Python scripting language a obliczenia i symulacje

Marcin Owsiany
porridge@debian.org
`http://student.uci.agh.edu.pl/~porridge/studia.pl.html`

czerwiec 2001

1 Charakterystyka języka i podstawowych pakietów

1.1 Kod

Python jest językiem interpretowanym, ale przy uruchomieniu kod źródłowy zawsze jest kompilowany do *byte-code*-u.
Po skompilowaniu python próbuje zapisać *byte-code* do pliku z rozszerzeniem `.pyc`. Jeśli mu się to uda, to przy następnym uruchomieniu skryptu ładowany jest gotowy *byte-code*, dzięki czemu unika się ponownego parsingu.
Python przyjmuje flagi wymuszające optymalizację, ale nie ma ona wpływu na szybkość działania kodu, a jedynie na czas ładowania do pamięci.
Python udostępnia też deterministyczny profiler (zawarty w moduły profile i pstats). Zawiera on dodatkowo narzędzia do generowania raportów na temat rezultatów operacji profilowania.

1.2 Składnia Pythona

Niżej wymieniłem niektóre cechy składni Pythona, dzięki którym kod aplikacji może być bardziej zwięzły niż w innych językach. Cechy te nie mają rzecz jasna wpływu na szybkość obliczeń, ale niejednokrotnie ułatwiają pisanie bardziej zwartego kodu:

- Równoczesne podstawienia.

```
>>> a, b = 1, 2
>>> a, b = b, a+b
>>> a
2
>>> b
3
>>>
```

- Łączenie operacji porównania w łańcuchy. Na przykład `a < b == c` jest tym samym, co `a < b and b == c`, z tym wyjątkiem, że w pierwszym przypadku wyrażenie `b` jest obliczane tylko raz.
- Obsługa operacji symbolicznych. Pozwala to niekiedy uniknąć iteracji na rzecz zwięzłego wyrażenia.

A oto inne właściwości Pythona, o których trzeba pamiętać podczas obliczeń.

- Operatory `and` i `or` są "leniwe", podobnie jak w C. Należy o tym pamiętać na przykład przy pisaniu wyrażeń wykorzystujących efekty uboczne funkcji.
- Przypisanie nie może nastąpić w wyrażeniu, inaczej niż ma to miejsce w języku C.

1.3 Właściwości numeryczne języka

1.3.1 Porównywanie

Obiekty różnych typów (poza typami numerycznymi) nigdy nie mogą być równe. Obiekty tej samej klasy zazwyczaj są domyślnie klasyfikowane jako różne, chyba że klasa definiuje specjalną metodę `__cmp__()`, która umożliwia porównywanie.

1.3.2 Typy numeryczne

Istnieją cztery typy numeryczne:

Zwykłe liczby całkowite (plain integers). Zwane zazwyczaj po prostu liczbami całkowitymi, są zaimplementowane przy pomocy typu `long` w języku C, dzięki czemu mają conajmniej 32 bity precyzji.

Długie liczby całkowite (long integers). Mają nieograniczoną precyzję.

Liczby zmiennoprzecinkowe (floating point numbers). Są zaimplementowane przy pomocy typu `double` w języku C. Ich precyzja zależy od konkretnej maszyny, na której jest zainstalowany Python.

Liczby zespolone (complex numbers). Posiadają część rzeczywistą i urojoną, z których każda zaimplementowana jest przy pomocy typu `double` w języku C.

Liczby tworzy się używając literałów liczbowych lub przy pomocy wbudowanych funkcji i operatorów. Czyste literały całkowite (w tym liczby ósemkowe i szestnastkowe) powodują utworzenie zwykłych liczb całkowitych (plain integers). Literały całkowite z przyrostkiem `L` lub `l` powodują utworzenie długich liczb całkowitych. Literały numeryczne zawierające kropkę dziesiętną lub symbol eksponenty powodują utworzenie liczby zmiennoprzecinkowej.
Część urojoną liczby zespolonej zapisuje się używając przyrostka "j" lub "J". Całą liczbę zespoloną z o częściach: rzeczywistej: x i urojonej: y zapisuje się $z = x + yj$ (lub $z = x + yJ$), o ile x i y są literałami, lub tworzy przy pomocy funkcji `complex(x, y)`. Argumentami funkcji `complex()` mogą być zmienne dowolnego typu numerycznego (w tym typu zespolonego).
Aby pobrać wartości części rzeczywistej i urojonej liczby zespolonej `z` należy użyć odpowiednio konstrukcji `z.real` i `z.imag`.
Python w pełni obsługuje arytmetykę mieszaną: gdy operator dwuargumentowy ma operandy dwóch różnych typów, przed wykonaniem operacji operand "mniejszego" typu jest konwertowany na ten drugi typ (gdzie podana wyżej kolejność jest kolejnością od "najmniejszego" do "największego"). Takiej samej reguły używa się przy wykonywaniu porównań. (Dzięki temu lista [1, 2] jest równa [1.0, 2.0] itd.) Aby wymusić konwersję można wykorzystać wbudowane funkcje `int()`, `long()`, `float()` i `complex()`.
Funkcje konwersji na liczbę zmiennoprzecinkową i całkowitą (`float()`, `int()` i `long()`) nie działają dla liczb zespolonych, ponieważ po prostu nie ma jednoznacznego sposobu przekształcenia liczby zespolonej na rzeczywistą. Aby otrzymać (jako liczbę zmiennoprzecinkową) moduł liczby zespolonej `z` należy użyć konstrukcji `abs(z)`, a `z.real` – aby otrzymać część rzeczywistą.
Wszystkie typy numeryczne obsługują następujące operacje (podane w rosnącej kolejności priorytetów):

1.3.3 Wyjątki związane z obliczeniami

ArithmeticError Klasa podstawowa dla wszystkich wyjątków związanych z obliczeniami.

FloatingPointError Występuje w przypadku błędu w operacji zmiennoprzecinkowej. Jest zawsze zdefiniowany, ale może być zgłoszony tylko jeśli Python został skonfigurowany z opcją –with-fpectl, lub gdy w pliku config.h jest zdefiniowany symbol `WANT_SIGFPE_HANDLER`.

OverflowError Zgłaszany gdy rezultat operacji arytmetycznej nie mieści się w reprezentacji. Nie może się to wydażyć w przypadku operacji na długich liczbach całkowitych (które prędzej zgłoszą `MemoryError`, niż się poddadzą). Z powodu braku standaryzacji sprawdzania wyjątków związanych z arytmetyką zmiennoprzecinkową w C, nie jest sprawdzana również większość operacji zmiennoprzecinkowych. W przypadku zwykłych liczb całkowitych sprawdzane są wszystkie operacje mogące wywołać przepełnienie, oprócz przesunięcia w lewo, ponieważ w większości jego zastosowań zamiast zgłoszenia wyjątku woli się utratę bitów.

ZeroDivisionError Zgłaszany w przypadku gdy drugi operand dzielenia lub operacji modulo jest równy 0. Z tym wyjątkiem związany jest łańcuch określający typy operandów i rodzaj operacji.

1.4 Moduł `math`

Zapewnia on dostęp do podstawowych funkcji matematycznych określonych w standardzie C. Nie obsługuje on jednak liczb zespolonych (obsługę tych liczb można znaleźć w module cmath). Zawiera on następujące funkcje:

- `acos()`, `asin()`, `atan()`, `atan2(y, x)` (`atan(y / x)`)

- `cos()`, `cosh()`, `sin()`, `sinh()`, `tan()`, `tanh()`
- `exp()`
- `fabs(x)` (bezwzględna wartość liczby zmiennoprzecinkowej `x`)
- `ceil(x)` (sufit(`x`) jako liczba zmiennoprzecinkowa), `floor()` (podłoga(`x`) jako liczba zmiennoprzecinkowa)
- `fmod(x, y)` (zwraca `fmod(x, y)`, tak jak jest zdefiniowany przez bibliotekę C danej platformy. Uwaga: wyrażenie Pythona `x % y` może zwrócić inną wartość)
- `frexp(x)` (zwraca mantysę i eksponent liczby `x` jako parę `(m, e)`, gdzie `m` jest liczbą zmiennoprzecinkową, a `e` jest liczbą całkowitą, taką że $x == m2^e$. Jeśli `x` jest zerem, zwraca `(0.0, 0)`, w przeciwnym wypadku $0.5 <= abs(m) < 1$)
- `hypot(x, y)` (odległość od środka euklidesowego układu współrzędnych, czyli $\sqrt{x^2 + y^2}$)
- `ldexp(x, i)` ($x2^i$)
- `log(x)` (logarytm naturalny `x`), `log10()`
- `modf(x)` (zwraca ułamkową i całkowitą część `x`. Obie części mają taki znak, jak `x`. Część ułamkowa jest zwracana jako liczba zmiennoprzecinkowa)
- `pow(x, y)` (`x` do potęgi `y`), `sqrt()`

Moduł ten zawiera także definicje dwóch stałych matematycznych: pi i e.

1.5 Moduł `cmath`

Zawiera definicje podstawowych funkcji matematycznych dla liczb zespolonych. Funcje te są oddzielone od tych z 'math', ponieważ niektórzy użytkownicy nie używają liczb zespolonych i wolą na przykład, aby $\sqrt{-1}$ zwróciło wyjątek zamiast liczby zespolonej. Funkcje te, to: `acos()`, `acosh()`, `asin()`, `asinh()`, `atan()`, `atanh()`, `cos()`, `cosh()`, `exp()`, `log()`, `log10()`, `sin()`, `sinh()`, `sqrt()`, `tan()`, `tanh()`. Wszystkie one zwracają liczbę zespoloną, nawet jeśli część urojona jest równa zero.

1.6 Moduł `random`

Moduł ten zawiera implementacje generatorów liczb pseudolosowych różnych rozkładów. Dla zmiennych całkowitych dostępny jest jednstajny wybór z przedziału (`randrange()`). Dla sekwencji – jednostajny wybór losowego elementu (`choice()`) i funkcja generująca losową permutację listy *in situ* (`shuffle()`). Dla liczb zmiennoprzecinkowych dostępne są rozkłady: jednostajny (`random()`, `uniform()`), normalny (`gauss()`, `normalvariate()`), log-normalny (`lognormvariate()`), wykładniczy (`expovariate()`), gamma (`gamma()`) i beta (`betavariate()`). Do generowania rozkładów kątów można wykorzystać rozkład kołowy jednostajny (`cunifvariate()`) lub von Misesa (`vonmisesvariate()`). Dostępne są także rozkłady Pareto (`patetovariate()`) i Weibulla (`wiebullvariate()`).
Inne funkcje związane z generatorami to seed(), getstate(), setstate() i jumpahead().
Prawie wszystkie funkcje tego modułu wykorzystują podstawową funkcję random(), która generuje losową liczbę zmiennoprzecinkową wg. rozkładu jednostajnego z przedziału (nawias zamknięty) 0.0, 1.0 (nawias otwarty). Python używa standardowego generatora Wichmanna-Hilla, łącząc trzy czysto multiplikatywne generatory o modułach 30269, 30307 i 30323. Jego okres wynosi 6,953,607,871,644. Choć generator ten jest dużo lepszej jakości niż funkcja rand() dostarczana przez większość bibliotek C, teoretyczne możliwości są bardzo podobne do tych, które oferuje pojedynczy liniowy generator o dużym module. Nie nadaje się on jednak do wielu zastosowań, w szczególności zupełnie nie nadaje się do zastosowań kryptograficznych.
Należy pamiętać o tym, że funkcje te nie nadają się do użycia przez wiele wątków. Aby to było możliwe należy utworzyć kilka instancji ukrytej klasy Random, zawierające niezależne generatory i przekazać je do poszczególnych wątków.

2 NumPy - numerical extensions to Python

Standardowe typy danych i procedury Pythona nie nadają się do poważnych zastosowań numerycznych. Konieczne było opracowanie zoptymalizowanych pod względem typowych operacji numerycznych typów danych i procedur. Stworzony został *Numerical Python*, zwany w skrócie *NumPy*. Podczas rozwoju tego pakietu kierowano się podejściem zastosowanym w takich językach jak Basis, MATLAB, rodzina języków APL, FORTRAN, S, S+ i innych. To dziedzictwo będzie widoczne natychmiast dla użytkowników Pythona, którzy już zetknęli się z tymi językami.
Można śmiało powiedzieć, że dzięki temu pakietowi możliwości obliczeniowe Python-a zbliżają się do tych, które posiada MATLAB czy IDL.
Numeric Python składa się z kilku modułów:

2.1 Numeric.py (z modułami pomocniczymi multiarray i umath)

Moduł ten rozszerza numeryczne możliwości Pythona dodając dwa nowe typy danych: sekwencję *wydajnie* implementującą tablice wielowymiarowe (`multiarray`) oraz nowy typ funkcji, zwany funkcją uniwersalną (`ufunc`), który działa wydajnie na tych nowych tablicach, a także innych typach sekwencyjnych. Definiuje również zbiór funkcji służących do operowania na obiektach tych typów oraz przeprowadzania konwersji między nimi a standardowymi typami Pythona. Stanowi on rdzeń NumPy będący podstawą do tworzenia bardziej skomplikowanych pakietów.

2.1.1 Obiekty tablic

Są to jednolite kolekcje liczb tego samego typu, w których poszczególne liczby mogą się zmieniać podczas 'życia' tablicy (w przeciwieństwie do wielkości samej tablicy). W niektórych zastosowaniach tablice liczb mogą zawierać pozycje niewłaściwe lub brakujące. W takim przypadku należy używać specjalnego pakietu opcjonalnego '`MA`'.
Operacje arytmetyczne na tych obiektach (tzn. z użyciem operatorów '+', '*' itp.) powodują wykonanie tych samych operacji na poszczególnych elementach tablicy i *nie* są odpowiednimi operacjami algebry liniowej.
Tablice posiadają takie atrybuty jak kształt (`shape`) - aby zmienić rozmiar tablicy wystarczy podstawić pod niego odpowiednią sekwencję, rząd - ilość wymiarów, czyli długość kształtu, kod typu – określający zawierane elementy, itp. Dostępnych jest kilka typów numerycznych (o różnych precyzjach, zależnych od danej platformy - np. na maszynie klasy Pentium od 8-bitowej liczby całkowitej po 128-bitową liczbę zespoloną), typ `Character`, a także referencja do dowolnego typu obiektu Pythona.

2.1.2 Funkcje uniwersalne

Wykonują operacje na tablicach i innych sekwencjach, w większości między odpowiednimi operacjami. Funkcja `array()` jest najprostszym sposobem utworzenia tablicy. Funkcja `resize()` tworzy nową tablicę o danym rozmiarze na podstawie istniejącej tablicy.

2.1.3 Inne funkcje i cechy tablic

Na elementach tablic *NumPy* można operować w podobny sposób jak na wbudowanym typie tablicowym Pythona.
Dostępne są też funkcje do manipulowania tablicami (np. transponowanie, sortowanie), wybierania części tablic na podstawie zawartości innych tablic, itd...
Python dokonuje automatycznego rzutowania typów tablic w przypadku niezgodności typów argumentów operatorów, na zasadach podobnych do tych przy rzutowaniu podczas standardowych operacji arytmetycznych na typach wbudowanych. Dostępna jest także funkcja `astype()` pozwalająca na dowolną konwersję, a także kilka innych służących do ograniczania wykorzystywanej pamięci, itp...

2.1.4 Matrix.py

Definiuje klasę `Matrix`, różniącą się tym od klasy tablicy (`Array`), że operator mnożenia '*' przeprowadza mnożenie macierzy, a operator potęgowania '**' jest zablokowany.

2.1.5 MLab.py

W założeniu jest to zbiór (obecnie ponad 30) funkcji, które mają zapewnić podstawową zgodność z MATLAB-em. Projekt zakłada maksymalne zbliżenie składni do tej z MATLAB-a.

2.2 Dodatkowe pakiety NumPy

Dostępne są także opcjonalne pakiety:

2.2.1 LinearAlgebra.py

Jest to prosty interfejs do niskopoziomowych procedur algebry liniowej udostępnianych przez bibliotekę LAPACK FORTRAN lub zgodną z nim bibliotekę C lapack_lite.
Oto niektóre funkcje:

`solve_linear_equations(a, b)` rozwiązuje układ równań liniowych z nieosobliwą macierzą kwadratową `a` i wektorem wyrazów wolnych `b`. Można rozwiązać jednocześnie kilka układów z różnymi wektorami wyrazów wolnych podając jako argument `b` tablicę dwuwymiarową (tzn. sekwencję wektorów).

`inverse(a)` odwraca nieosobliwą macierz kwadratową `a`.

`eigenvalues(a)` zwraca wartości własne macierzy kwadratowej `a`.

`generalized_inverse(a, rcond=1e-10)` – zwraca uogólnione odwrócenie (zwane też pseudo-odwróceniem albo odwróceniem Moore'a-Penrose'a) macierzy `a`.

`determinant(a)` zwraca wyznacznik kwadratowej macierzy `a`.

2.2.2 FFT.py

Jest to prosty interfejs do biblioteki FFTPACK FORTRAN przeprowadzającej szybką transformatę Fouriera rzeczywistych i zespolonych zbiorów danych. Ewentualnia jest biblioteka fftpack języka C, która jest oparta algorytmicznie na FFTPACK i udostępnia zgodny interfejs.
Na niektórych platformach można korzystać z optymalizowanych wersji tych bibliotek.
Moduł ten zawiera najczęściej używane procedury FFT:

`fft(dane, n=None, oś=-1)` przeprowadza n-punktową dyskretną transformatę Fouriera danych,

`inverse_fft(dane, n=None, oś=-1)` przeprowadza odwrotną dyskretną transformatę,

`real_fft(dane, n=None, oś=-1)`

`inverse_real_fft(dane, n=None, oś=-1)` podobnie, ale na danych rzeczywistych

`fft2d(dane, n=None, osie=(-2,-1))`

`real_fft2d(dane, n=None, osie=(-2,-1))` dwuwymiarowa FFT.

2.2.3 RandomArray.py

Moduł ten (w połączeniu z plikiem ranlibmodule.c) udostępnia interfejs wysokiego poziomu do modułu ranlib, zawierającego dobrej jakości implementację generator liczb losowych. Dzięki niemu można na przykład uzyskać w łatwy sposób całą tablicę liczb losowych o zadanym typie i parametrach rozkładu.

2.2.4 RNG.py

Ten moduł udostępnia niezależne strumienie liczb losowych. Dostępne są rozkłady: jednostajny, normalny, wykładniczy i log-normalny.

2.2.5 MA.py

Ten moduł udostępnia obsługę tablic przesłanianych (masked arrays), to znaczy posiadających nieznane lub niewłaściwe pola. Maska może zawierać jedynki, oznaczające elementy niewłaściwe. W takim przypadku moduł pilnuje, aby elementy te nie brały udziału w obliczeniach.
Moduł ten może okazać się bardzo pomocny w przypadku wykonywania obliczeń na niepełnych danych.

3 Scientific Python

Jest to pakiet modułów przydatnych przy obliczeniach naukowych. Do działania wymagana jest instalacja NumPy. Obecna wersja (2.2) zawiera szereg modułów zorganizowanych hierarchicznie. Oto ciekawsze z nich:

3.1 Functions

3.1.1 Derivatives

Ten moduł umożliwia automatyczne różniczkowanie funkcji o dowolnej liczbie zmiennych do dowolnego stopnia.
Bardziej wydajne liczenie pierwszych pochodnych umożliwia moduł `FirstDerivatives`

3.1.2 FindRoot

Zawiera funkcję `newtonRaphson(funkcja, lox, hix, xacc)`, która umożliwia znalezienie pierwiastka funkcji `funkcja` w przedziale od `lox` do `hix` z dokładnością do +/- `xacc`. Używany jest algorytm będący bezpieczną wersją algorytmu Newtona-Raphsona. `funkcja` musi być funkcją tylko jednej zmiennej i może używać wyłącznie operacji zdefiniowanych dla obiektów DerivVar w module FirstDerivatives.

3.1.3 Interpolation

Zawiera definicję klasy InterpolatingFunction, reprezentującą funkcję n zmiennych o m-wymiarowych wartościach. Konstruktorem jest `InterpolatingFunction(osie, wartości, default=None)`. `osie` to sekwencja tablic jednowymiarowych, jedna na każdą zmienną funkcji, określające wartości zmiennych na każdej z osi. `wartości` to tablica zawierająca wartości zmiennej na siatce. Wartość None zmiennej `default` oznacza, że funkcja jest niezdefiniowana poza siatką i każda próba zbadania jej wartości tam spowoduje wyjątek.
Klasa ta posiada metody służące do pobierania wartości w dowolnym punkcie, liczenia funkcju pochodnych i całek tej funkcji.
Dostępna jest także klasa NetCDFInterpolatingFunction wykorzystująca pliki NetCDF do definicji punktów siatki funkcji.

3.1.4 LeastSquares

Umożliwia dopasowywanie do danych ogólnych funkcji nieliniowych lub wielomianów metodą najmniejszych kwadratów z użyciem algorytmu Levenberg'a-Marquardt'a i automatycznych pochodnych.

3.1.5 Polynomial

Zawiera klasę ułatwiającą reprezentację wielomianów dowolnej liczby zmiennych.

3.1.6 Romberg

Zawiera funkcje przeprowadzające całkowanie metodą trapezów oraz metodą Romberga.

3.2 Geometry

Zawiera klasy dotyczące wielkości i obiektów geometrycznych. Klasy te obejmują takie zagadnienia jak tensory, wektory, kule, płaszczyzny, stożki, koła, proste i kraty w przestrzeni trójwymiarowej. Dostępne są też klasy opisujące pola tensorowe, skalarne i wektorowe wraz z niezbędnymi metodami umożliwiającymi liczenie dywergencji, laplasjanów, gradientów itp. wielkości.
Oprócz tego dostępne są klasy opisujące różne przekształcenia.

3.3 IO

Zawiera funkcje umożliwiające łatwe zapisywanie do plików tekstowych i odczytywanie z nich tablic dwuwumiarowych, funkcje wejścia/wyjścia zgodne z FORTRAN-em, NetCDF i PDB (Protein Data Bank). Dla tego ostatniego formatu zdefiniowane są klasy opisujące cząsteczki różnego stopnia złożoności.

3.4 MPI

Moduł ten zawiera interfejs Pythona do Interfejsu Przekazywania Komunikatów (Message Passing Interface – MPI). Istnienie takiego modułu pociąga za sobą jasno widoczne konsekwencje dla użyteczności Pythona w obliczeniach równoległych.

3.5 Physics

Zawiera różne moduły użyteczne przy prowadzeniu obliczeń i modelowaniu zjawisk fizycznych.

3.5.1 PhysicalQuantities

Zawiera typy danych reprezentujące wielkości fizyczne wraz z jednostkami. Zdefiniowane są operacje arytmetyczne, przy czym wynik będzie miał odpowiednią jednostkę. Wartości stałych fizycznych oparte są na wartościach zalecanych CODATA z 1986 roku.

3.5.2 Potential

Jest to jeszcze jeden moduł ułatwiający operacje na polach skalarnych, takie jak liczenie gradientów.

3.6 Statistics

Zawiera funkcje służące do liczenia różnych parametrów danych (takich jak wariancja, mediana itp.) oraz tworzenia histogramów.

3.7 Threading

Umożliwia zrównoleglenie obliczeń na maszynach wieloprocesorowych ze wspólną pamięcią.

3.8 Visualization

Dzięki temu modułowi można łatwo wizualizować obiekty trójwymiarowe przy pomocy różnych backend'ów (VRML, VMD, VPython), ale korzystając z praktycznie identycznego interfejsu. Moduł ten jest zorientowany na wizualizację danych naukowych, przez co nie zawiera wielu wyszukanych obiektów 3D, podobnie jak skomplikowanych definicji powierzchni takich jak tekstury.

4 PyLab – rozszerzenia ze strony Travisa Oliphanta

Interfejs do biblioteki gist, dzięki któremu jej używanie staje się bardziej podobne do MATLAB-a.

PyLapack niskopozioziomowy interfejs (autorstwa Konrada Hinsena) Pythona do całej biblioteki LAPACK. Aby go używać konieczne jest zainstalowanie bibliotek LAPACK i BLAS.

SparsePy moduł zawierający klasę implementującą macierze rzadkie w Pythonie. Atrybutami tej klasy są tablice (z modułu Numeric), a metody opierają się na bibliotekach SPARSEKIT2 autorstwa Yousefa Saada (w FORTRANIE) i SuperLU autorstwa Xiaoye Li i Jima Demmela (w C).

Cephesmodule udostępnia programistom Pythona większość funkcji specjalnych (jak funkcje eliptyczne i Bessela) z bibliotek cephes i amos. Implementuje te funkcje jako funkcje uniwersalne (ufunc), dzięki czemu można je wykonywać (bardzo szybko) na dowolnych tablicach. Udostępnia on także specjalną funkcję, dzięki której można tak 'opakować' zwykłą funkcję działającą na skalarach, aby operowała na całych tablicach.

Signaltools projekt ten ma na celu przybliżenie możliwości obróbki sygnałów Pythona do tej, jaką posiadają inne systemy zorientowane na tablice, jak MATLAB.

quadrature.py moduł umożliwiający wykonywanie kwadratur Gaussa na skończonym obszarze z dowolnych funkcji Pythona.

optimize.py moduł zawierający procedury optymalizacyjne napisane w czystym Pythonie. Obecnie zawiera m.in. implementacje algorytmów: simpleksowego Neldera-Meada i quasi-Newtonowskiego Broydena-Fletchera-Goldfarba-Shanno.

FFTW interfejs do biblioteki FFTW (w C). Jest to bardzo szybka implementacja FFT (testy autora wskazują, że jest o około 18-25% szybsza od fftpack-a). Zawiera również obsługę tablic wielowymiarowych (nie tylko 2D).

5 Multipack

Jest to interfejs dla NumPy do zbioru procedur w FORTRANIE – ODEPACK, QUADPACK, MINPACK itd... Obecnie pakiet zawiera funkcje, dzięki którym można numerycznie:

- rozwiązać N równań nieliniowych z N niewiadomymi
- zminimalizować m równań nieliniowych z n niewiadomymi (Levenberg-Marquardt)
- zcałkować zwykłe równanie różniczkowe
- zcałkować funkcję 1, 2 lub 3 zmiennych
- dopasować spline 1 lub 2D do zbioru punktów i znaleźć pochodne, całki, interpolacje, itp. tych spline'ów

6 Pysimplex

Moduł ten zawiera podstawowe podstawowe narzędzia programowania symbolicznego służące do konstruowania, rozwiązywania i optymalizowania układów równań i nierówności liniowych. Zawiera on implementację klasycznego algorytmu optymalizacji liniowej SIMPLEX, a także filtr służący do analizowania i optymalizacji modeli liniowych zakodowanych w standardowym formacie MPS.

7 Inne ciekawe moduły

7.1 real.py

Jest to biblioteka definiująca klasę `Real`, pozwalająca wyrażać dowolnie precyzyjne liczby, dzięki czemu można przeprowadzać obliczenia z 'nieskończoną' precyzją. Każda liczba zawiera przybliżenie swojej precyzji. Udostępnione są metody pozwalające obliczać liczby z coraz większą ilością cyfr. `real.py` pozwala

- liczyć z bardzo dużą precyzją, np. 1000 cyfr

- liczyć na bardzo dużych lub bardzo małych liczbach - do około $10^5$00000000
- liczyć bez niespodziewanej utraty precyzji: gwarancja na każdą cyfrę.
- liczyć z nieograniczoną precyzją, drukując cyfry w locie
- drukować liczby zmiennoprzecinkowe z ich rzeczywistą dokładnością

7.2 Pyfortran i FPIG

– umożliwiają łatwe łączenie kodu w Pythonie i Fortranie.

7.3 surd i yarn

– moduły implementujące liczby wymierne

7.4 PYML

Jeden z interfejsów Pythona do Mathematica'i, wykorzystujący MathLink. Dzięki temu programista Pythona może obliczać wyrażenia Mathematica'i.

7.5 Sparse Linear Equation Solver

Ten moduł (autorstwa Neila Schemenauera) umożliwia rozwiązywanie rzadkich równań liniowych. Używa załączonej biblioteki Kennetha S. Kunderta Sparse1.3. Moduł ten przydaje się przy rozwiązywaniu dużych układów równań nieliniowych metodą Newtona-Raphsona.

7.6 Inne

Dostępnych jest jeszcze wiele najróżniejszych pakietów ułatwiających obliczenia lub integrujących Pythona z innymi narzędziami i językami programowania oraz ułatwiających różnego rodzaju obliczenia. Wszystkich nie sposób tutaj wymienić, można jednak podać kilka odnośników do stron zawierających te rozszerzenia.

Number Crunching and other Scientific Extensions and Tools `http://www.python.org/topics/scicomp/numbercrunching.html`

Math section of the Contributed Modules FTP site `ftp://ftp.python.org/pub/python/contrib/Math/`

Contributed Modules Web page `http://www.python.org/download/Contributed.html#Math`

Vaults of Parnassus `http://www.vex.net/parnassus/apyllo.py/684222876`

8 Narzędzia do wizualizacji

Oto niektóre moduły służące do wizualizacji danych, niezwiązane bezpośrednio z wyżej wymienionymi pakietami obliczeniowymi.

Mathematica.py moduł generujący kod dla procedur rysujących wykresy Mathematica'i. Dzięki niemu można przy pomocy programu Mathematica rysować wykresy danych obliczonych w Pythonie.

Gnuplot.py interfejs do znanego programu do tworzenia wykresów – gnuplot. Jest zorientowany obiektowo, dzięki czemu można ustawiać parametry wizualicji (style, kolory itp.) dla każdego zbioru danych osobno.

grace_np.py interfejs do programu do tworzenia wykresów 'Grace'. Na razie otwiera tylko potok do programu, przez który można wydawać mu komendy.

BLT rozszerzenie umożliwiające tworzenie wykresów X/Y i słupkowych z wykorzystaniem Tk

DISLIN wysokopoziomowa i łatwa w użyciu biblioteka pozwalająca na wizualizację danych w postaci krzywych, wykresów słupkowych, kołowych, kolorowych trójwymiarowych, powierzchni, konturów i map.

gdmodule umożliwia rysowanie prostych kształtów w formacie GIF

PyOpenGL interfejs do biblioteki OpenGL, który może posłużyć do stworzenia nowego narzędzia do wizualizacji

Więcej modułów do wizualizacji można znaleźć na stronach `http://www.python.org/topics/scicomp/plotting.html` lub `http://starship.python.net/crew/jhauser/plot-res.html`. O wszystkich nie sposób tutaj wspomnieć, z racji na ograniczone miejsce.

9 Podsumowanie

Python jako język interpretowany, bez możliwości optymalizacji kodu pod konkretną platformę jako taki nie nadaje się raczej do bardzo intensywnych obliczeń. Jednak ze względu na swoją elastyczną budowę, możliwość łączenia z gotowymi bibliotekami napisanymi w innych językach, a także na bardzo rozbudowane struktury danych i podejście obiektowe, bardzo dobrze służy jako "klej" łączący sprawdzone procedury numeryczne (napisane na przykład w FORTRANIE) z różnorodnymi narzędziami służącymi do wizualizacji. Można wywoływać dostarczone funkcje na danych, a dane wynikowe (po odpowiedniej obróbce) wysłać do odpowiedniego narzędzia do wizualizacji. Taka jest głównie rola języków skryptowych i tak Python jest używany na przykład przez fizyków z Los Alamos National Laboratory, gdzie był wykorzystywany przy obróbce danych pochodzących z kodu SPaSM [1] służącego do modelowania dynamiki cząsteczek.

Bibliografia

[1] David M. Beazley. Feeding a large-scale physics application to python. 1997.

[2] David Ascher et al. *Numerical Python*. `http://www.pfdubois.com/numpy/`, 2001.

[3] Konrad Hinsen. *ScientificPython User's Guide*. `http://starship.python.net/crew/hinsen/scientific.html`, 2001.

[4] Travis Oliphant. *Travis Oliphant's Python Pages*. `http://pylab.sourceforge.net/`, 2001.

[5] Guido van Rossum i Fred L. Drake, Jr., editor. *Python Library Reference*. `http://www.python.org/doc/current/lib/lib.html`, 2001.

[6] Guido van Rossum i Fred L. Drake, Jr., editor. *Python Tutorial*. `http://www.python.org/doc/current/tut/tut.html`, 2001.