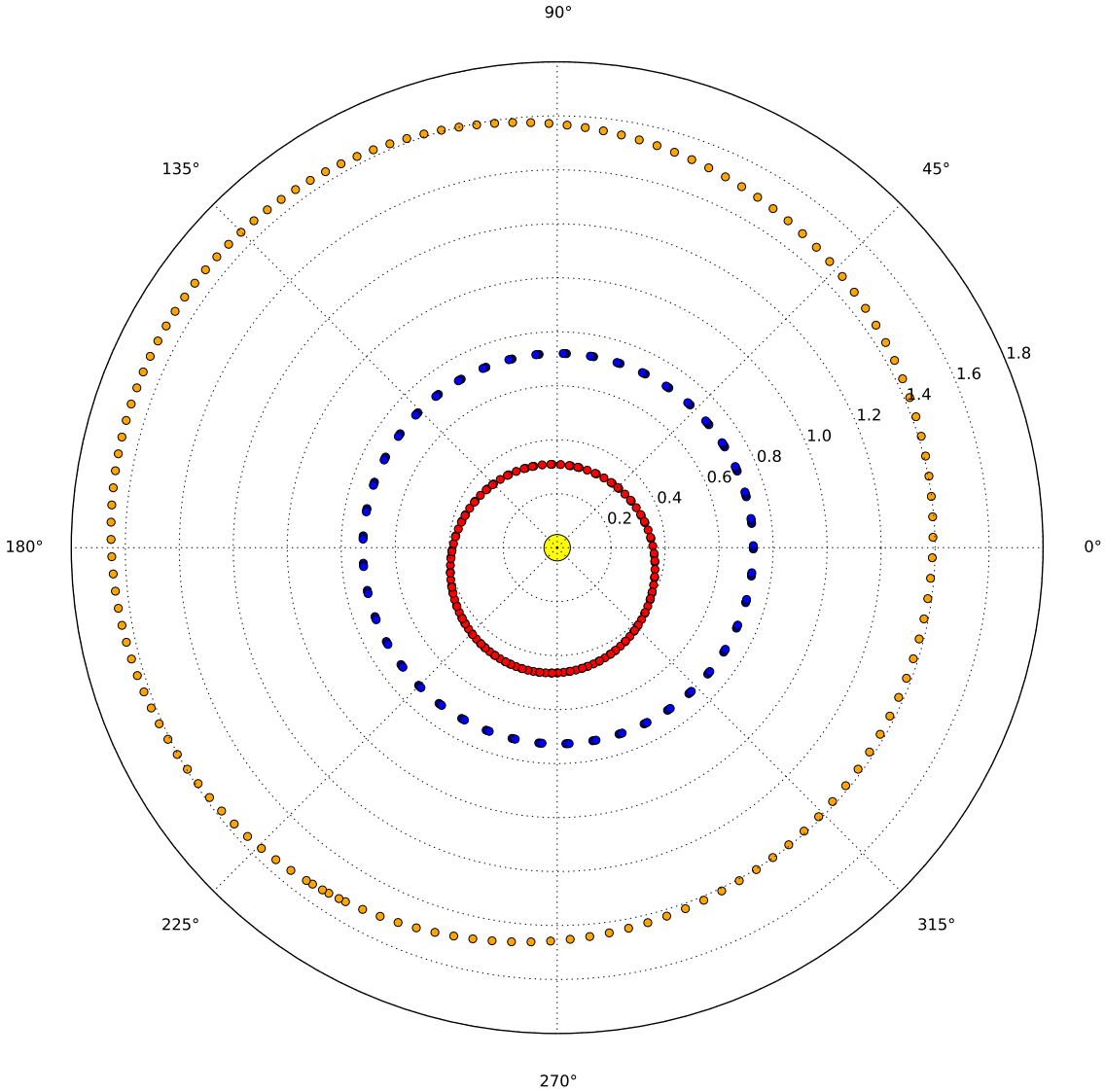
Układ słoneczny – początek programu

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# IMPORTOWANIE
from pylab import *
import ephem as ep

# OBIEKTY
Merkury = ep.Mercury()
Wenus   = ep.Venus()
Mars    = ep.Mars()

# CZAS
tm = ep.now() # początek
dt = ep.hour*24*5 # krok

# WSPÓŁRZĘDNE BIEGUNOWE
subplot(111, polar=True)

# SŁOŃCE
plot([0], [0], marker="o", color="yellow", markersize=20)
```

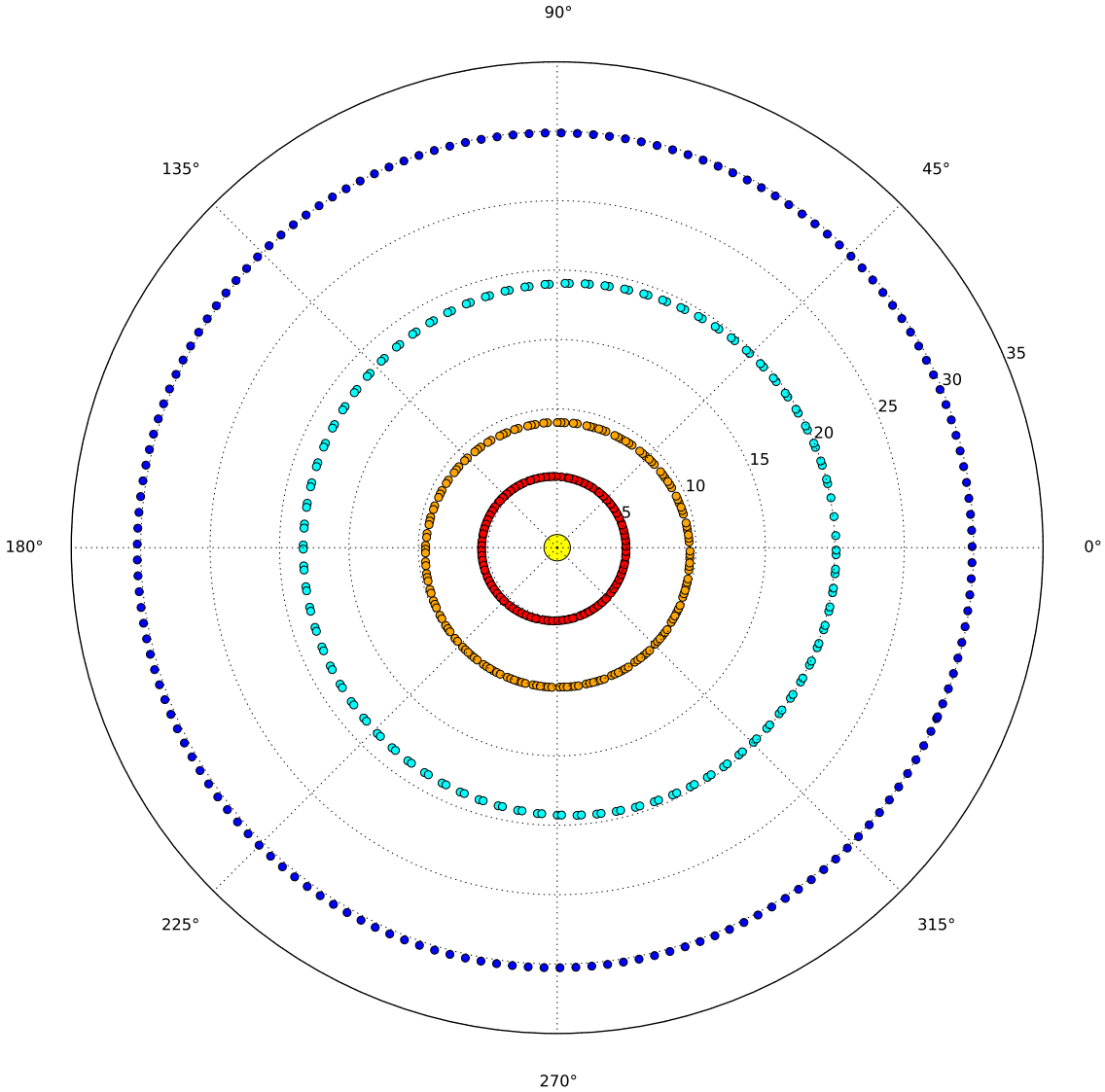
Układ słoneczny – główna pętla

```
# GŁÓWNA PĘTLA
for i in range(140):
    # obliczenia
    Merkury.compute(tm)
    Wenus.compute(tm)
    Mars.compute(tm)
    # punkty na wykresie
    plot([Merkury.hlong], [Merkury.sun_distance],
         ls="", marker="o", c="red")
    plot([Wenus.hlong], [Wenus.sun_distance],
         ls="", marker="o", c="blue")
    plot([Mars.hlong], [Mars.sun_distance],
         ls="", marker="o", c="orange")
    # zwiększamy krok czasowy
    tm += dt

show()
```

Planety wewnętrzne



Planety zewnętrzne



Lista programów

1. ./python/pyephem1.py
2. ./python/pyephem2.py
3. ./python/pyephem3.py
4. ./python/pyephem4.py
5. ./python/pyephem5.py
6. ./python/pyephem6.py
7. ./python/pyephem7.py
8. ./python/pyephem8.py
9. ./python/pyephem9.py
10. ./python/pyephem10.py
11. ./python/pyephem11.py
12. ./python/pyephem12.py
13. ./python/pyephem13.py
    - ./python/Messier.edb
14. ./python/pyephem14.py
    - ./python/SKY2k65.edb
15. ./python/pyephem15.py
16. ./python/pyephem16.py

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

WOJEWÓDZTWO
KUJAWSKO-POMORSKIE

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

KUJAWSKO-POMORSKIE
CENTRUM EDUKACJI
NAUCZYCIELI
W TORUNIU

KPCEN
TORUŃ

PAŃSTWOWA
WYŻSZA
SZKOŁA
ZAWODOWA
WE WŁOCŁAWKU