

# Programowanie obiektowe - wykład 1

Roman Putanowicz

Katedra Technologii Informatycznych w Inżynierii

2 października 2019



- o przedmiocie
- o programowaniu obiektowym
- o Pythonie
- o narzędziach
- o kontekście dla projektów
- proste przykłady

## Cele:

- Zapoznanie studentów z podstawowymi koncepcjami programowania obiektowego.
- Zapoznanie studentów z obiektowo zorientowanym językiem programowania wykorzystywanym do budowy i rozszerzania naukowych i inżynierskich środowisk obliczeniowych

## Efekty:

- Znajomość klasyfikacji języków programowania. Znajomość charakterystyki obiektowych języków programowania.
- Znajomość składni języka Python w zakresie umożliwiającym programowanie obiektowe.
- Umiejętność analizy problemów z punktu widzenia analizy obiektowej. Umiejętność wyodrębniania obiektów i opisywania relacji między obiektami.
- Umiejętność tworzenia, uruchamiania i debugowania programów w języku Python.



## Wymagania:

- Umiejętność programowania strukturalnego w dowolnym języku programowania.

## Wykłady:

- Tematyka wykładów choć w zarysie naszkicowana, jest otwarta. Oczekuję propozycji zagadnień do poruszenia na wykładach czy też konkretnych pytań, na które Państwo chcielibyście usłyszeć odpowiedź.

## Laboratoria:

- Laboratoria mają służyć przede wszystkim wyjaśnianiu konkretnych problemów z Państwa programami przy monitorze. Są okazją do pokazania Państwa **umiejętności i zaangażowania**. Podstawowy sposób pracy do **dyskusja**. Nad projektami możecie Państwo pracować w parach.



		Zaangażowanie		
		małe	średnie	duże
Efekty	małe	2.0	3	3.5
	średnie	3.5	3.5	4
	duże	4	4.5	5

- Generalnie dzielę studentów na dwie kategorie: ci którzy chcą się czegoś nauczyć i pozostałych. Reszta to mniej istotne szczegóły.
- Trafność i obiektywność oceniania zależy w gruncie rzeczy od ilości informacji na temat efektów i zaangażowania w pracę, którą mi Państwo dostarczycie.
- **Zawsze można zawsze przyjść i podyskutować na temat konkretnej oceny.**

- Rzut ukośny w polu grawitacyjnym z oporami ruchu i bez, wersja 2D i 3D.
- Transformacje afiniczne siatek.
- Obliczanie charakterystyk geometrycznych figur płaskich i brył.
- Numeryczne całkowanie funkcji dwu zmiennych na siatkach.
- Generacja modelu geometrycznego obszaru ze sferycznymi pustkami (2D/3D).
- Analiza statyczna ramy płaskiej.

## Podstawowa

- Tony Gaddis, Python dla zupełnie początkujących, 2019, Helion S.A.
- Mark Lutz, Python, Wprowadzenie, 2011, Helion S.A.

## Dodatkowa:

- Michał Jaworowski, Tarek Ziade, Profesjonalne programowanie w Pythonie. Poziom ekspert, 2017, Helion S.A.
- Gilles Dowek, Principles of Programming Languages, 2009, Springer *Dostępne w Bibliotece PK - zasoby elektroniczne*
- Kent D. Lee, Foundations of Programming Languages, 2014, Springer *Dostępne w Bibliotece PK - zasoby elektroniczne*
- Iain Graig, The Interpretation of Object Oriented Programming Languages, 2002, Springer *Dostępne w Bibliotece PK - zasoby elektroniczne*



# Paradygmaty programowania

Programowanie:

- **Imperatywne** : C, C++, Java, Kotlin, PHP, Python, Ruby, Wolfram Language
- **Strukturalne** : C, C++, Java, Kotlin, Pascal, PHP, Python, Wolfram Language
- **Proceduralne** : C, C++, Java, Kotlin, Pascal, PHP, Python, Wolfram Language
- **Funkcyjne** : C++, Clojure, Elixir, Erlang, F#, Haskell, Java (od wersji 8), Kotlin, Lisp, Python, JavaScript
- **Zdarzeniowe** : JavaScript, ActionScript, Visual Basic, Elm
- **Obiektowe** : Common Lisp, C++, C#, Eiffel, Java, Kotlin, PHP, Python, Ruby, Scala, JavaScript
- **Deklaratywne** : SQL, regular expressions, Prolog, OWL, SPARQL, Prolog

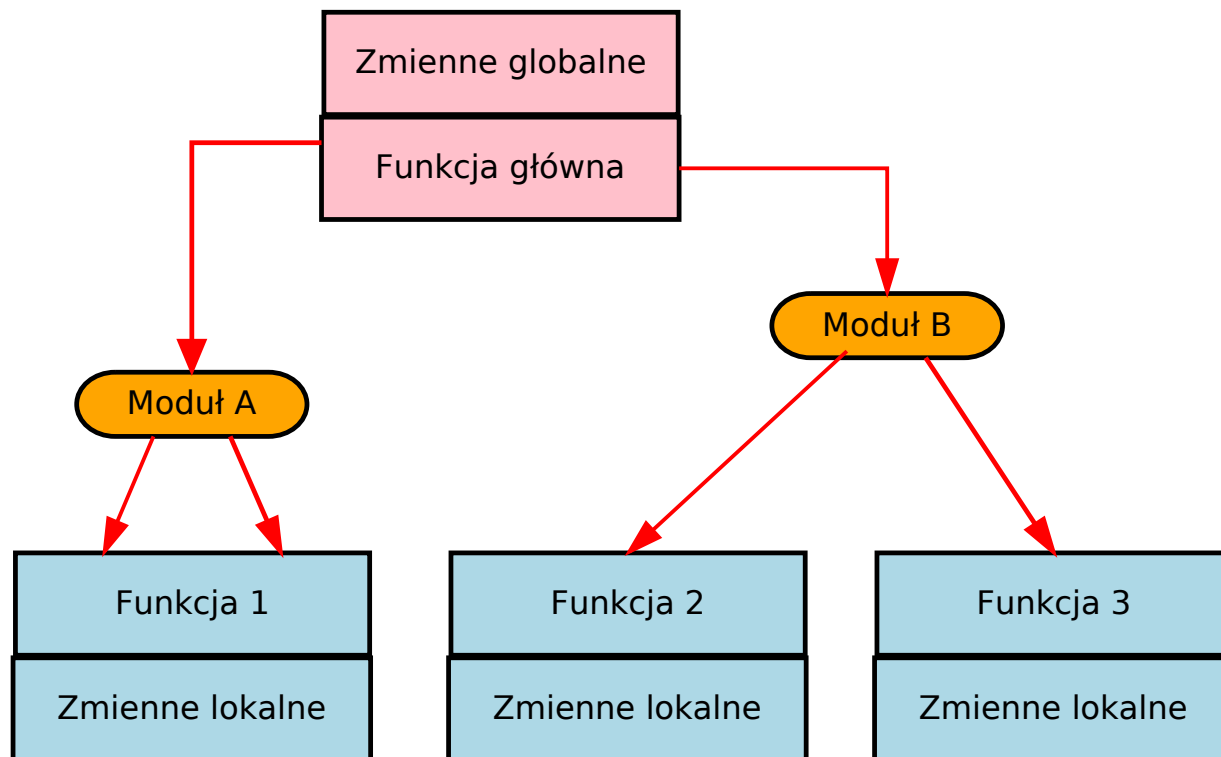
Więcej szczegółów :

[https://en.wikipedia.org/wiki/Comparison\\_of\\_programming\\_paradigms](https://en.wikipedia.org/wiki/Comparison_of_programming_paradigms)

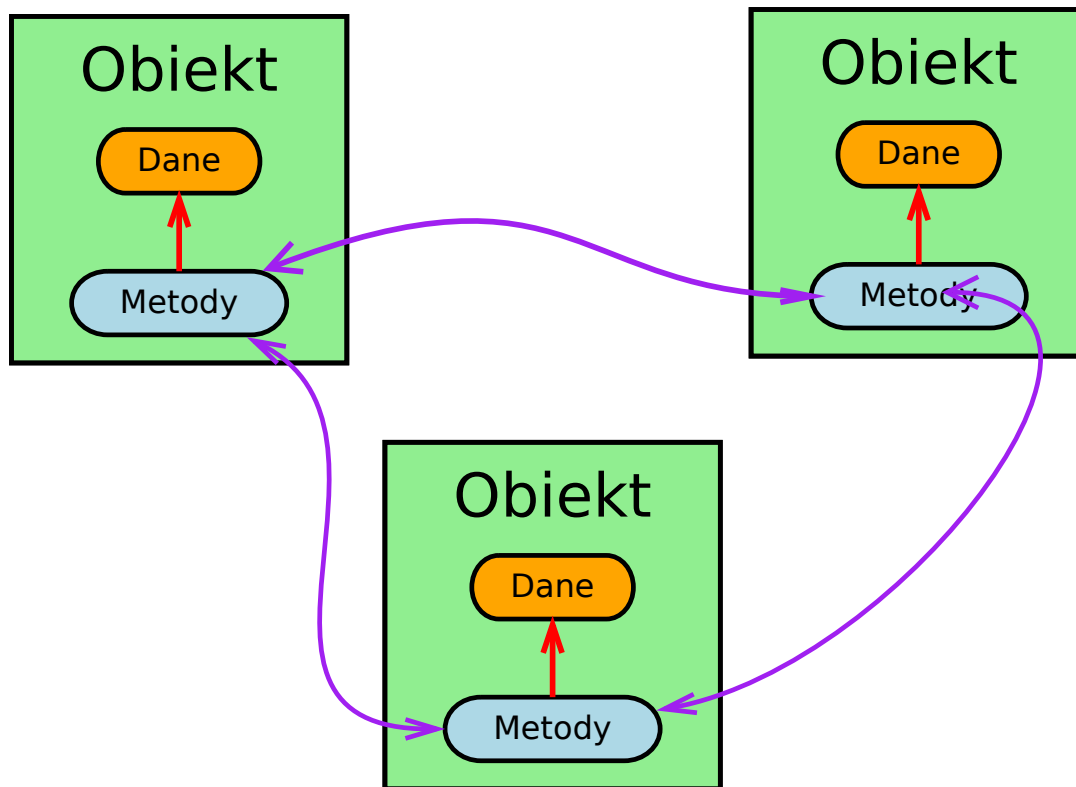




# Programowanie strukturalne



# Programowanie obiektowe



# Jakie znaczenia ma wybór języka

- Który język jest “najlepszy”?
- Domain Specific Languages (DSL)
- “Hot language” : Python

Można zadać pytanie jaki jest wpływ wyboru języka programowania na efektywność prowadzenia badań naukowych (lub dydaktyki)? Pytanie wydaje się proste ale odpowiedź jest bardzo skomplikowana.



# “Konkurs piękności” języków

TIOBE Index : <https://www.tiobe.com/tiobe-index/>

Sep 2019	Sep 2018	Change	Language	Ratings	Change
1	1		Java	16.661%	-0.78%
2	2		C	15.205%	-0.24%
3	3		Python	9.874%	+2.22%
4	4		C++	5.635%	-1.76%
5	6	change	C#	3.399%	+0.10%
18	13	change	MATLAB	1.062%	-0.21%
34			Fortran	0.359%	
35			Ada	0.346%	
36			Julia	0.338%	
37			Kotlin	0.337%	

Latami: Fortran (1957/1977), C(1972), C++ (1980), Matlab(1984), Wolfram (1988), Python (1991), Fortran90 (1992), R (1993), Java (1995), C# (2001), Scala (2003), Kotlin (2011), Julia (2012)



# Domain Specific Languages (DSL)

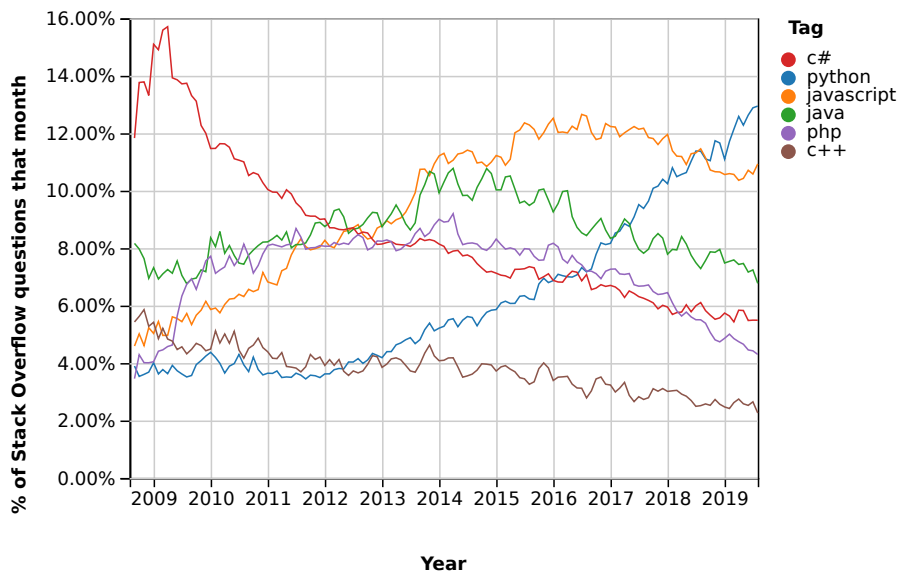
- Języki do modelowania geometrycznego : e.g. GMSH
- Języki dla MES: FreeFem, Fiac(FENics), Unified Form Language(FENics), GetFEM high level assembly
- Języki dla operacji na tensorach: packages for Mathematica, Maple, Maxima, Matlab, R, Yoric,
- Języki do programowania grafiki: Asymptote, tikz(LaTeX)
- Języki programowania wizualnego : Dynamo, Grashooper



- Fortran: 77, 95, 2008
- C++: 1998, C++03, C++11, C++14, C++17, C++20 (wkrótce)
- Python: Python2, Python3 (od 2008)
- Matlab: 2008 - znaczące rozszerzenia obiektowe

# “Hot” language: Python

From: Stack Overflow Trends



*Now, the Developer Survey 2019 reveals that Python has “edged out Java” and is the second most loved language. Stack Overflow refers to Python as the “fastest-growing major programming language”.*

<https://jaxenter.com/stack-overflow-dev-survey-2019-157815.html>



# Python: podstawowe informacje

[https://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))  
Python - interpretowany język wysokiego poziomu, ogólnego przeznaczenia. Zaprojektowany przez Guido van Rossuma i po raz pierwszy opublikowany w 1991. Umożliwia programowanie z wykorzystaniem różnych paradygmatów programowania, w szczególności programowanie obiektowe. W roku 2008 została opublikowana wersja 3.0 Pythona, która jest niekompatybilna z wersją 2. Wersja 2.7 jest ostatnią z wersji 2 i nie będzie dalej rozwijana.

```
1     def hello_world():
2         print('Hello world')
3
4     if __name__ == '__main__':
5         hello_world()
6
```





# Implementacje Pythona

- CPython - podstawowa implementacja w języku C
- Stackless Python - implementacja w C (unika korzystania ze stosu systemowego)
- Jython - implementacja w Javie
- IronPython - implementacja dla środowiska .NET Framework
- PyPy - implementacja Pythona w Pythonie



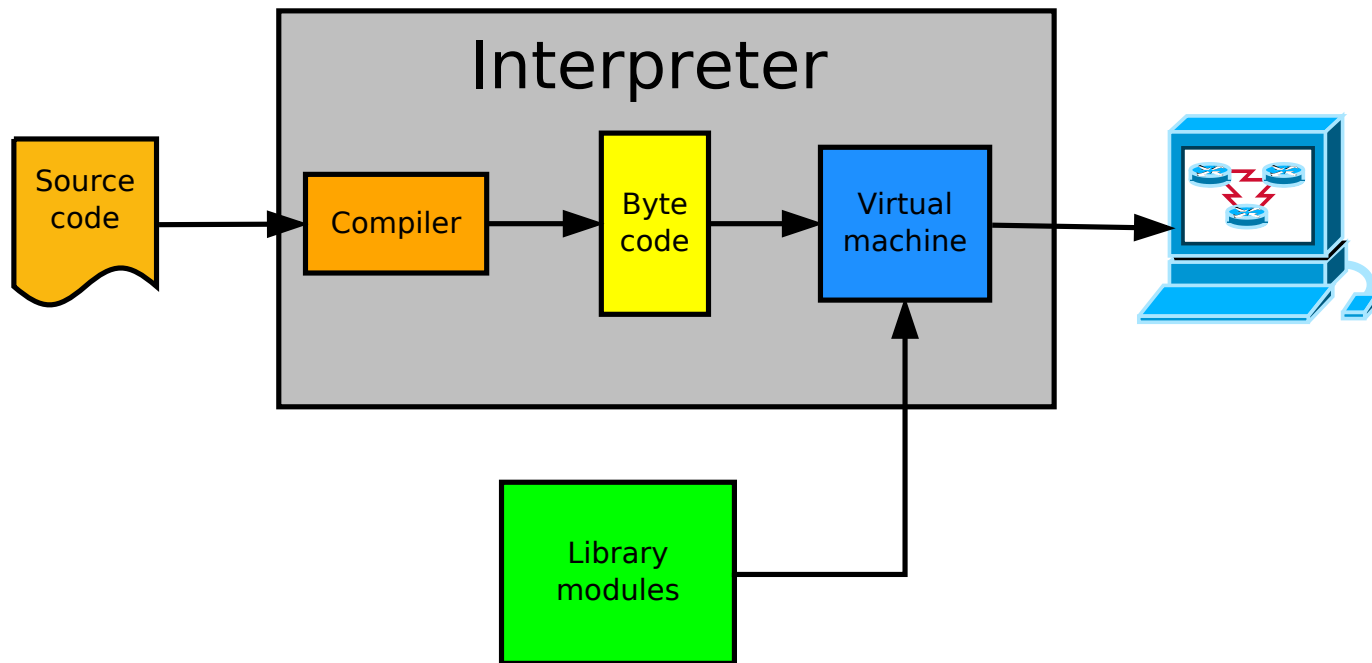
# Organizacja kodu: pakiety i moduły

- Dla prostych programów cały kod w jednym skrypcie.
- Złożone programy podzielone są na pakiety i moduły:
  - moduł - plik \*.py zawierający kod
  - pakiet - folder zawierający minimum plik `__init__.py` oraz jakieś moduły lub inne pakiety.

Korzystanie z zawartości modułów i pakietów wymaga ich “importowania”, które jest realizowane za pomocą instrukcji `import`.



# Sposób wykonywanie programów Pythona



Rysunek na podstawie <https://indianpythonista.wordpress.com/2018/01/04/how-python-runs/>

<https://indianpythonista.wordpress.com/2018/01/04/how-python-runs/>



# “Prosty” program

```
1      # -*- coding: utf-8 -*-
2      """
3      Calculate distance between two points with coordinates
4      read from file my_points.txt
5      """
6      import math
7
8      def read_points(filename):
9          with open('my_points.txt') as input_file:
10             points = []
11             for line in input_file:
12                 points.append(list(map(float, line.split()))
13                                ))
14             return points
15
16     def segment_length(pt1, pt2):
17         length = 0
18         for c1, c2 in zip(pt1, pt2):
19             length += (c1-c2)**2
20         return math.sqrt(length)
21
22     if __name__ == '__main__':
23         pts = read_points('my_points.txt')
24         if len(pts) == 2:
25             distance = segment_length(pts[0], pts[1])
26             print(f"Distance between points : {distance}")
27         else:
28             print(f"Error: expected 2 points got {len(pts)}")
```



- Oficjalna dystrybucja CPython
- Anaconda (Continuum Analytics)
- Canopy (Enthought)
- ActivePython (ActiveState)

# Zen of Python in Easter Egg

```
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than *right* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!
```



Absolutne minimum: interpreter Pythona + edytor

Efektywne programowanie wymaga dodatkowych narzędzi:

- „inteligentne” zintegrowane środowisko tworzenia programów (ang. IDE - Integrated Development Environment), np. **PyCharm**
- system kontroli wersji: np. **Git**, **Subversion** + webowy serwis, np. **GitHub**, **BitBucket**
- narzędzia do tworzenia dokumentacji, np. **Sphinx**, **pandoc**, **LaTeX**
- powłoka: np. **Bash**, **PowerShell**

Dodatkowo potrzebujemy narzędzi specyficznych dla obszaru zastosowań:

- modelowanie geometryczne i generacja siatek
- wizualizacja danych

The screenshot displays the PyCharm IDE interface. The main editor window shows a Python script named `demo_translate_and_scale.py` with the following code:

```
15
16 PKGDIR = os.path.dirname(femcalc.__file__)
17 DATADIR = os.path.join(PKGDIR, 'test', 'data')
18
19
20 class DemoEdgeBoundingBox(examples.Demo):
21     def demonstrate(self):
22         master_box = meshgrid.geometry.mrg.BoundingBox([0, 0, 10, 10])
23         for edge in meshgrid.geometry.mrg.BoundingBox.edges:
24             edge_box = master_box.getEdgeBox(edge, eps=1.0)
25             print(f"{edge} -> {edge_box.lbf} : {edge_box.hbf}")
26
27 if __name__ == '__main__':
28     demo = DemoEdgeBoundingBox()
29     demo.run()
```

The SciView window on the right shows a 2D plot with a grid. A red rectangle highlights a 3x3 green grid in the bottom-left corner, and a yellow 4x4 grid is visible in the top-right corner. The plot axes range from 0.0 to 17.5 on the y-axis and 0 to 15 on the x-axis.

The Run console at the bottom shows the message: "Process finished with exit code 0".

<https://www.jetbrains.com/pycharm/>





The screenshot displays the Spyder Python IDE interface. The editor window shows a Python script for creating a meshgrid. The IPython console shows the execution output, including the IPython version and the start of an interactive session.

```
1# -*- coding: utf-8 -*-
2# Python packages
3import numpy as np
4
5
6""" FOR CLASS ROOM *****
7import sys
8#sys.path.insert(0, "C:\\Users\\magoulesf\\Desktop\\")
9#from pytransform import pyarmor_runtime
10#pyarmor_runtime()
11""" *****
12
13
14# MRG packages
15from femcalc import meshgrid
16import femcalc.meshgrid.graphics
17
18
19def simpleMesh():
20    """
21
22
23
24
25
26
27
28
29
30
31
32    out_mesh = meshgrid._meshrg.Mesh()
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
Python 3.6.7 (default, Jul 2 2019, 02:21:41) [MSC v.1900 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: |
```

<https://www.spyder-ide.org/>



System kontroli wersji (ang. revision control system) to narzędzie bez którego nie zapewne nie powstałby żaden poważny projekt oprogramowania Git (<https://git-scm.com/>) to rozproszony system kontroli wersji, obecnie jedno z najpopularniejszych narzędzi w swojej kategorii.

Minimalny zestaw poleceń Git'a: `clone`, `add`, `commit`, `status`, `diff`, `pull`, `push`

Git'a najlepiej nauczyć się obsługiwać z poziomu wiersza poleceń a potem dopiero używać jakiegoś GUI.



Jeżeli korzystamy z dystrybucji Anaconda:

```
conda install nazwa_pakietu
```

Jeżeli korzystamy z **pip**:

```
python -m pip install nazwa_pakietu  
pip install nazwa_pakietu
```

Generalnie zaleca się nie mieszać **conda** z **pip**. Jeżeli już to należy zainstalować wszystkie potrzebne pakiety używając **conda** a dopiero gdy czegoś brakuje można użyć **pip**. Zaleca się też korzystać z dobrodziejstw wirtualnych środowisk (**virtualenv** - temat na później).

# Dokumentowanie kodu

- <https://numpydoc.readthedocs.io/en/latest/format.html>
- [https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example\\_numpy.html](https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_numpy.html)

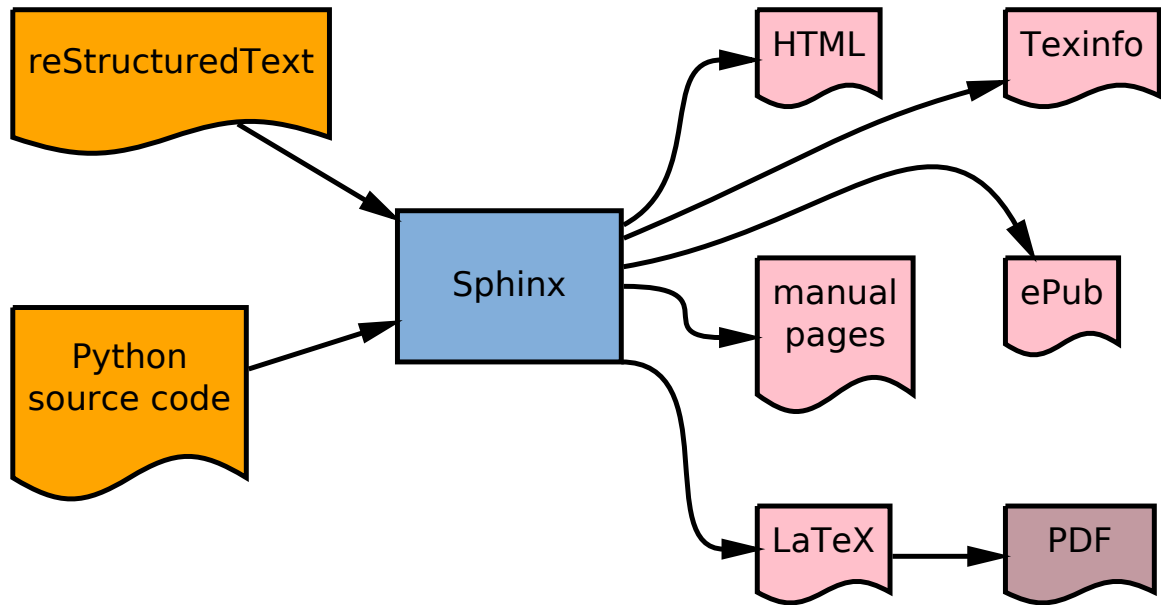
Każdy pakiet, moduł, klasa, funkcja powinny być opatrzone notką dokumentacyjną (ang. docstring) (poniżej w konwencji pakietu Numpy)

```
1  def translate(points, displacement):
2      """Translate points.
3
4      Parameters
5      -----
6      points : array
7          Array o points (one point per row).
8      displacement : vector
9          The translation vector
10
11     Returns
12     -----
13     array
14         Array of translated points.
15     """
16     return points + displacement
```



# Sphinx

Sphinx <http://www.sphinx-doc.org> - to system tworzenia dokumentów w różnych formatach ze źródeł w formacie reStructuredText. Pozwala na automatyzację tworzenia wysokiej jakości dokumentacji bezpośrednio z notek informacyjnych i elementów składni Pythona.



**reStructuredText** to język znaczników przeznaczony do szybkiego tworzenia dokumentacji technicznej. Jest wykorzystywany w:

- Docutils
- Sphinx
- Trac

W połączeniu z LaTeX'em jest wykorzystywany do tworzenia zaawansowanej dokumentacji technicznej (LaTeX - zwłaszcza do zaawansowanego składania tekstów, np. matematycznych)

```
1. Chapter one
```

```
=====
```

- \* the first list element
- \* the second **important** element
- \* the third with some formula :math:'x^2'



L<sup>A</sup>T<sub>E</sub>X (wymowa latech lub lejtech) to język do formatowania dokumentów tekstowych, oraz związane z nim oprogramowanie pozwalające na renderowanie tych dokumentów w formatach dla tradycyjnego druku lub elektronicznego rozpowszechniania.

Jaki jest związek z LaTeX'a z programowaniem obiektyw i Pythonem? Pierwsza odpowiedź to tworzenie dokumentacji. Druga, to że LaTeX podobnie jak Python to świetne narzędzie a lepiej jest znać dwa świetne narzędzia niż jedno. Wreszcie praca z Pythonem i LaTeX'em jest powszechnie zalecana jako środek zapobiegawczy atrofii umysłowej.

Edytory: Kile, Texmaker, TeXworks, TeXnicCenter  
Dystrybucje (dla Windows): MikTeX'



Python nie ma żadnych mechanizmów grafiki wbudowanych w język ale dostarcza znakomite pakiety to programowania grafiki:

- Matplotlib - <https://matplotlib.org/>
- Pillow - fork of PIL (Python Imaging Library) for Python 3
- scikit-image - <https://scikit-image.org/>

Matplotlib będzie naszym podstawowym narzędziem do wizualizacji danych i algorytmów. **UWAGA:** Matplotlib dostarcza dwu interfejsów a) obiektowego i b) wzorowanego na funkcjach Matlaba. Warto do początku zacząć używać interfejsu obiektowego.



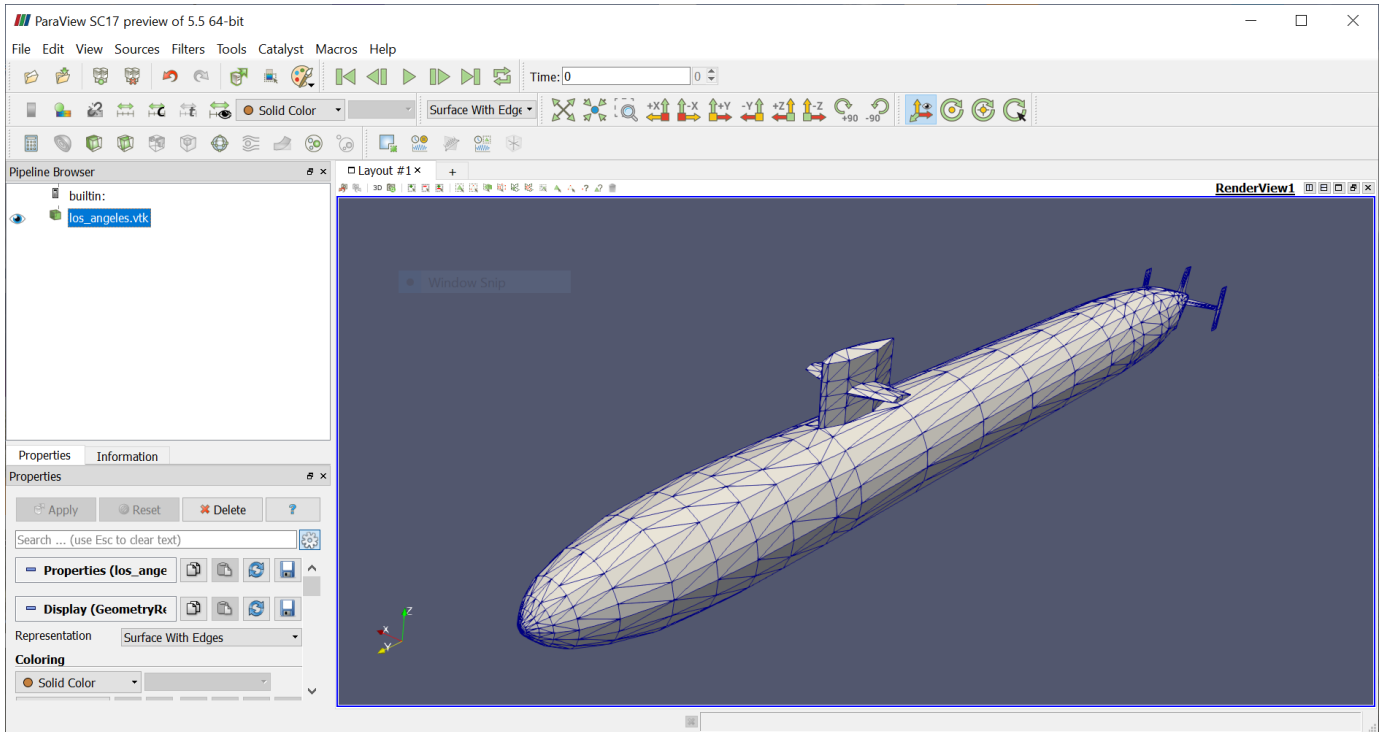


Jeżeli mamy jakiś problem z zakresu numeryki (generalnie jakiś problem) to z 98% pewnością został on już rozwiązany w Pythonie. Trzeba tylko znaleźć odpowiedni moduł. W przypadku numeryki szukamy w pakietach:

- NumPy - <https://numpy.org/>
- SciPy - <https://www.scipy.org/>
- Sympy - <https://www.sympy.org>

Jak zawsze pomaga nam Stack Overflow <https://stackoverflow.com> czyli społeczność programistów.





<https://www.paraview.org/>

ParaView można kontrolować i rozbudowywać poprzez skrypty Pythona.

# O co chodzi z tymi kontekstami?

Programowanie w zasadzie nie jest niczym innym jak tworzeniem precyzyjnego opisu rzeczy i procesów. Najłatwiej opisywać coś co się dobrze zna i rozumie, a jednocześnie coś co już jest dalece sformalizowane. Dodatkowo dobrze wybrać takie zagadnienia, które w miarę prosto jest przedstawić graficznie - ułatwi to ocenę czy program działa dobrze i jednocześnie samo zadanie będzie atrakcyjniejsze.

- Rozwiązywanie problemów początkowych dla układów równań różniczkowych zwyczajnych
- Operacje na siatkach obliczeniowych
- Całkowanie numeryczne
- Metoda Elementów Skończonych (konstrukcje ramowe)



- Modelowy problem - rzut ukośny w polu grawitacyjnym z uwzględnieniem oporów ruchu
- Wyprowadzenie równań ruchu
- Metody całkowania ODE

- Siatki strukturalne i niestukturalne
- Opis geometrii i “topologii” siatki
- Transformacje afiniczne siatek
- Operacje kombinatoryczne na siatkach

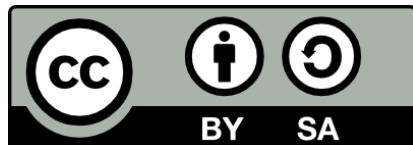
- Koncepcja elementu wzorcowego
- Opis geometrii elementu za pomocą funkcji kształtu
- Konstrukcja kwadratur w 2D

- Macierze elementowe
- Opis obciążeń i podparcia konstrukcji
- Algorytm agregacji równań algebraicznych MES

Dziękuję za uwagę







Copyright (CC-BY-SA) 2019 Roman Putanowicz  
Roman.Putanowicz@L5.pk.edu.pl

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/>.