

Małgorzata Klisowska
goskli@op.pl
Katedra Fizyki Doświadczalnej
Wydział Matematyczno – Przyrodniczy
Uniwersytet Rzeszowski
Rzeszów
Stanisław Topolewicz
stopolewicz@gmail.com
Katedra Fizyki Teoretycznej
Wydział Matematyczno – Przyrodniczy
Uniwersytet Rzeszowski
Rzeszów

Otwarte środowisko aplikacji matematycznych SAGE w (samo)kształceniu fizyczno – informatycznym

Zagadnienia kluczowe

Alternatywą dla postawy „zarządzanie wiedzą z sieci” (bez przyswojenia i włączenia nowych elementów do posiadanej już operacyjnej struktury wiedzy np. matematyczno-przyrodniczej) jest możliwość połączenia kompetencji matematycznych i informatycznych (zarówno uczących się, jak i nauczyciela [Klisowska, 2003]) oraz rozwiązań metodyczno-dydaktycznych upodabniających uczenie się do procesu badawczego [Klisowska, 2013]. Podstawową kompetencją nauczyciela w rozważanym zakresie powinna być umiejętność metodycznego powiązania stosowanych programów komputerowych z elementami zajęć tak, by wspomagały proces myślenia produktywnego uczącego się [Adamczyk, 2008] oraz jego rzeczywistą aktywność na danym poziomie kształcenia (przykłady: [Kowalewski, 2014]).

Stymulatorem aktywizacji procesu dydaktycznego oraz determinantą przejścia od statycznego przekazu (zarówno słownego, jak i e-medialnego) do kształtowania *w działaniu i przez działanie* mogą okazać się otwarte zasoby narzędzi informatycznych nowej generacji. Pakiety zawierające wypracowane wzorce i rozwiązania, umożliwiające jednocześnie tworzenie przez uczących się własnego środowiska zadaniowego (na miarę aktualnych umiejętności, własnych zainteresowań i samoświadomości w zakresie samokształcenia) mogą wpłynąć korzystnie na kompetencje kluczowe z matematyki i informatyki.

Odpowiednio wybrane aplikacje – z dominantą na działanie w przestrzeni treści merytorycznych z fizyki lub innych nauk przyrodniczych [Francikowski, Kaczmarzyk, 2010] – mogą przyczynić się zmian w sposobach wykorzystania TiK w kształceniu [Nowoczesne..., 2014]. Istotnym jest, by aplikacje takie umożliwiały transformację aktualnych strategii kształcenia – w których technologie informacyjne pełnią funkcję wspierającą – w kształcenie z dominującą funkcją integracyjną. Okazuje się to możliwe (i to bez znaczących nakładów finansowych) dzięki coraz bardziej popularnym pakietom typu *open source* [What is...], np. takim, jak SAGE [www.sagemath.org].

Środowisko aplikacji matematycznych SAGE

SAGE jest przykładem bezpłatnego „otwartego” oprogramowania matematycznego, które (wypierając powoli drogie programy komercyjne) zaczyna być wykorzystywane nie tylko w celach naukowo-badawczych, ale i w kształceniu [Topolewicz, 2012(b)]. Jest w pełni otwarty nie tylko pod względem kodu, ale i metodologii rozwoju. Jako projekt integracyjny, zbierający w jeden spójny pakiet cały szereg bibliotek oraz programów pomocniczych, jest dystrybucją wielu darmowych pakietów [SAGE Components...] dedykowanych zadaniowo: każdy pakiet wykonuje dobrze to do czego został utworzony (np. Matplotlib, NumPy, Sage Notebook, itp. – zob. [Aplikacje]). Poprzez dodawanie wspólnego interfejsu, SAGE pozwala na bezkonfliktową, kompatybilną wymianę wyników pomiędzy pakietami.

Kluczowym elementem, zapewniającym *spójność* jest obiektowy język programowania: **Python** [www.python.org], rozprowadzany na otwartej licencji. Otwarte środowisko aplikacji matematycznych SAGE, w którym wiedza matematyczna została obudowana wokół interpretera Pythona, jest doskonałą alternatywą *otwartoźródłową* dla takich programów, jak Mathematica, Maple czy Matlab. Interaktywna powłoka (*interactive shell*) Pythona – **IPython** [ipython.org] – potrafi wykonywać dowolne polecenia w tym języku. Od najprostszyc zadań aż po bardzo złożone: może wyliczać dowolne wyrażenia matematyczne, możemy także przypisywać pewne wartości do zmiennych, które są pamiętane tak długo, jak długo powłoka jest uruchomiona. Ta funkcjonalność interaktywnej powłoki decyduje, że IPython jest wykorzystywany w obliczeniach numerycznych, wykonywanych np. przy użyciu bibliotek: SciPy, SymPy, Matplotlib, Pandas, itp. – zob. [Aplikacje], z którymi się bardzo dobrze integruje [Topolewicz, 2012(a)]. Co ważne: może również wykonywać obliczenia symboliczne, jak też np. załadować jakiś zupełnie zewnętrzny moduł.

W odróżnieniu od innych języków programowania, w języku Python – *import* – jest instrukcją. Oznacza to, że może być użyta nie tylko na początku, ale w dowolnym miejscu

programu, w szczególności – może być częścią instrukcji warunkowej (przykład, zapisany w całości w Pythonie, przedstawiono na rysunku 1.).

1 Fragment kodu

To jest fragment kodu realizujący proste całkowanie funkcji $f(x) = \sin(x)$ metodą Monte Carlo.

W poniższym kodzie można zauważyć dwa sposoby importowania metod z danego modułu: importowanie konkretnej metody lub importowanie wszystkich funkcji danego modułu. W kolejnych wierszach mamy przypisanie wartości do zmiennej (Python sam określa typ zmiennej).

Następnie typowa konstrukcja pętli *for*, funkcja *range* generuje listę liczb całkowitych od 1 do *iteracje*. Po dwukropku zaczyna się (wcięte o 4 spacje - to standard w Pythonie) kod wykonywany w pętli - losowanie współrzędnych punktu (x, y) oraz obliczana jest wartość funkcji $\sin(x)$ dla wylosowanej współrzędnej.

Kolejnym krokiem jest instrukcja warunkowa *if* - program sprawdza, czy wylosowana współrzędna y jest równa lub mniejsza niż y obliczone - wtedy do sumy dodawane jest jeden. Ta funkcja również kończy się dwukropkiem, a wykonywana zawartość jest wcięta w stosunku do *if*.

Następnie wypisujemy obliczone pole jako stosunek liczby trafień do całkowitej liczby strzałów pomnożony przez pole obejmujące funkcję $\sin(x)$ (funkcja $n()$ pozwala na otrzymanie numerycznego przybliżenia liczby π). Powyżej znajduje się też obliczona analitycznie całka $\int_0^\pi \sin(x) dx$

```
import random
from sage.symbolic.integration.integral import definite_integral
sum = 0
iteracje = 100000
for l in range(1,iteracje):
    x_random = random.uniform(0,pi)
    y_random = random.uniform(0,1)
    y_calculated = sin(x_random)
    if y_random <= y_calculated:
        sum = sum+1
print "Warto dok adna: " + str(definite_integral(sin(x),x,0,pi))
print "Warto obliczona metod Monte Carlo: " + str((sum/iteracje)*n(\
pi))
Wartość dokładna: 2
Wartość obliczona metodą Monte Carlo: 2.00216841405931
```

Rysunek 1. Fragment kodu w języku Python. Źródło własne (wygenerowano 10.06.2014).

Umożliwia to wykorzystanie implementacji (w formie kodów źródłowych procedur bądź programów) zawartych w poszczególnych pakietach. Na przykład: implementacje szeregu procedur numerycznych, jak: różniczkowanie, całkowanie numeryczne, algorytmy rozwiązywania równań różniczkowych, algorytmy z algebry liniowej, funkcje specjalne, narzędzia do programowania równoległego i wiele innych, użytecznych do obliczeń naukowych (fizycznych, inżynierskich) zawarte zostały w pakiecie SciPy. Pakiet tworzą moduły implementujące konkretne procedury matematyczne czy graficzne (rysunek 2).

SciPy.org

Install Getting Started Documentation Report Bugs Blogs

SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:

- NumPy: Base N-dimensional array package
- SciPy library: Fundamental library for scientific computing
- Matplotlib: Comprehensive 2D Plotting
- IPython: Enhanced Interactive Console
- Sympy: Symbolic mathematics
- pandas: Data structures & analysis

Rysunek 2. Oficjalna strona z pakietem SciPy. Źródło: <http://www.scipy.org/>.

„Pythonowe” skrypty i aplikacje przydają się nie tylko do zamkniętych projektów w zastosowaniach naukowych, inżynieryjno-obliczeniowych czy finansowych, gdzie potrzebne jest przetwarzanie danych i generowanie wykresów, zestawień (np. z arkuszy Excela [www.python-excel.org]). Python wykorzystywany jest do tworzenia dynamicznych stron oraz serwisów internetowych, aplikacji desktopowych działających na komputerach użytkowników (wliczając w to także gry [zob. *Invent...*]). Można go także wykorzystywać w aplikacjach sieciowych, czy skryptach np. generujących zestawienia i raporty. Używany jest jako wszechstronny język programistyczny, a jego zastosowanie do tworzenia różnorodnych aplikacji (np. dla serwisów społecznościowych czy WWW działających w chmurze) czyni go też atrakcyjnym z edukacyjnego punktu widzenia [*Online Python...*]. Świadczą o tym zrealizowane krajowe projekty, jak np. PPCG [*Projekt...*], iCSE [*Nowoczesne...*, 2014], czy subprojekt TI: Programowanie z Pythonem [*TI: Programowanie...*] zrealizowany w ramach projektu *Fizyka wobec wyzwań XXI w.* [fizykaxxi.fuw.edu.pl]. W sieci dostępne są liczne e-przewodniki (np. [*Invent...*], [*Dokumentacja...*], [Maliński, 2008]) i kursy (np. [*Google's...*], [*TI: Programowanie...*], [*Zanurkuj...*]), tematyczne blogi oraz biblioteki tworzone zarówno przez profesjonalnych programistów, jak i pasjonatów (np. projekt PPCG (Polish Python Coders Group) Stowarzyszenia Polska Grupa Użytkowników Pythona [*Projekt...*]).

Jako warstwa odpowiadająca za integrację środowiska aplikacyjnego różnorodnych modułów, komponentów oraz samodzielnych programów, Python ma za zadanie zapewnienie spójnej semantycznie wymiany komunikatów oraz poprawnego działania wszystkim tym elementom, a w efekcie: udostępnienie synergicznej funkcjonalności – większej, niż w przypadku niezależnego korzystania z poszczególnych modułów czy programów.

Konwersja wymienianych danych do ujednoczonych struktur komunikatów stanowi swoisty tester jakości. Okazuje się, że Python to stosunkowo proste narzędzie, ale z doskonałą obsługą błędów: posiada niejako wbudowany mechanizm testowania napisanych przez użytkowników metod – na zasadzie: wpisujemy wywołanie metody z odpowiednimi parametrami i oczekiwany wynik, a Python informuje o błędach. Ponieważ ten mechanizm jest używany w komentarzach będących dokumentacją programu (tzw. *docstring*) jest łatwy do stosowania. Cała dokumentacja Pythona oraz SAGE została wygenerowana za pomocą (napisanego w również w Pythonie) programu Sphinx [sphinx-doc.org]. Ten przyjazny dla użytkownika, funkcjonalny system tworzenia dokumentacji dla projektów w języku Python pozwala na wygenerowanie jej w różnych formatach (np. HTML, LaTeX, epub), jak również umożliwia przeprowadzenie testowania kodu źródłowego [Topolewicz, 2012(c)].

Dostęp do kodu źródłowego i możliwość jego testowania to istotne atrybuty otwartego oprogramowania (chodzi przede wszystkim o to, by dodawane nowe kody nie popsęły wcześniejszych oraz by nie rozwijać kodu, który nie współdziała z istniejącymi już kodami). W przypadku aplikacji SAGE mamy do dyspozycji jeszcze inne moduły do testowania kodu: np. UnitTest – moduł samego Pythona do testów jednostkowych (np. klas, czy też metod). SAGE wykorzystuje testy do zapewnienia wysokiej jakości kodu (np. zmiany w kodzie nie wprowadzają kolejnych błędów, a jeśli by się to przytrafiło to testy – przynajmniej w części – to wyłapują). Nadmienmy, że otwarte oprogramowanie oznacza zwiększone bezpieczeństwo – ponieważ kod źródłowy jest wystawiany na widok publiczny, jego użytkownicy sprawdzają go z ekstremalną dokładnością. Błędy są natychmiastowo wykrywane i poprawiane. Skutkuje to stosunkowo wysoką niezawodnością aplikacji *open source* w porównaniu do zamkniętych, własnościowych programów. Otwarty kod oznacza również nieograniczone możliwości *rozbudowy* i *dostosowywania* do potrzeb, a także *integracji* z innymi rozwiązaniami (wykorzystując środowisko SAGE można uzyskać dostęp do zainstalowanych programów komercyjnych jak np. Mathematica, Matlab czy Maple).

Implikacje metodologiczno – metodyczne

Z dydaktycznego punktu widzenia – wykorzystanie testów do testowania kodu źródłowego pozwala na kształcenie informatyczne studentów w zakresie tzw. programowania zwinnego (*Agile software development*). Częścią metodologii *Agile* jest tzw. „programowanie oparte na testach” (*test-driven development*), zaś metodyka oparta jest na zdyscyplinowanym zarządzaniu projektem, które zakłada częste inspekcje wymagań i rozwiązań wraz z procesami adaptacji (zarówno specyfikacji jak i oprogramowania). W myśl jednej z podstawowych wartości, na których bazują metodologie *Agile*: „reakcja na zmianę ponad realizację planu” [Rycharski, 2010] – strategię działania można ująć schematycznie: wpieryw definiujemy problem, piszemy testy, które powinny zostać spełnione, następnie przeprowadzamy taki iteracyjny proces: *TEST* → *POPRAWIANIE PROGRAMU* → *KOLEJNY TEST*, itd. Test powinien być prosty, implementujący funkcjonalność, z uporządkowanym kodem (tak, by w wyniku wprowadzania zmian w projekcie/programie funkcjonalność zasadniczo nie ulegała zmianie).

Dla refleksyjnego umysłu uzyskany wynik nie powinien być czymś „samym w sobie” ostatecznym, lecz elementem danym do przetwarzania, również mentalnego przez myśl, która nada mu określony sens oraz interpretację: fizyczną, aplikacyjną (a nawet metodologiczną lub filozoficzną). Być może dzięki takim ćwiczeniom uda się – przez analogię – zredukować

zbiory zawiłych problemów (wynikających z dynamicznego rozwoju świata fizycznego, bądź z równie dynamicznego rozwoju cywilizacji) do prostszych, łatwiej interpretowalnych zależności. Takie nastawienie pozwala zasadniczo przeobrazić koncepcję „wyniku końcowego” (zarówno jako konkretnej liczby, wykresu czy aplikacji) w orientację na dynamiczny charakter procesu: zmienność, stawanie się, prawidłowości, powtarzalność oraz inne atrybuty, jak uniwersalny charakter procedur i języka. Jak widać przypomina to bardzo *proces badawczy* (choć fizycy nie testują programów lecz badają rzeczywistość fizyczną).

Poznanie uwarunkowań zmian jakościowych we współczesnym kształceniu fizyczno – informatycznym domaga się poszukiwania sposobów i metod ożywiania łączności między studentem (uczniem) a kształceniem matematycznym. Z tego też względu otwartoźródłowa aplikacja matematyczna SAGE może sprzyjać integracji międzyprzedmiotowej w zakresie fizyki i informatyki – nie tylko merytorycznej – ale i metodologicznej. Tak świadomie ukierunkowana modyfikacja działań dydaktycznych, których celem będzie integracja przedmiotów kierunkowych z metodami komputerowymi może przyczynić się do wzrostu aktywności poznawczej i badawczej uczących się poprzez realizację strategii nauczania – uczenia się problemowego z wykorzystaniem TI [zob. Klisowska, 2003, s.119-120].

Znaczenie matematyki w fizyce, astronomii, inżynierii, ekonomii czy informatyce oraz rosnący udział tej dyscypliny (np. dzięki możliwości modelowania i symulacji) w rozwoju i postępie dyscyplin, w których modelowanie matematyczne zaczyna być stosowane na szeroką skalę (jak np. biologia, ekologia, fizjologia, epidemiologia, ochrona środowiska, socjologia czy psychologia) implikują warunek konieczny stosowania takich narzędzi matematycznych, które ułatwią integrację metod i środków (przy równoczesnej minimalizacji kosztów kształcenia np. dzięki darmowym zasobom aplikacyjnym). SAGE dysponuje wieloma sposobami wizualizacji funkcji matematycznych, zarówno jednej zmiennej, jak i dwóch zmiennych. Można również przedstawiać na wykresie funkcje parametryczne (dla których współrzędne danego punktu na wykresie są funkcją pewnego parametru), wizualizować wygląd macierzy, a nawet pól wektorowych. Jest to niezwykle istotne (nie tylko z punktu widzenia fizyka-badacza): w kształceniu fizyczno-informatycznym umożliwia różnicowanie toku zajęć.

W fizyce niejednokrotnie parametry danego modelu są wyznaczane za pomocą rozwinięcia funkcji układowej w *szereg Taylora*. Przykład implementacji w SAGE dla tego zagadnienia przedstawiono w formie rzutu zapisu kodu (rysunek 3.), zaś wygenerowane przykładowe wykresy powstałe w wyniku modelowania i symulacji zebrano na rysunku 5.

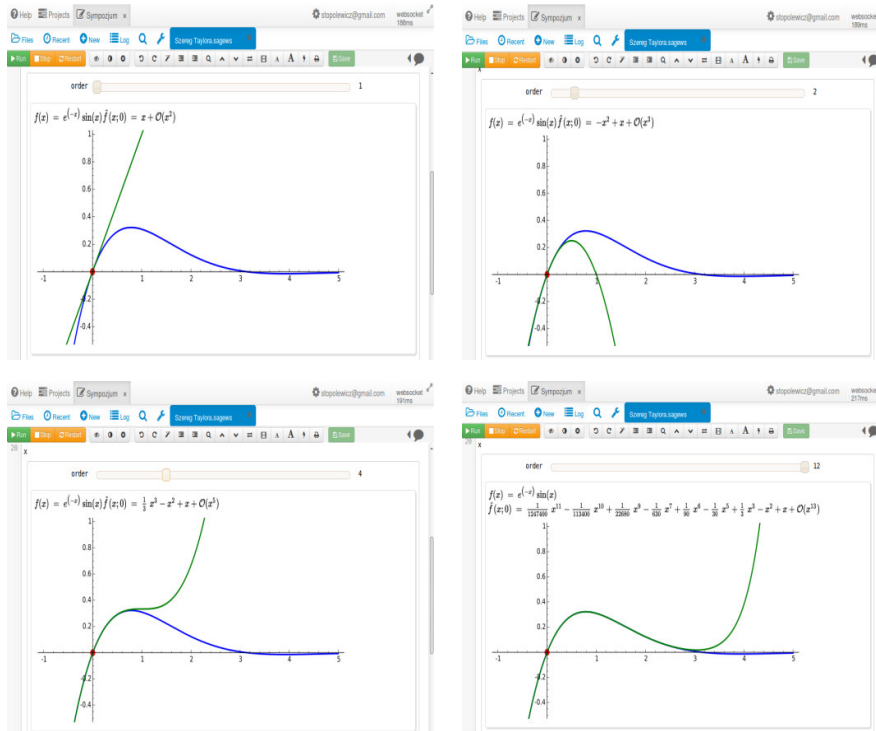
1 Szereg Taylora

Dekorator `@interact` pozwala na stworzenie interaktywnych wykresów i obliczeń:

```
var('x')
x0 = 0
f = sin(x)*e^(-x)
p = plot(f,-1,5, thickness=2)
dot = point((x0,f(x=x0)),pointsize=80,rgbcolor=(1,0,0))
@interact
def _(order=(1..12)):
    ft = f.taylor(x,x0,order)
    pt = plot(ft,-1,5, color='green', thickness=2)
    html('$f(x)\;=\;\%s$\'%latex(f))
    html('$\hat{f}(x;\%s)\;=\;\%s+\mathcal{O}(x-\%s)$\'%(x0,latex(ft),order\
+1))
    show(dot + p + pt, ymin = -.5, ymax = 1)
x
```

W powyższym kodzie f to funkcja, którą aproksymujemy szeregiem Taylora w punkcie x_0 , p to wykres tej funkcji, dot to obiekt graficzny - czerwona kropka, natomiast pt to zielony wykres funkcji ft - wielomianu Taylora

Rysunek 3. Przykład implementacji. Źródło własne (wygenerowano 10.06.2014).

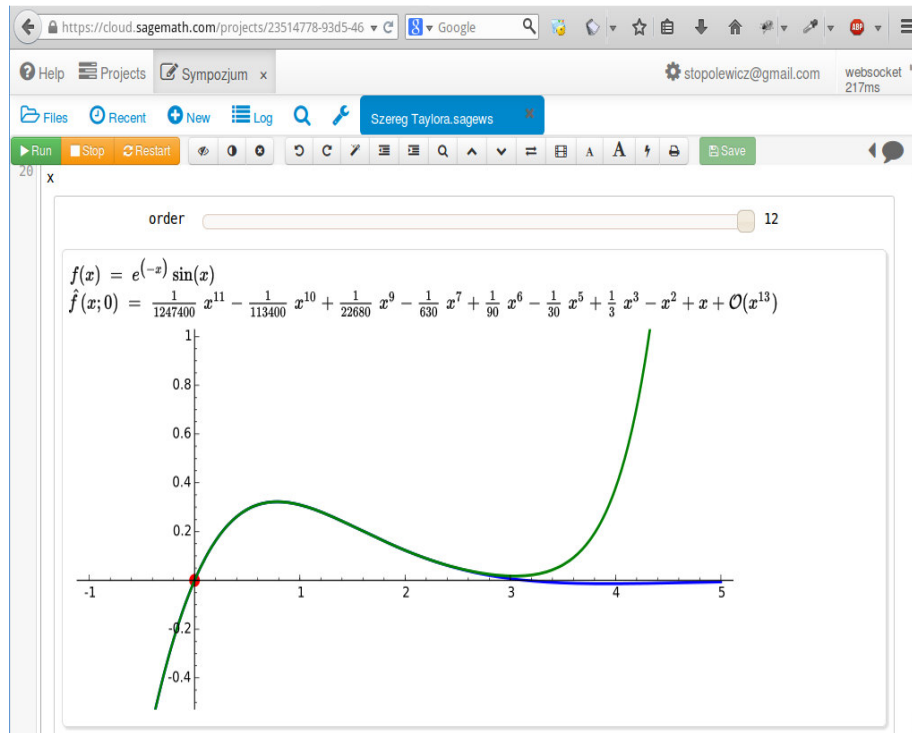


Rysunek 4. Widok wyników modelowania i symulacji – zrzut ekranu programu SAGE.

Źródło własne (wygenerowano 10.06.2014).

Publikacja książkowa uniemożliwia dynamiczną prezentację aplikacji. Aplikacja może zostać zaprezentowana podczas referatu aplikacyjnego lub w e-publikacji. Ułatwieniem jest, że SAGE może działać na danym komputerze (dostępny przez przeglądarkę lub powłokę IPython), na serwerze (dostępny przez przeglądarkę lub linię komend), w bezpłatnej chmurze SAGE Cloud [<https://cloud.sagemath.com/>]. SAGE Cloud umożliwia prezentację *online*

wykonanych projektów bezpośrednio z profilu autora. Przykładowy zrzut ekranu przedstawiono na rysunku 5. (więcej przykładów własnych: [Topolewicz, 2014]).



Rysunek 5. Projekt „Szereg Taylora” zrealizowany w bezpłatnej chmurze SAGE Cloud.
Źródło własne (wygenerowano 10.06.2014).

Zalew obrazowości osłabia zdolność widzenia. Jednakże, dynamiczne wygenerowanie wykresu pobudza intelektualnie i emocjonalnie; opisanie świata w takiej perspektywie staje się istotnym czynnikiem powalającym fizykom na zrozumienie zjawisk i procesów; uzyskane efekty wywołują wrażenia estetyczne, rozbudzają ciekawość („jaki obraz uzyskamy, gdy...”, itp.) i kreatywność (zróżnicowaną w zależności od zainteresowań). Celem staje się dążenie do zrozumienia różnorodnych ekspresji, będących elementami zmiennej, dynamicznej całości, której wyrazem jest wykres. Dzięki atrakcyjności wizualnej wykresów przestrzennych (w 3D) grafika komputerowa stała się kolejną dyscypliną artystyczną. Preferowany w kształceniu w zakresie nauk ścisłych tok poznawczy, problemowy czy praktyczny – może zostać wsparty tokiem eksponującym, poruszającym się w sferze odczuć i emocji. Wizualna (w tym również estetyczna) strona tego, co wydaje się zbyt trudne teoretycznie, może wpływać na zmianę postaw w odniesieniu do nastawienia wobec przedmiotów kształcenia oraz własnych działań: samokształcenie staje się przygodą intelektualną, sposobem relacji społecznościowych w sieci (fora, grupy sympatyków) oraz poza nią (spotkania, warsztaty, konferencje pasjonatów [zob. <http://pl.pycon.org>]).

Podsumowanie

SAGE to środowisko algebry komputerowej, które łączy w sobie wiele istniejących pakietów matematycznych w jedno środowisko. SAGE oferuje Python jako podstawowy język środowiska, szereg modułów matematycznych (np. algebra liniowa, kombinatoryka, analiza matematyczna, teoria grafów, statystyka), udostępnia różne wykresy, umożliwia modelowanie i wizualizacje (w 2D, 3D), zapewnia integrację z innymi, komercyjnymi, pakietami matematycznymi.

W perspektywie integracji kształcenia informatyczno-fizycznego za pozytywną oceną środowiska aplikacji matematycznych SAGE przemawiają takie czynniki, jak:

- otwartość (pod względem dostępu do kodu źródłowego, ale i metodologii rozwoju);
- funkcjonalność i elastyczność (możliwość modyfikowania, dodawania nowych funkcji, dostosowywania do potrzeb projektu, poprawiania błędów),
- spójność (działanie w oparciu o jeden z najlepszych języków programowania – Python),
- kompatybilność (ucząc się działać w SAGE uczymy się jednocześnie języka Python),
- redukcja czasu implementacji (typy w SAGE, jak i w Pythonie, są dynamiczne, można między nimi dokonywać konwersji),
- gotowe komponenty i rozwiązania (SAGE zawiera mnóstwo bibliotek z różnych dziedzin matematyki, można też wykorzystać każdą bibliotekę Pythona),
- notatniki (notebook) bazują na IPython; mogą zawierać tekst (HTML lub Markdown), wzory w LaTeX'u i obliczenia,
- dwa interfejsy oferowane do wyboru: konsolowy i przeglądarkowy,
- może działać na danym komputerze, na serwerze lub w bezpłatnej chmurze SAGE Cloud,
- integracja (w SAGE możemy uruchomić również instancje programów komercyjnych).
- aspekt prawny i finansowy: SAGE to środowisko oprogramowania *open source* na licencji GPL, wolne od opłat za program i jego użytkowanie.

Aplikacje [dostęp 15.06.2014]

IPython. Interactive computing. <http://ipython.org/>

Matplotlib. Python 2D plotting library. <http://matplotlib.org/>

Pandas. Python Data Analysis Library. <http://pandas.pydata.org/>

Plotly Python API. <https://plot.ly/python/>

Python. <https://www.python.org/>

Python-Excel. <http://www.python-excel.org/>

SciPy. <http://www.scipy.org/>

SAGE: open-source mathematics software. <http://www.sagemath.org/>

SAGE Notebook. <http://nb.sagemath.org/>

Sphinx. Python Documentation Generator. <http://sphinx-doc.org/>

SymPy. Python library for symbolic mathematics. <http://sympy.org/>

Bibliografia

Adamczyk A., *Animacje do zajęć z fizyki (i nie tylko). To może zrobić każdy!* „Fizyka w Szkole” 2008, nr 4 (297 (LIV)), s. 27-33

Dokumentacja Pythona. Wydanie 2.3 (22 grudnia 2003). <http://pl.python.org/docs/> [dostęp 15.06.2014]

Francikowski J., Kaczmarzyk M.: *Szkoła in silico, czyli druga młodość komputera szkolnego.* „Edukacja i Dialog” 2010, 01-02 (214/215), s. 44-47

Google's Python Class. Educational materials provided by Google. <https://developers.google.com/edu/python/> [dostęp 15.06.2014]

Invent with Python. Learn to program by making computer games. <http://inventwithpython.com/> [dostęp 15.06.2014]

Klisowska M.: „Infobroker” vs „badacz”. *O e-determinantach (nie)efektywności transferu wiedzy fizycznej.* [W:] *Człowiek – Media – Edukacja.* Red. nauk. J. Morbitzer. Wydawca: Katedra Technologii i Mediów Edukacyjnych, Uniwersytet Pedagogiczny, Kraków 2013, s. 206-213

Klisowska M.: *Kompetencje nauczyciela przedmiotów przyrodniczych w świetle edukacyjnego modelu komunikacji.* [W:] *Jakość kształcenia a kompetencje zawodowe nauczycieli przedmiotów przyrodniczych.* Red. nauk. R. Gmoch. Wydawca: Wydawnictwo Uniwersytetu Opolskiego, Opole 2003, s.117-122

Kowalewski M.: *Matematyka łatwiejsza niż myślisz. Sage w nauczaniu matematyki w szkołach ponadgimnazjalnych.* Wydawca: Projekt iCSE. Uniwersytet Śląski, Katowice 2014, e-book: <http://icse.us.edu.pl/e-book/> [dostęp 10.03.2014]

Maliński P.: *Podstawy Pythona. Kurs programowania w Pythonie. Opis podstaw języka i popularnych modułów dodatkowych* (2008). <http://www.python.rk.edu.pl/w/p/podstawy/> [dostęp 15.06.2014]

Nowoczesne komputerowe metody kształcenia dla regionalnych kadr innowacyjnej gospodarki: iCSE. UŚ, Katowice 2014. <http://icse.us.edu.pl/> [dostęp 15.06.2014]

Online Python Tutor. <http://www.pythontutor.com/> [dostęp 15.06.2014]

- Projekt PPCG (Polish Python Coders Group) Stowarzyszenia Polska Grupa Użytkowników Pythona.* <http://pl.python.org/> [dostęp 15.06.2014]
- Rycharski Ł. *Zwinne programowanie, czyli o metodologii Agile (30.08.2010)*
http://www.computerworld.pl/news/359379_1/Zwinne.programowanie.czyli.o.metodologii.Agile.html [dostęp 15.06.2014]
- SAGE Components.* <http://www.sagemath.org/links-components.html/> [dostęp 15.06.2014]
- TI: Programowanie z Pythonem. Podręcznik dla studentów I roku neuroinformatyki i fizyki medycznej na Wydziale Fizyki Uniwersytetu Warszawskiego.*
https://brain.fuw.edu.pl/edu/TI:Programowanie_z_Pythonem [dostęp 15.06.2014]
- Topolewicz S.: *IPython. Alternatywna powłoka interaktywna Pythona.* „Linux Magazine” 2012(a), nr 9 (103), s. 36-37
- Topolewicz S.: *Prezentacja wybranych zagadnień fizycznych w programie SAGE (plakat aplikacyjny).* 24. Ogólnopolskie Sympozjum Naukowe „CZŁOWIEK - MEDIA - EDUKACJA”. Kraków, 26-27 września 2014
- Topolewicz S.: *Sage. Matematyczny mędrzec.* „Linux Magazine” 2012(b), nr 11 (105), s. 42-47
- Topolewicz S.: *Sphinx. Idealna dokumentacja oprogramowania.* „Linux Magazine” 2012(c), nr 11 (105), s. 40-41
- What is open source?* <http://opensource.com/resources/what-open-source/> [dostęp 15.06.2014]
- Zanurkuj w Pythonie.* http://pl.wikibooks.org/wiki/Zanurkuj_w_Pythonie/ [dostęp 15.06.2014]