Python for Economists

EconBrew Series

Pranjal Rawat

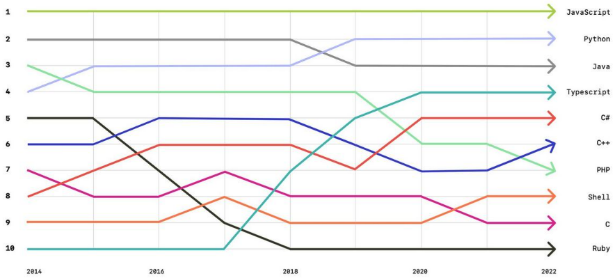
December 26, 2023

Georgetown University

Trends

GitHub Developer Survey

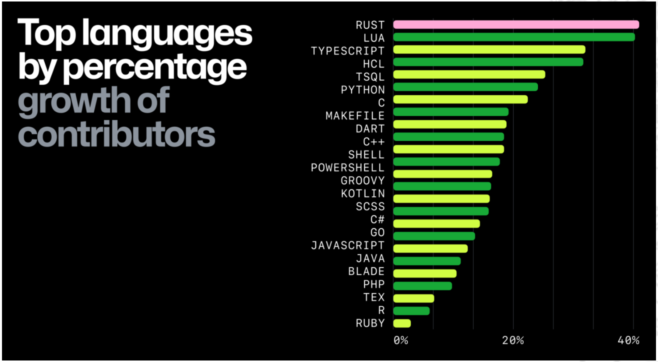
Python is the second most popular programming language (by usage):



Top programming languages on GitHub in 2022 (Source: GitHub)

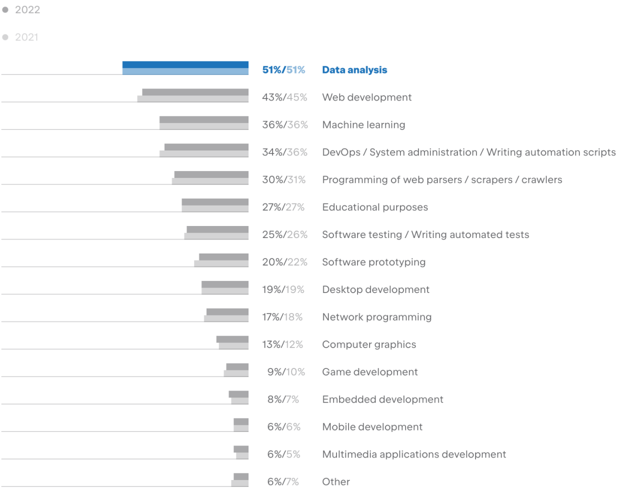
GitHub Developer Survey

Python adoption continues to climb.



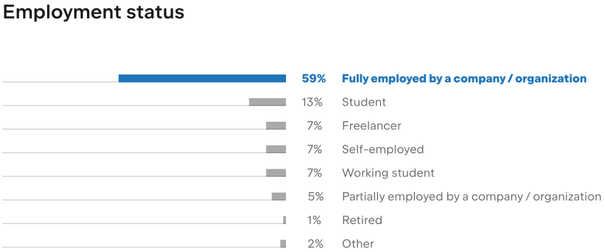
Python Developer Survey

Uses of Python:



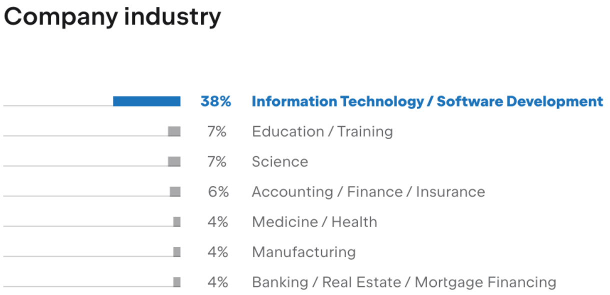
Python Developer Survey

Python is mainly used by professionals.



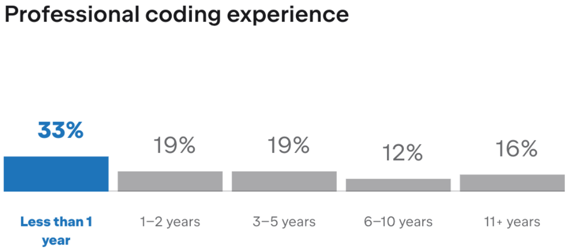
Python Developer Survey

Python is mainly used in information technology, education, and research.



Python Developer Survey

Most Python users are newcomers.



Overview

Why Python?

Pros:

- Free, Open Source
- Easy to learn, write, read, debug
- Mature package ecosystem
- Fast in development time

Cons:

- Slow execution due to Dynamic Types
    - Sol: Type-Hints, Multithreading, Compilers (Cython, Numba, JAX)

Example - I

$$\forall x \in \mathbb{X}, V_{i+1}(x) \leftarrow \max_{0 \le x' \le x} \left[\sqrt{x - x'} + \beta V_i(x')\right]$$

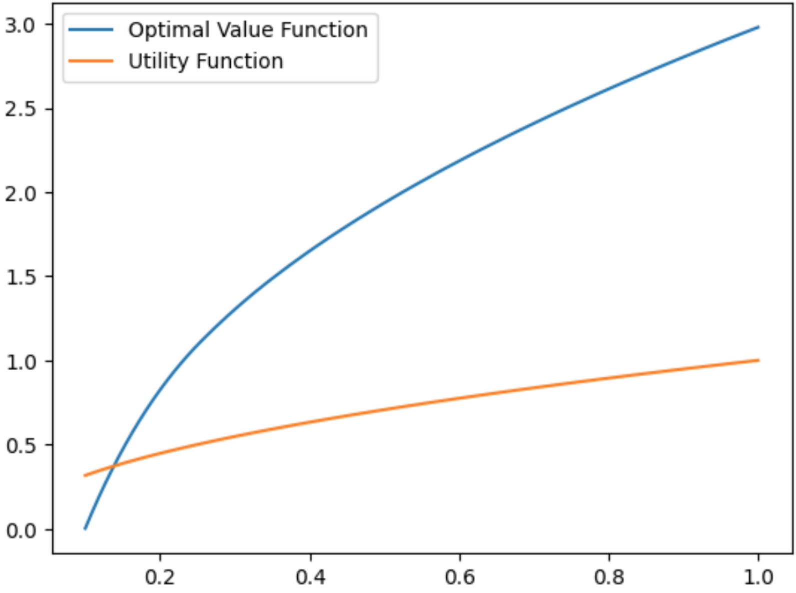
```
# Packages
import numpy as np
import matplotlib.pyplot as plt

# Parameters and Arrays
beta  = 0.95
grid  = np.linspace(0.1, 1, 100)
v_old = np.sqrt(grid)
v_new = np.sqrt(grid)

# Value Function Iteration
for i in range(100):
    for idx, cake in enumerate(grid):
        v_new[idx] = np.max((np.sqrt(cake - grid) + beta * v_old)[grid <= cake])
    v_old[:] = v_new

# Plot
plt.plot(grid, v_old, label = 'Optimal Value Function')
plt.plot(grid, np.sqrt(grid), label = 'Utility Function')
plt.legend()
plt.show()
```

Example - I



Optimal Value Function
Utility Function
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0

Example - II

```
# Packages
from statsmodels.sandbox.regression.gmm import IV2SLS
import pandas as pd

# Matrices
df = pd.read_csv('wage_data.csv')
Y = df[['lwage']]
X = df[['educ', 'IQ', 'KWW', 'exper', 'tenure', 'age', 'married', 'black', 'south', 'urban']]
Z = df[['IQ', 'KWW', 'exper', 'tenure', 'age', 'married', 'black', 'south', 'urban', 'sibs', 'brthord', 'meduc', 'feduc']]

# Fit
X = sm.add_constant(X)
Z = sm.add_constant(Z)
IV2SLS = IV2SLS(Y, X, instrument = Z).fit()
print(IV2SLS.summary())
```

Example - III

```python
# Packages
import pyblp
import numpy as np
import pandas as pd

# Product Characteristics
product_char = pd.read_csv(pyblp.data.BLP_PRODUCTS_LOCATION)
product_char_spec = (pyblp.Formulation('1 + hpwt + air + mpd + space'),                    # linear coeff
                     pyblp.Formulation('1 + prices + hpwt + air + mpd + space'),            # random coeff
                     pyblp.Formulation('1 + log(hpwt) + air + log(mpg) + log(space) + trend')) # cost coeff

# Demographics
demog = pd.read_csv(pyblp.data.BLP_AGENTS_LOCATION)
demog_spec = pyblp.Formulation('0 + I(1 / income)')

# Initialize
BLP1995 = pyblp.Problem(product_char_spec, product_char, demog_spec, demog, costs_type='log')
sigma0 = np.diag([3.612, 0, 4.628, 1.818, 1.050, 2.056])
pi0 = np.c_[[0, -43.501, 0, 0, 0, 0]]

# Solve
results = BLP1995.solve(sigma0, pi0, costs_bounds=(0.001, None))
```

General Packages

- **Arrays and Data**
  `numpy, pandas, dask`
- **Plots**
  `matplotlib, seaborn`
- **Solvers**
  `scipy, sympy, fenicsx`
- **Machine Learning**
  `sklearn, lightgbm`
- **Deep Learning**
  `torch, tensorflow, jax`
- **Web-Scraping**
  `requests, bs4, selenium`
- **Text**
  `nltk, spacy, huggingface`
- **Bayesian**
  `stan, pymc3`
- **Multi-Agent**
  `mesa, ray`

Econ-Specific Packages

- **Econometrics**
  statmodels, linearmodels, pingouin, econml
- **Industrial Org.**
  pyBLP, torch-choice, nashpy
- **Macroeconomics**
  econpizza, sequence-jacobian
- **Time Series**
  pyflux, darts, kats
- **Finance**
  vnpy, qlib, tf-quant-finance
- **Structural Models**
  quantecon, OpenSourceEconomics, econ-HARK

Python Syntax

Usage

- Installing Python
- pip: Package Installation
- venv: Virtual Environments
- How to use Python?
  - Command Line Interface (CLI)
  - Scripts
  - IDEs - Spyder, Visual Code Studio
  - Jupyter Notebooks
  - **Cloud - Google Collab**

Concepts

- Variables and Assignment
  - Strings
  - Int, Float
- Data Structures
  - List
  - Numpy Arrays
  - Pandas Dataframe
  - Others: Dict, Tuple, Set
- Flow control
  - Conditionals (if-elif-else)
  - Loops (for, while)
  - Break, Continue, Pass
- Functions
- Classes
- Comments

Errors

- Common Errors
  - Syntax Error
  - Runtime Error
- Diagnosis
  - Traceback
  - try-except
- How to get help?
  - help()
  - ChatGPT
  - Stack Overflow
  - Package Documentation

Project Files

A typical Python project:

```
README.md
LICENCE.txt
requirements.txt
main.py
utils.py
data/
    data.csv
```

Research with AI/ML

Backpropagation

PyTorch, TensorFlow, and JAX permit backpropagation through a combination of arbitrary functions on data arrays.

- X is data array.
- $g_1(X;\beta) = \beta' X$
- $g_2(Y;\gamma) = e^{\gamma Y}$
- $g_3(Z;\tau) = \log \tau Z$
- $g(X;\beta,\gamma,\tau) = g_3(g_2(g_1(X)))$
- Backprop gives us: $\frac{dg}{d\beta}, \frac{dg}{d\gamma}, \frac{dg}{d\tau}$
- This allows us to tweak (β,γ,τ) to increase $g(X)$

This enables the construction and solution of complex objective functions.

High Dimensional Covariates

- Causal Inference with high dimensional X or Z.
  - $Y_i = \beta T_i + g(X_i; \theta) + \epsilon_i$
  - $Y_i = g_1(X_i; \theta) T_i + g_2(X_i; \theta) + \epsilon_i$
  - Chernozhukov et al 2018, Farrell et al 2021
- Discrete Choice with High Dimensional X:
  - $s_j = \frac{e^{\delta_j}}{\sum_k e^{\delta_k}}$
  - $\delta_j = \alpha p_j + g(X_j; \theta) + \xi_j$
  - Quan and Williams 2021

Solving Dynamic Models

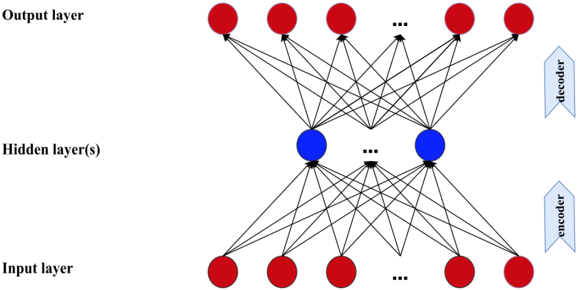
- Using neural nets to represent decision rules:
  - Neural Network: $c_t = g(k_t; \theta)$
  - Objective: $J(\theta) = E_{\epsilon, k_0}\left[\sum_{t=0}^{T} \beta^t u(c_t)\right]$
  - Optimize θ to maximize $J(\theta)$ subject to constraints.
  - Maliar et al 2021
- Maximum Likelihood Models:
  - $y_i = g(x_i, \epsilon_i; \theta)$
  - $P(y_i|x_i, \theta) = \int 1\{y_i = g(x_i, \epsilon_i; \theta)\} dP(\epsilon_i)$
  - $\theta^{MLE} = \text{argmin} \frac{1}{N} \sum_i \log P(y_i|x_i, \theta)$
  - if argmin step is infeasible, then we use gradient descent.
  - Wei and Jiang 2021
- Can solve non-linear PDEs using deep learning (Duarte 2023).

Hypothesis Generation

- When $X- > Y$ mapping is used to generate an interesting hypothesis.
  - X: mugshot of prisioner
  - Y: bail assignment
  - D: numerical attributes
  - Can X predict Y beyond what D can? If so, can we generate hypothesis X' and study its prediction of Y.
  - Ludwig and Mullainathan (2023)

Embeddings

- Latent Factors:
  - $Y_i = \beta * Z_i + \epsilon_i$
  - Z_i is low dimensional representation of X_i
  - Obtained from a hidden-layers of a neural net s.t. $W_i = g(X_i; \theta)$
  - Asset Pricing Factors (Gu et al., 2021), Demand for Fonts (Han et al., 2021)



Exogenous Shocks

- Monetary Policy Shocks:
  - $i_t = g(X_t; \theta) + \epsilon_t$
  - We want to extract ϵ_t which represents shock uncorrelated to the rest of the economy.
  - Use any data (blue book, macro indicators, etc.) to construct X_t to predict i_t using neural networks.
  - The residual will necessarily be an "exogenous" shock.
  - Aruoba and Drechsel (2022)

Agent Learning

- Learning through Reinforcement
  - Policies: $a_t = g(s_t; \theta)$
  - Value function: $V(s) = g(s; \theta)$
  - Sample actions through trial and error and improve valuations.
  - Momentum and Reversals in Artificial Stock Markets (Chiarella et al 2016, Maeda et al 2020).

Market Design

- Myerson Auctions as a Deep Learning Problem
  - Valuations $\vec{v} \sim F$, Bids $\vec{b}$
  - Auction: Allocation $g(\vec{b}; \theta)$, Pricing $p(\vec{b}; \gamma)$
  - Maximize Exp Revenue: $E_{v \sim F}\left[\sum_i p_i(b; \gamma)\right]$
  - Constraint: $\forall i$, given $v_{-i}, E_{v_i}\left[\max_{v_i'}(v_i' - p_i) - (v_i - p_i)\right] = 0$
  - Sample valuations v and find (θ, γ) that solves the empirical version of the problem.
  - Optimal Auctions through Deep Learning (Dutting et al., 2023)

Conclusion

What AI/ML can offer Economics beyond Prediction:

- **Heterogenous Treatment Effects**
- **Handling High-Dimensional Covariates**
- **Dimensionality Reduction**, Latent Factors
- Solving Theoretical Models - Macro, IO, Auctions, Finance
- Estimating Models with Data
- Extracting Exogenous Components
- Modelling Agent Learning
- Hypothesis Generation

Highlighted topics are especially useful in industry.

References

References (A-G)

- Aruoba, S. B., & Drechsel, T. (2022). Identifying Monetary Policy Shocks: A Machine Learning Approach.
- Chiarella, C., & Ladley, D. (2016). Chasing trends at the micro-level: The effect of technical trading on order book dynamics. Journal of Banking & Finance, 72, S119-S131.
- Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., & Robins, J. (2018). Double/debiased machine learning for treatment and structural parameters.
- Dütting, P., Feng, Z., Narasimhan, H., Parkes, D., Ravindranath, S. S. (2019, May). Optimal auctions through deep learning. In International Conference on Machine Learning (pp. 1706-1715). PMLR.
- Duarte, V. (2018). Machine learning for continuous-time finance. SSRN, 2023a. URL https://ssrn.com/abstract, 3012602.
- Farrell, M. H., Liang, T., & Misra, S. (2021). Deep neural networks for estimation and inference. Econometrica, 89(1), 181-213.
- Gu, S., Kelly, B., & Xiu, D. (2021). Autoencoder asset pricing models. Journal of Econometrics, 222(1), 429-450.

References (H-P)

- Han, S., Schulman, E. H., Grauman, K., & Ramakrishnan, S. (2021). Shapes as product differentiation: Neural network embedding in the analysis of markets for fonts. arXiv preprint arXiv:2107.02739.
- Inbal Shani, G. S. (2023, September 29). Survey reveals Ai's impact on the developer experience. The GitHub Blog. https://github.blog/2023-06-13-survey-reveals-ais-impact-on-the-developer-experience/
- Ludwig, J., & Mullainathan, S. (2023). Machine Learning as a Tool for Hypothesis Generation (No. w31017). National Bureau of Economic Research.
- Maeda, I., DeGraw, D., Kitano, M., Matsushima, H., Sakaji, H., Izumi, K., & Kato, A. (2020). Deep reinforcement learning in agent based financial market simulation. Journal of Risk and Financial Management, 13(4), 71.
- Maliar, L., Maliar, S., & Winant, P. (2021). Deep learning for solving dynamic economic models. Journal of Monetary Economics, 122, 76-101.
- Python Developers Survey 2022 Results. (n.d.). JetBrains: Developer Tools for Professionals and Teams. https://lp.jetbrains.com/python-developers-survey-2022/

References (Q-Z)

- Quan, T. W. (2023). Extracting Characteristics from Product Images and its Application to Demand Estimation. The RAND Journal of Economics, Revise and Resubmit.
- Wei, Y. M., & Jiang, Z. (2022). Estimating parameters of structural models using neural networks. USC Marshall School of Business Research Paper.
- 3.12.0 documentation. (n.d.). Python 3.12.0 Documentation. https://docs.python.org/3/