Modeling Public Pensions with Mathematica and Python II

Brian Drawert, PhD

UC Santa Barbara & AppScale Systems, Inc

Sponsored by

Novim & Laura and John Arnold Foundation

Pension Calculation:

From Mathematica to the Cloud

1. Mathematica model to cloud app
2. Cloud Computing
3. Developing a Pension Calculator-as-a-service
4. Demo of the Pension calculator

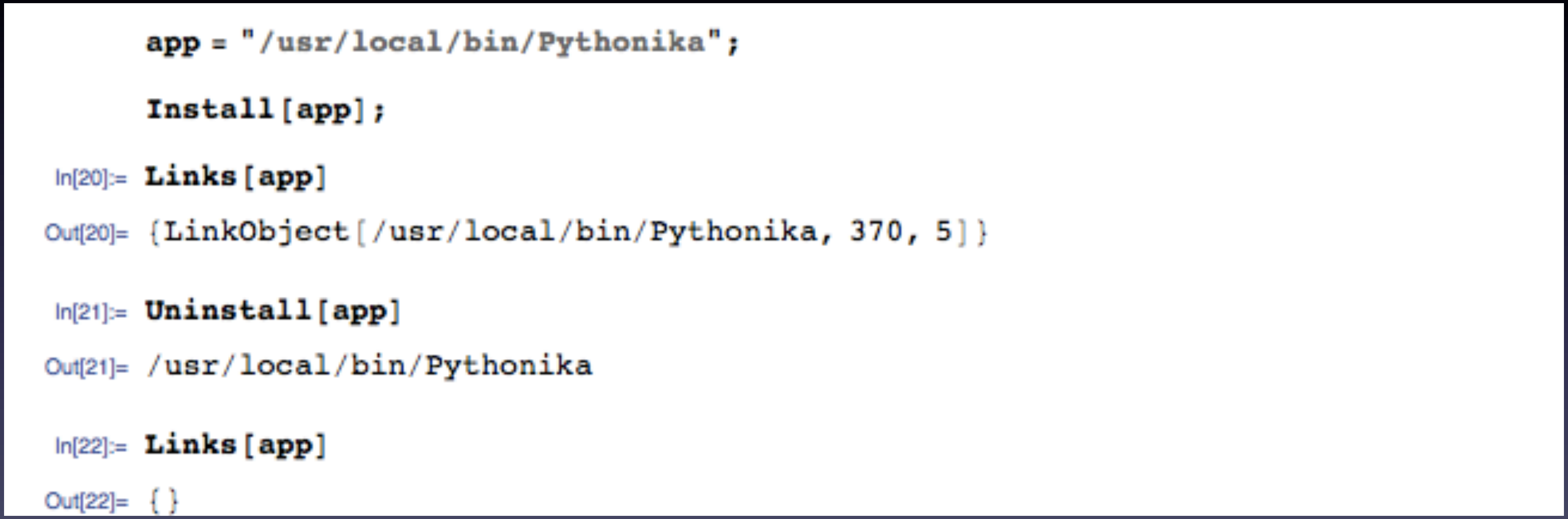
From Mathematica to the Cloud

- Publicly accessible platform
  - Web app built on Google App Engine / AppScale
  - Pension Computation model in Mathematica
- First attempt: Appscale to Web Mathematica
  - Limited ability to scale up / down
- Solution: rewrite engine in Python

Pythonika

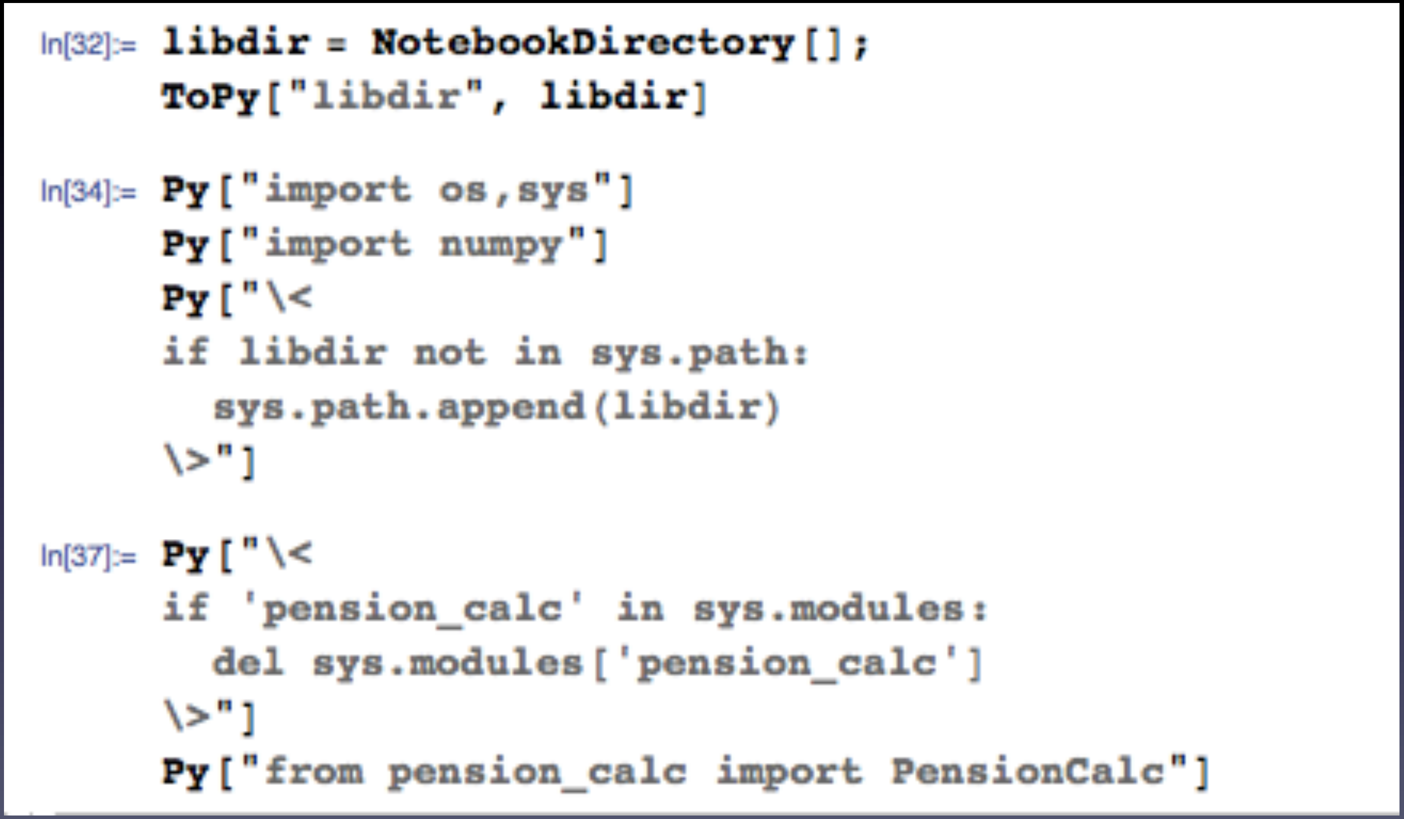
- Evaluate Python code from within a Mathematica Notebook
  - MathLink module
  - Automatically translates all basic data types
  - Define Mathematica functions with Python code
- Open Source
  - http://code.google.com/p/pythonika
  - http://github.com/briandrawert/pythonika

Connect to Pythonika



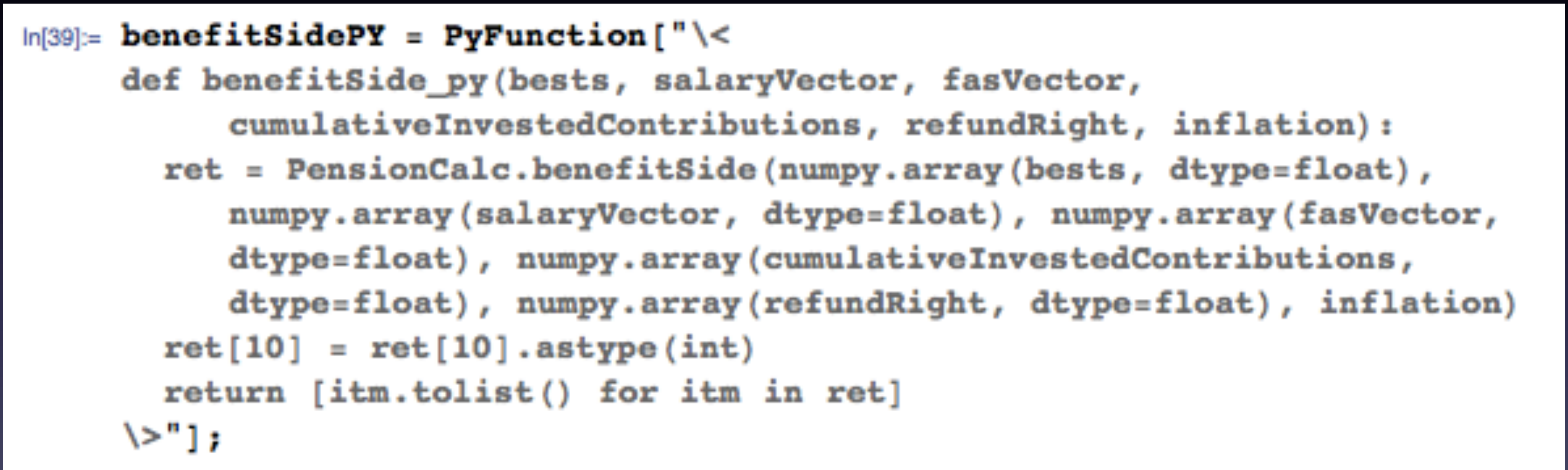
```
         app = "/usr/local/bin/Pythonika";

         Install[app];

In[20]:= Links[app]

Out[20]= {LinkObject[/usr/local/bin/Pythonika, 370, 5]}

In[21]:= Uninstall[app]

Out[21]= /usr/local/bin/Pythonika

In[22]:= Links[app]

Out[22]= {}
```

Setup Pythonika



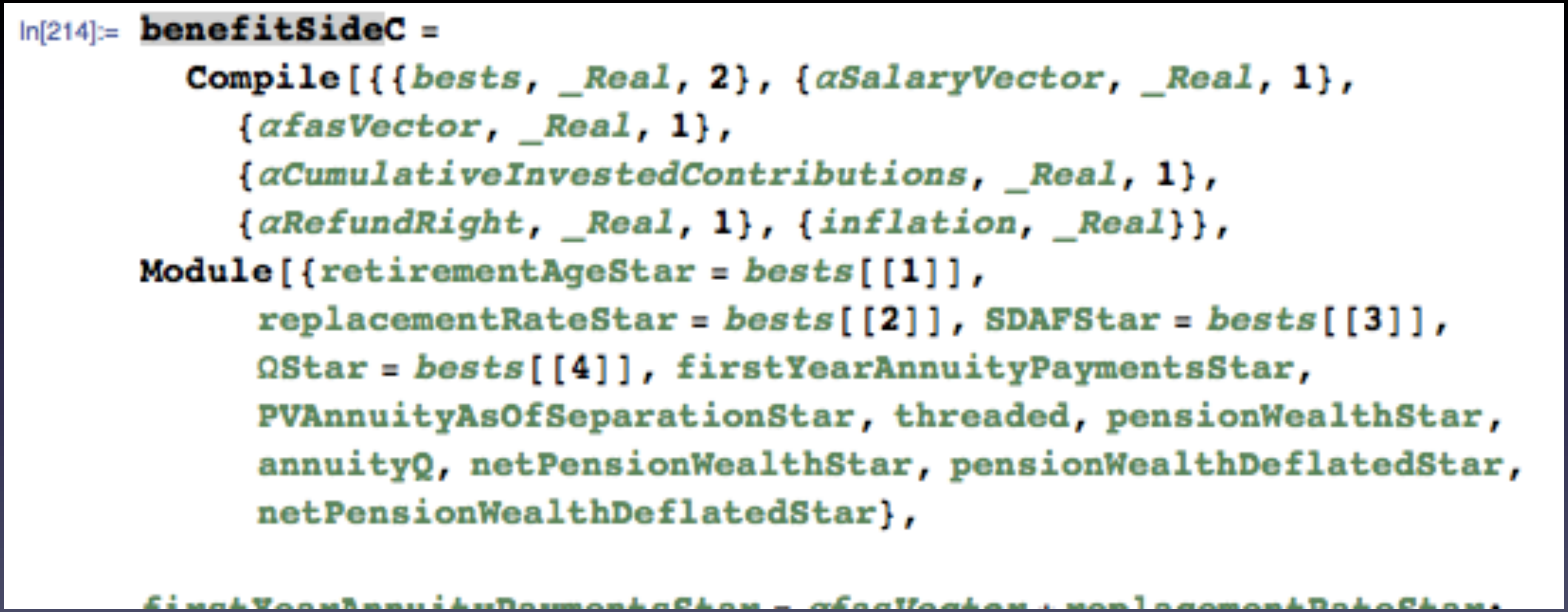
```
In[32]:= libdir = NotebookDirectory[];
         ToPy["libdir", libdir]

In[34]:= Py["import os,sys"]
         Py["import numpy"]
         Py["\<
         if libdir not in sys.path:
           sys.path.append(libdir)
         \>"]

In[37]:= Py["\<
         if 'pension_calc' in sys.modules:
           del sys.modules['pension_calc']
         \>"]
         Py["from pension_calc import PensionCalc"]
```

Pythonika function



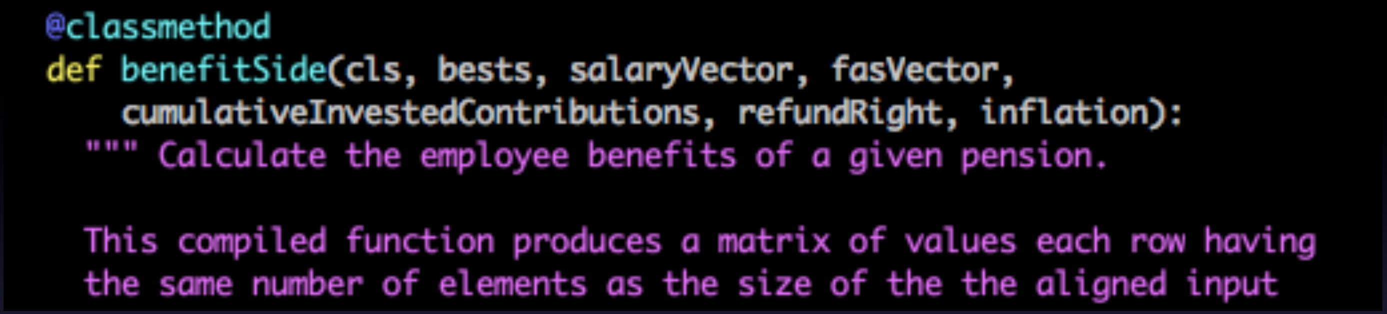
```
In[39]:= benefitSidePY = PyFunction["\<
         def benefitSide_py(bests, salaryVector, fasVector,
                 cumulativeInvestedContributions, refundRight, inflation):
           ret = PensionCalc.benefitSide(numpy.array(bests, dtype=float),
                 numpy.array(salaryVector, dtype=float), numpy.array(fasVector,
                 dtype=float), numpy.array(cumulativeInvestedContributions,
                 dtype=float), numpy.array(refundRight, dtype=float), inflation)
           ret[10] = ret[10].astype(int)
           return [itm.tolist() for itm in ret]
         \>"];
```

Mathematica function

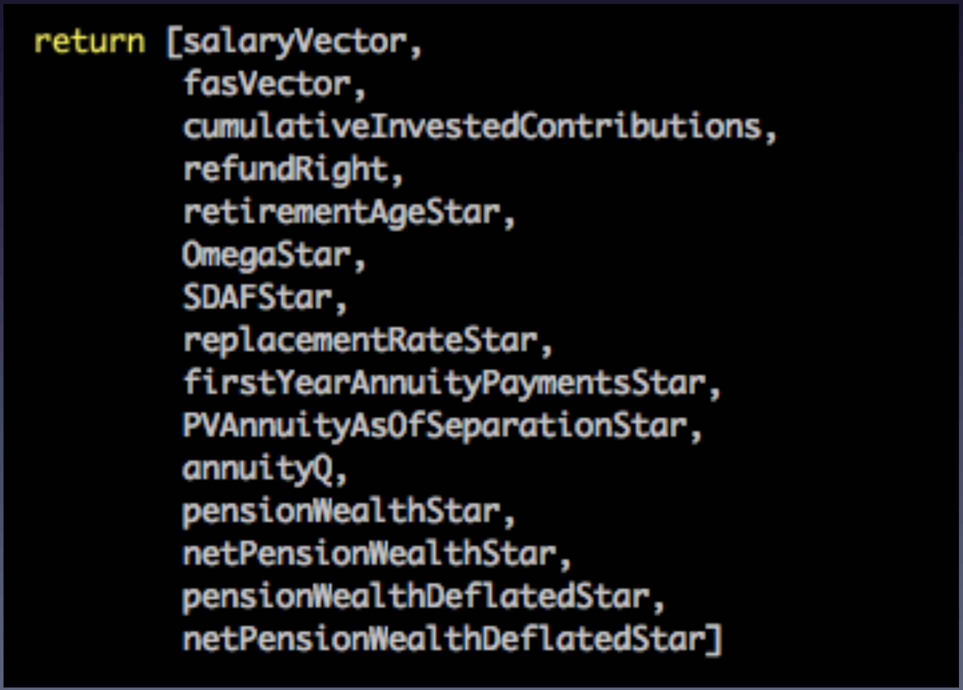


```
In[214]:= benefitSideC =
            Compile[{{bests, _Real, 2}, {αSalaryVector, _Real, 1},
              {αfasVector, _Real, 1},
              {αCumulativeInvestedContributions, _Real, 1},
              {αRefundRight, _Real, 1}, {inflation, _Real}},
          Module[{retirementAgeStar = bests[[1]],
               replacementRateStar = bests[[2]], SDAFStar = bests[[3]],
               ΩStar = bests[[4]], firstYearAnnuityPaymentsStar,
               PVAnnuityAsOfSeparationStar, threaded, pensionWealthStar,
               annuityQ, netPensionWealthStar, pensionWealthDeflatedStar,
               netPensionWealthDeflatedStar},

          firstYearAnnuityPaymentsStar = αfasVector * replacementRateStar;
```

Complex data structures preserved

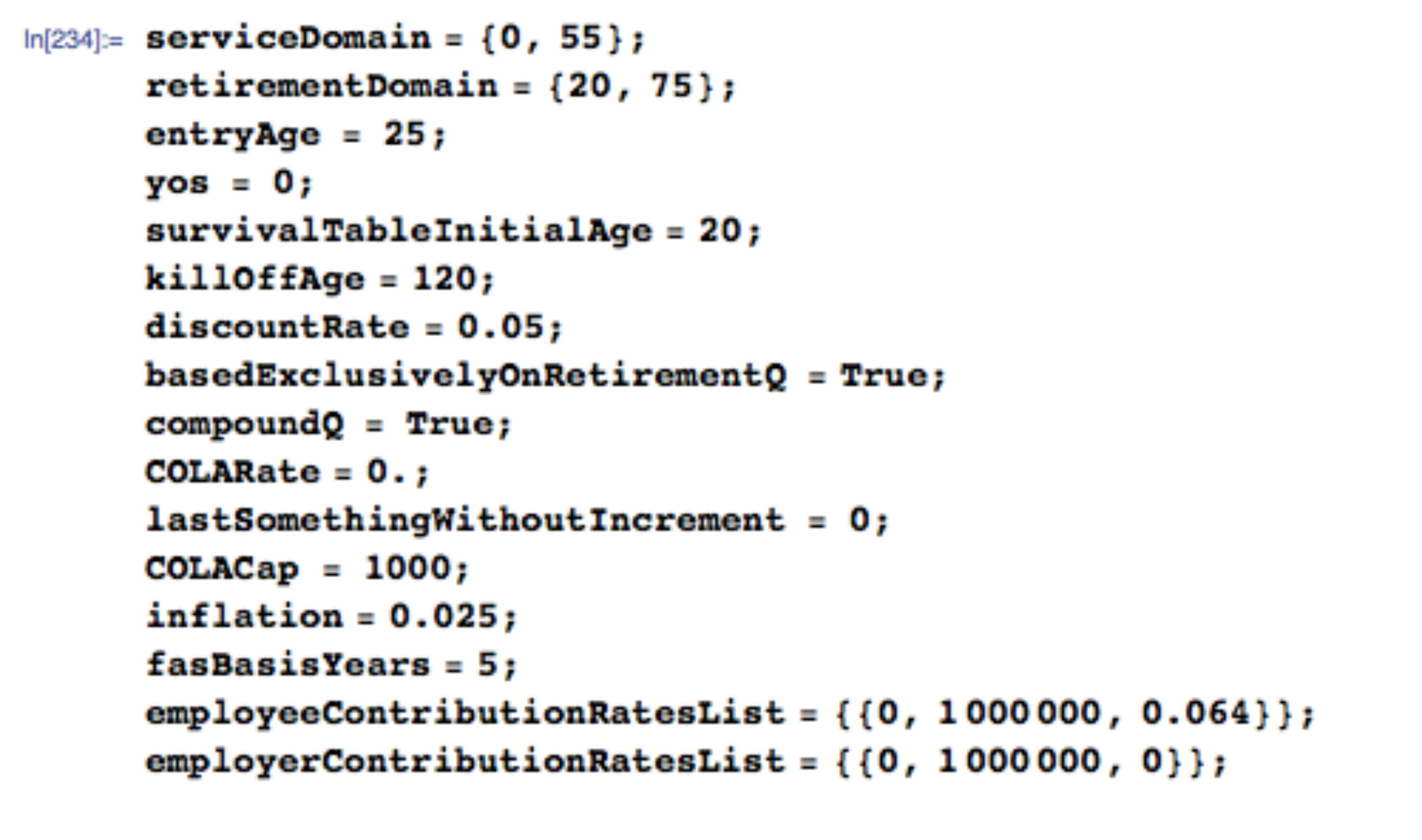


```python
@classmethod
def benefitSide(cls, bests, salaryVector, fasVector,
    cumulativeInvestedContributions, refundRight, inflation):
  """ Calculate the employee benefits of a given pension.

  This compiled function produces a matrix of values each row having
  the same number of elements as the size of the the aligned input
```



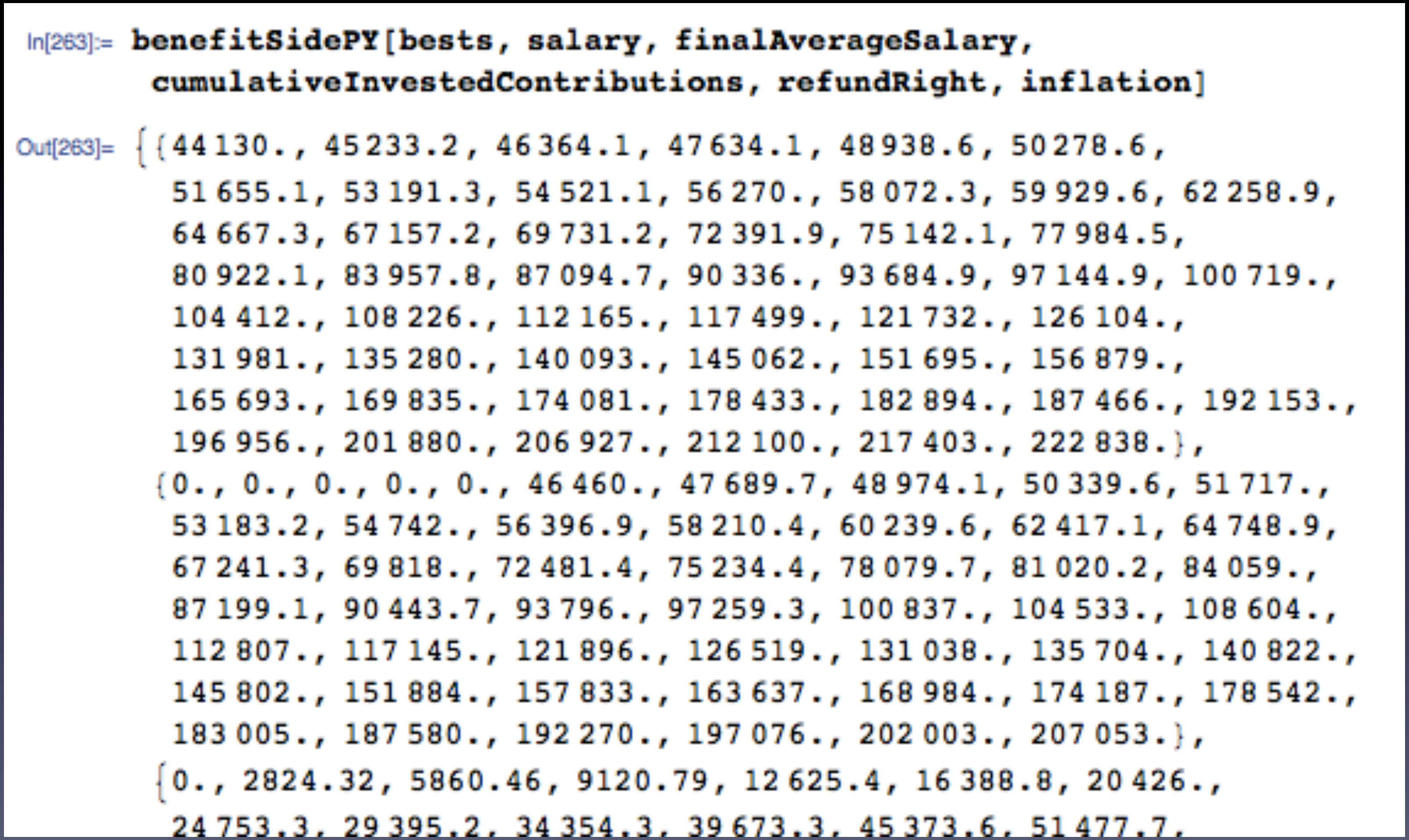
```python
return [salaryVector,
        fasVector,
        cumulativeInvestedContributions,
        refundRight,
        retirementAgeStar,
        OmegaStar,
        SDAFStar,
        replacementRateStar,
        firstYearAnnuityPaymentsStar,
        PVAnnuityAsOfSeparationStar,
        annuityQ,
        pensionWealthStar,
        netPensionWealthStar,
        pensionWealthDeflatedStar,
        netPensionWealthDeflatedStar]
```

Identical parameters



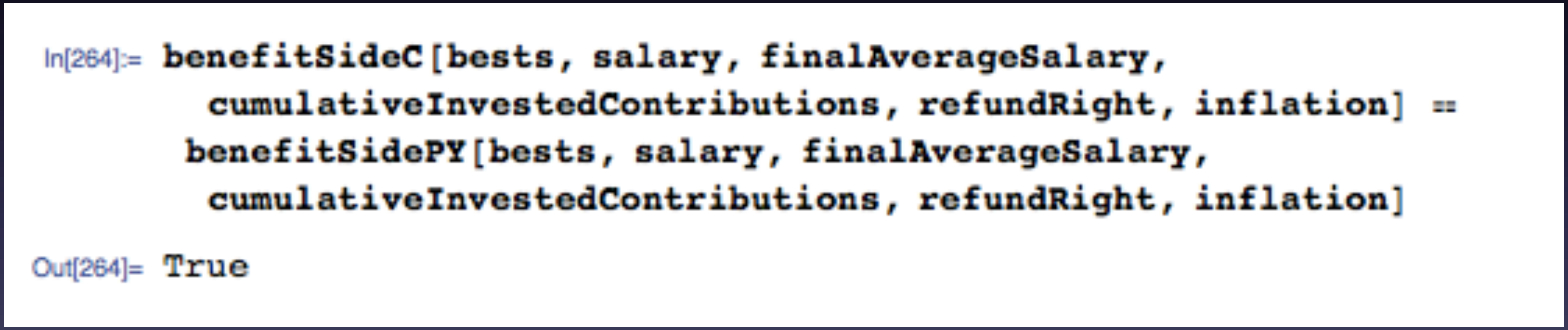
```
In[234]:= serviceDomain = {0, 55};
          retirementDomain = {20, 75};
          entryAge = 25;
          yos = 0;
          survivalTableInitialAge = 20;
          killOffAge = 120;
          discountRate = 0.05;
          basedExclusivelyOnRetirementQ = True;
          compoundQ = True;
          COLARate = 0.;
          lastSomethingWithoutIncrement = 0;
          COLACap = 1000;
          inflation = 0.025;
          fasBasisYears = 5;
          employeeContributionRatesList = {{0, 1000000, 0.064}};
          employerContributionRatesList = {{0, 1000000, 0}};
```

Run Python



```
In[263]:= benefitSidePY[bests, salary, finalAverageSalary,
           cumulativeInvestedContributions, refundRight, inflation]

Out[263]= {{44 130., 45 233.2, 46 364.1, 47 634.1, 48 938.6, 50 278.6,
            51 655.1, 53 191.3, 54 521.1, 56 270., 58 072.3, 59 929.6, 62 258.9,
            64 667.3, 67 157.2, 69 731.2, 72 391.9, 75 142.1, 77 984.5,
            80 922.1, 83 957.8, 87 094.7, 90 336., 93 684.9, 97 144.9, 100 719.,
            104 412., 108 226., 112 165., 117 499., 121 732., 126 104.,
            131 981., 135 280., 140 093., 145 062., 151 695., 156 879.,
            165 693., 169 835., 174 081., 178 433., 182 894., 187 466., 192 153.,
            196 956., 201 880., 206 927., 212 100., 217 403., 222 838.},
           {0., 0., 0., 0., 0., 46 460., 47 689.7, 48 974.1, 50 339.6, 51 717.,
            53 183.2, 54 742., 56 396.9, 58 210.4, 60 239.6, 62 417.1, 64 748.9,
            67 241.3, 69 818., 72 481.4, 75 234.4, 78 079.7, 81 020.2, 84 059.,
            87 199.1, 90 443.7, 93 796., 97 259.3, 100 837., 104 533., 108 604.,
            112 807., 117 145., 121 896., 126 519., 131 038., 135 704., 140 822.,
            145 802., 151 884., 157 833., 163 637., 168 984., 174 187., 178 542.,
            183 005., 187 580., 192 270., 197 076., 202 003., 207 053.},
           {0., 2824.32, 5860.46, 9120.79, 12 625.4, 16 388.8, 20 426.,
            24 753.3, 29 395.2, 34 354.3, 39 673.3, 45 373.6, 51 477.7,
```

Unit Test



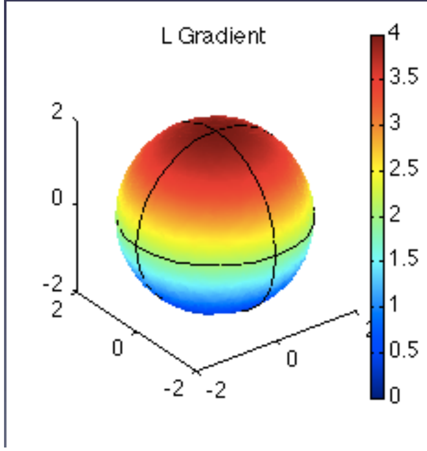
```
In[264]:= benefitSideC[bests, salary, finalAverageSalary,
            cumulativeInvestedContributions, refundRight, inflation] ==
          benefitSidePY[bests, salary, finalAverageSalary,
            cumulativeInvestedContributions, refundRight, inflation]

Out[264]= True
```

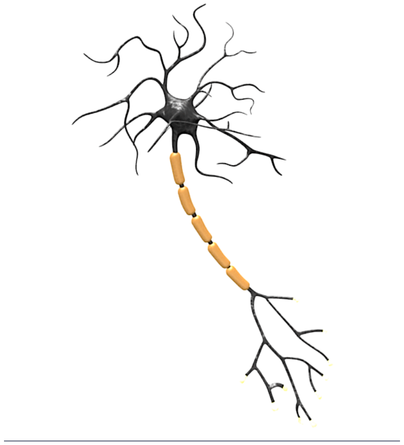
Robust Collaboration

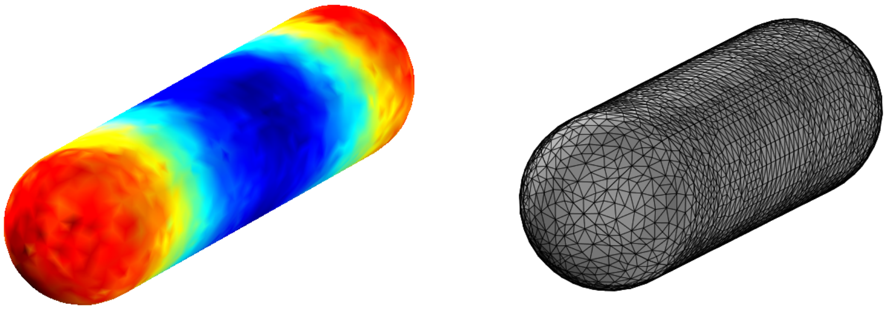
- Pythonika unit tests allowed our team to efficiently work together
  - Mathematica used to develop and validate models
  - Python used for the web app
  - Multiple development iterations, accounting for additional pension plans with new complexities
- 50+ functions converted
- 2000+ lines of code in the Python library

Other Advantages of Pythonika

- Enhance your Mathematica with Python specific software
  - PyURDME: Spatial stochastic simulation of bio-chemical systems







Drawert et al., BMC Systems Biology (2012)

Cloud Computing

- What is Cloud Computing?
  - Resources acquired on-demand and self-service
  - Resources are pooled across multiple customers
  - Rapid elasticity: scale up and scale down
  - Metered service: pay for what you use

Cloud Computing

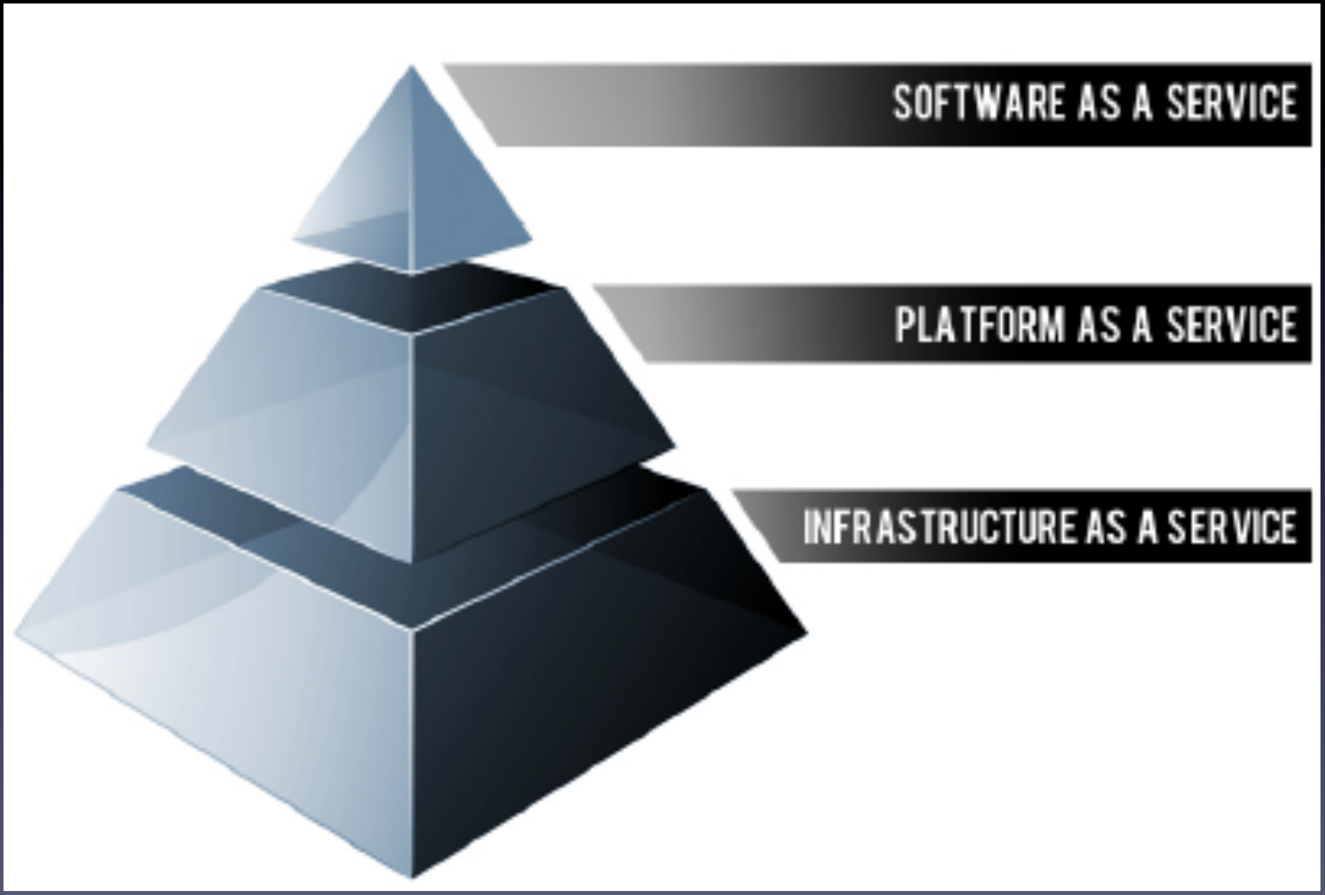


Image courtesy: Rackspace.com

IaaS : Infrastructure-as-a-service

- Cloud Computing infrastructure
  - Servers, storage, network and operating systems as an on-demand service
- Public cloud service providers
  - Amazon EC2, Microsoft Azure, Rackspace, Google Compute Engine
- Private cloud: IaaS on your own hardware
  - OpenStack, Eucalyptus, CloudStack

SaaS : Software-as-a-service

- On-demand software designed for end users
  - Delivered over the web
  - Centrally managed: no install, no upgrades
  - Metered: Subscriptions or pay-as-you-go
- Salesforce, Google Gmail/Docs, Adobe Creative Cloud, Microsoft Office 365

PaaS : Platform-as-a-service

- Platform for web application development
  - Rapid development and deployment
  - Abstracts away infrastructure complexity
  - Scalability, load balancing and failover
- Public: Google App Engine, Heroku, Microsoft Azure
- Private: AppScale, OpenShift

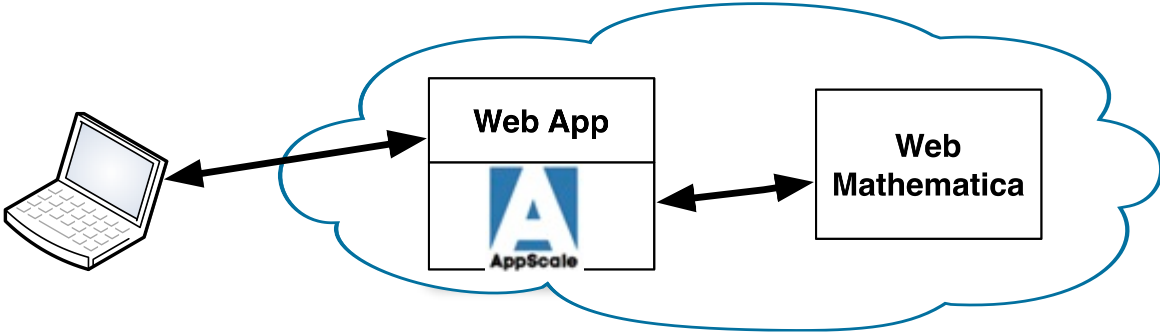
Google App Engine

- PaaS: run web apps on Google's cloud infrastructure
  - Pay for what you use: CPU, storage, bandwidth
  - Automatic scaling and load balancing
  - Many useful services
    - User auth, Data store, Background task queue
  - Languages: Python, Java (Go, PHP experimental)

AppScale

- Private PaaS: run GAE apps on any public/private cloud infrastructure
- Open Source: Can be customized
  - – Cython: optimized static compiler for Python

    Pension calculation: Python 27ms, Cython 15ms (COLAmatrix)
  - – Integrate with Web Mathematica



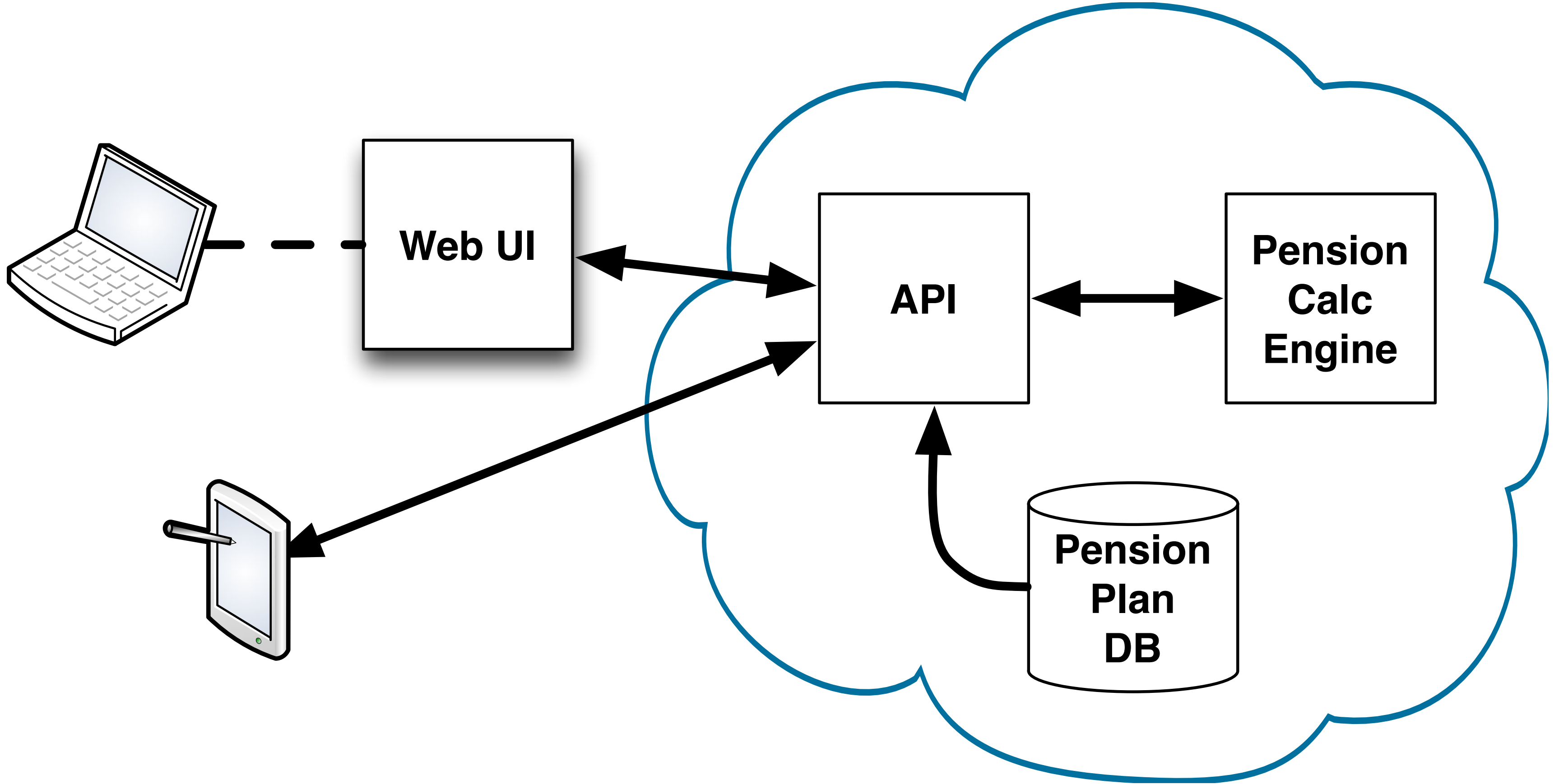
Pension Calculator as-a-service

- Service to enable plan members and plan managers to analyze plan costs and benefits.
- Democratizing the ability to analyze and evaluate complex pension issues
- Novim: non-profit, non-advocacy
  - Not taking sides, no value judgments
  - Providing tools for informed discourse

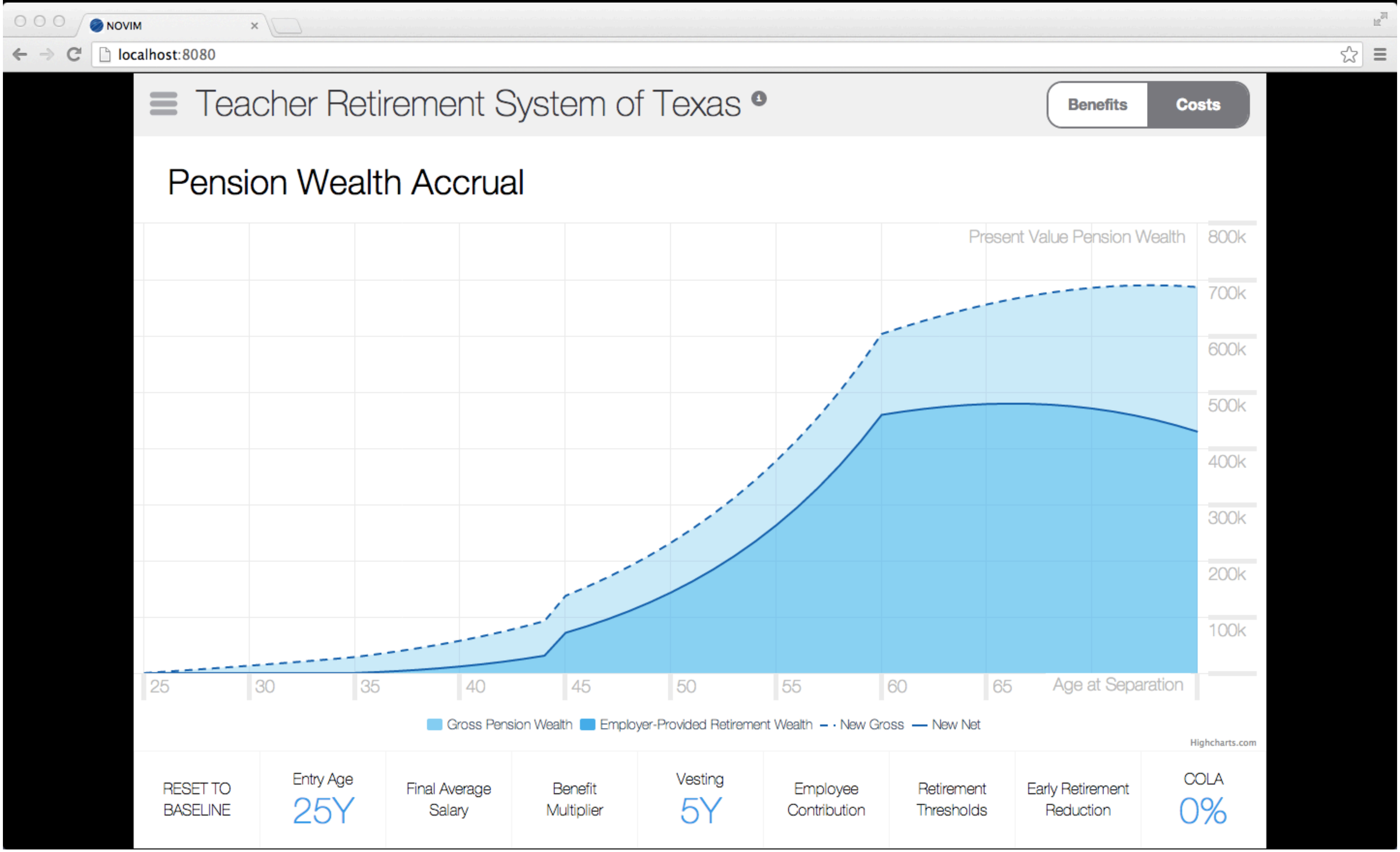
Pension Calculator as-a-service

- UI for intuitive manipulation of pension parameters
  - without overwhelming users with complexity
- Widely accessible web interface for public use
  - Mathematica package for researchers
  - Open API for researcher web development

Pension Calculator as-a-service



Pension Calc Web App



Teacher Retirement System of Texas

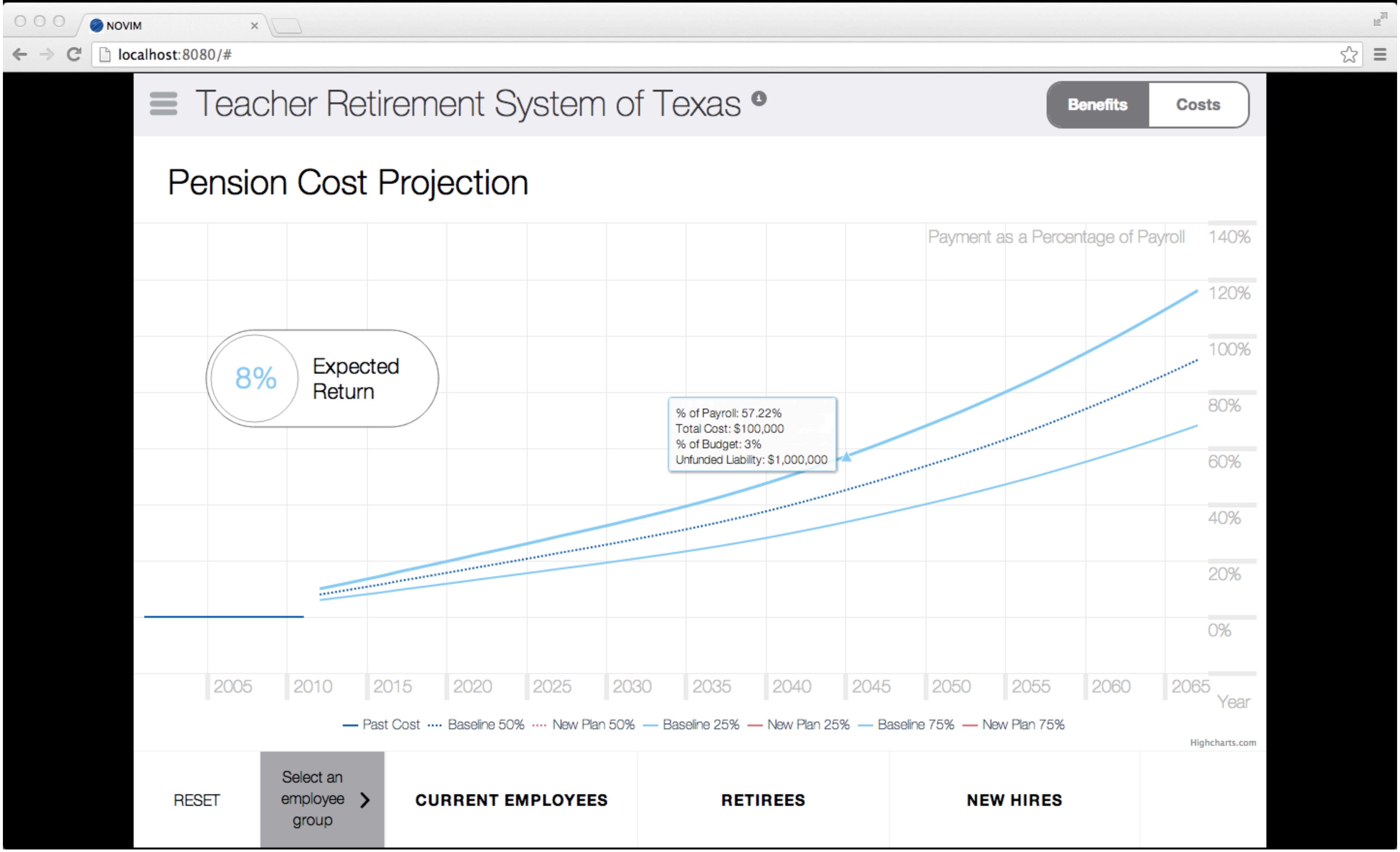
Pension Wealth Accrual

Entry Age 25Y

Vesting 5Y

COLA 0%

Pension Plan Costs



Teacher Retirement System of Texas

Pension Cost Projection

% of Payroll: 57.22%
Total Cost: $100,000
% of Budget: 3%
Unfunded Liability: $1,000,000

Thank You

Novim

ljaf

laura and john arnold foundation®

- Seth Chandler
- Wolfram & the Mathematica community