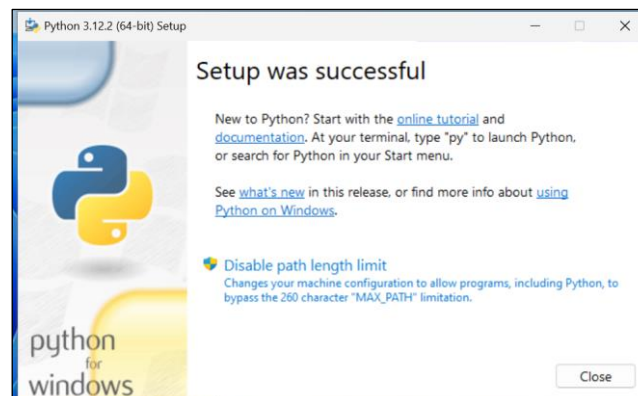
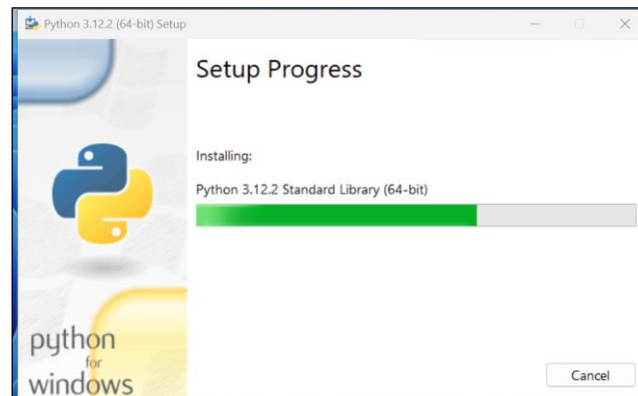


## Chapter 1: A Gentle Introduction to Python



## Chapter 2: Built-In Data Types

The screenshot shows the Python Tutor interface with the following elements:

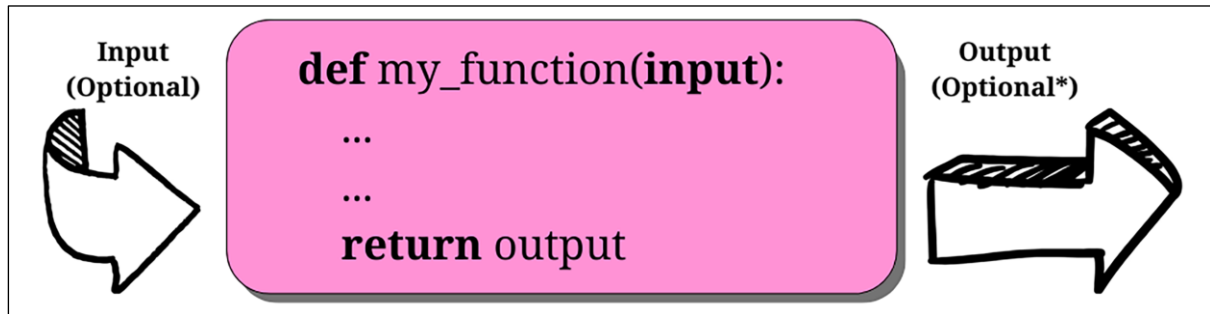
- Code Editor:** Contains the code `age = 42`. The line number `1` is highlighted in green, indicating the current execution step. Above the code, it says "Python 3.11" and "known limitations". Below the code, there is a link "Edit this code".
- Execution Controls:** On the left, there are two arrows: a green arrow pointing right labeled "line that just executed" and a red arrow pointing right labeled "next line to execute". Below these is a progress bar and navigation buttons: "<< First", "< Prev", "Next >", and "Last >>".
- Status:** At the bottom, it says "Done running (1 steps)".
- Frames and Objects:** On the right, there is a diagram showing the "Global frame" containing a variable "age". An arrow points from "age" to an "int" object with the value "42".

Positive Indexing									
0	1	2	3	4	5	6	7	8	9
H	e	l	l	o	T	h	e	r	e
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Negative Indexing									
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
H	e	l	l	o	T	h	e	r	e
0	1	2	3	4	5	6	7	8	9

## Chapter 4: Functions, the Building Blocks of Code



Python 3.11  
[known limitations](#)

```
1 x = 3
2
3 def func(y):
4     print(y)
5
6 func(x) # prints: 3
```

[Edit this code](#)

→ line that just executed  
→ next line to execute

<< First < Prev Next > Last >>

Step 6 of 6

Print output (drag lower right corner to resize)

3

Frames      Objects

Global frame

x → int 3

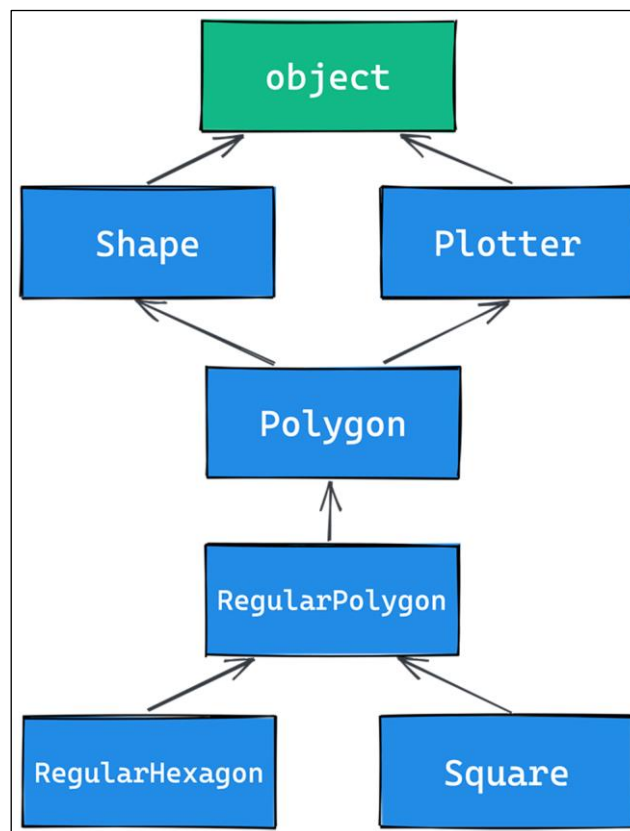
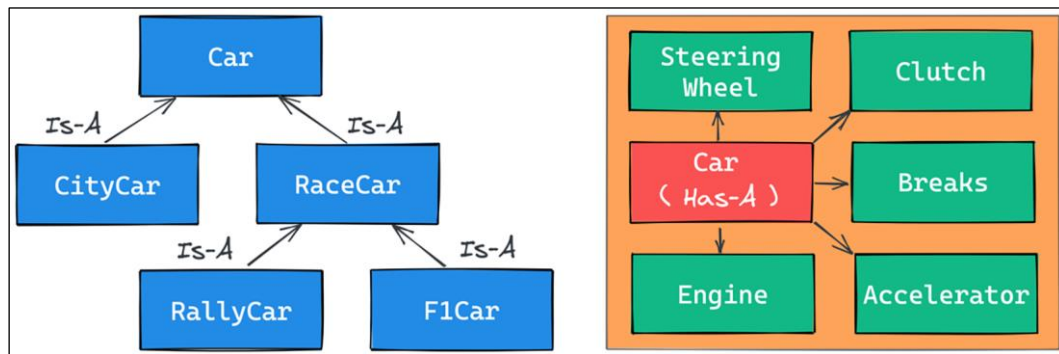
func → function func(y)

func

y → NoneType None

Return value

## Chapter 6: OOP, Decorators, and Iterators



# Chapter 13: Data Science in Brief

jupyter example Last Checkpoint: 13 seconds ago

File Edit View Run Kernel Settings Help

Trusted

JupyterLab Python 3 (ipykernel)

```
[1]: 1 def fibonacci(N):
      2     a, b = 0, 1
      3     while a < N:
      4         yield a
      5         a, b = b, a + b

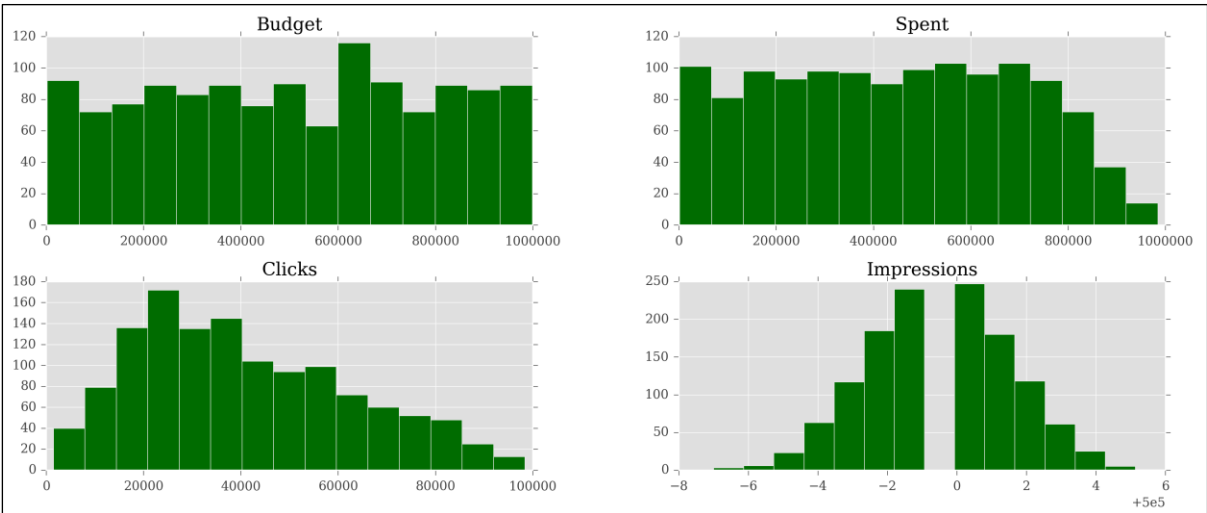
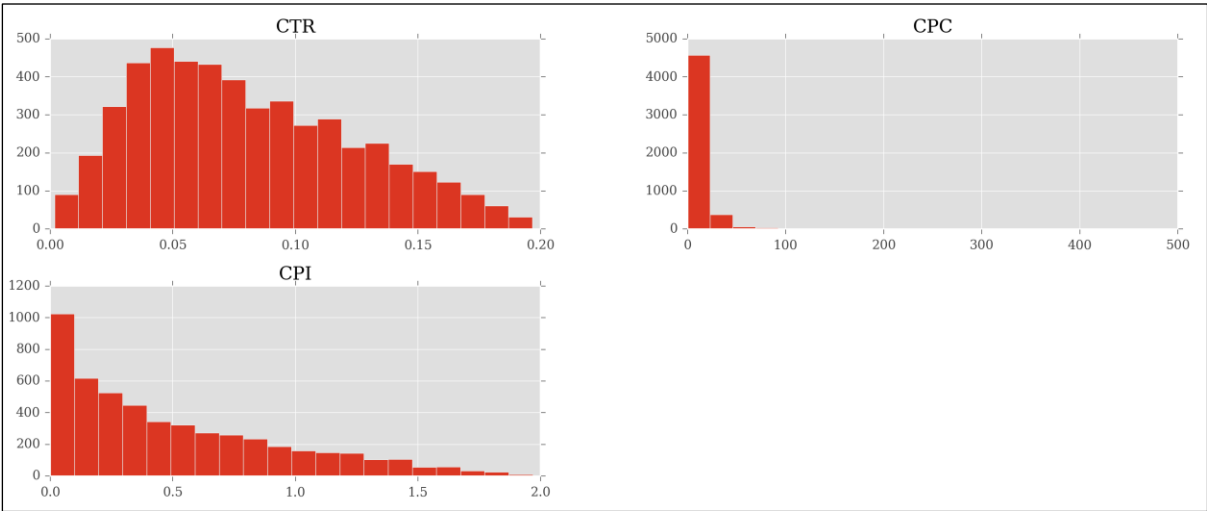
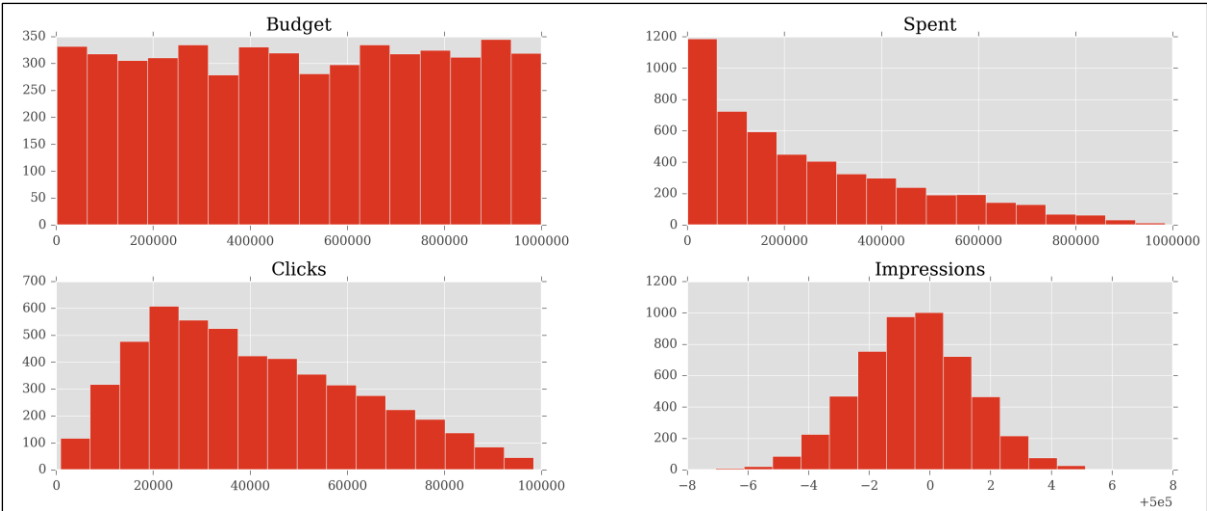
[2]: 1 list(fibonacci(100))

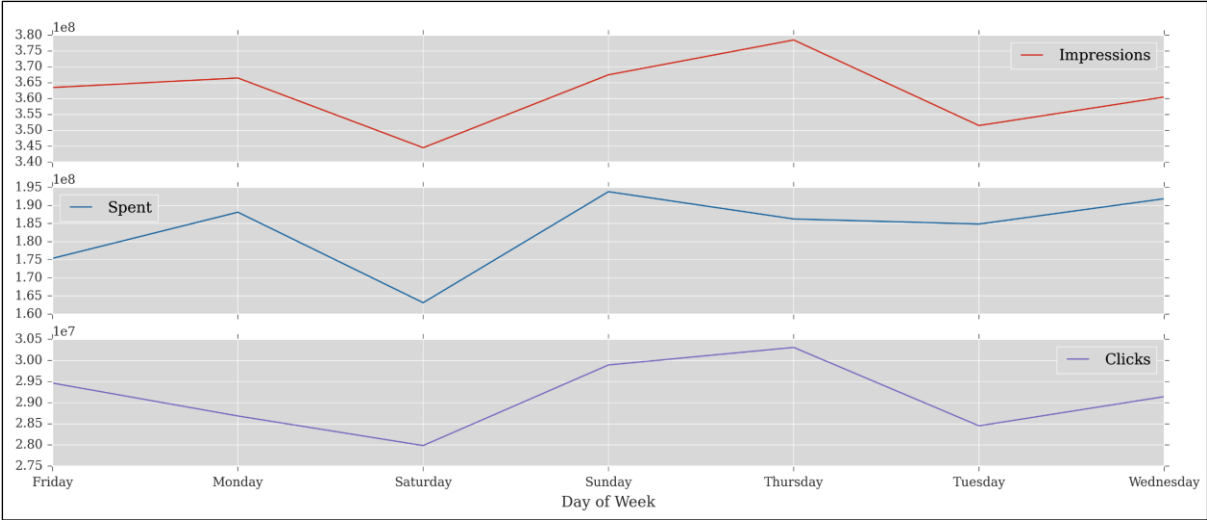
[2]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

[3]: 1 %timeit list(fibonacci(10**4))
      1.16 µs ± 48 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
```

	cmp_name	cmp_bgt	cmp_spent	cmp_clicks	cmp_impr	user
0	KTR_20250404_20250916_35-50_A_EUR	964496	29586	36632	500001	{"username": "epennington", "name": "Stephanie..."}
1	AKX_20240130_20241017_20-25_M_GBP	344739	166010	67325	499999	{"username": "epennington", "name": "Stephanie..."}
2	BYU_20230828_20250115_25-45_M_GBP	177403	125738	29989	499997	{"username": "joshua61", "name": "Diana Richar..."}
3	AKX_20250216_20261129_45-60_F_USD	618256	75017	76301	500000	{"username": "joshua61", "name": "Diana Richar..."}
4	AKX_20231229_20250721_20-40_F_GBP	113805	12583	48915	500001	{"username": "joshua61", "name": "Diana Richar..."}

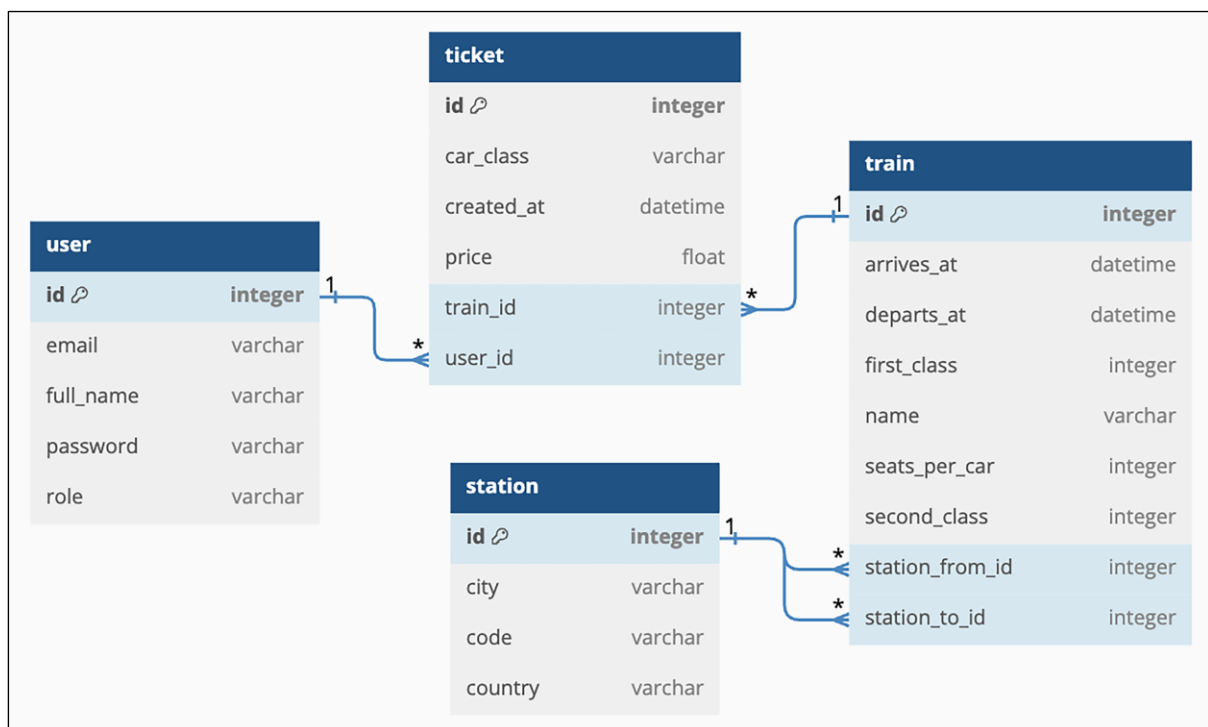
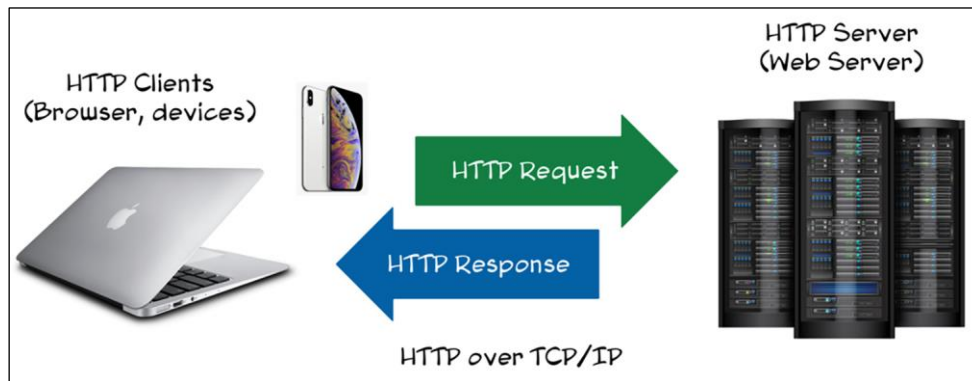
	Duration	Budget	Clicks	Impressions	Spent	CTR	CPC	CPI	Age
count	5065.000000	5065.000000	5065.000000	5065.000000	5065.000000	5065.000000	5065.000000	5065.000000	5065.000000
mean	369.488253	502965.054097	40265.781639	499999.474630	253389.854689	0.080532	10.131895	0.506780	53.882330
std	209.852969	290468.998656	21840.783154	2.023801	222774.897138	0.043682	19.527218	0.445550	21.170077
min	1.000000	1764.000000	899.000000	499992.000000	107.000000	0.001798	0.001493	0.000214	18.000000
25%	192.000000	251171.000000	22575.000000	499998.000000	67071.000000	0.045150	1.756140	0.134143	35.000000
50%	371.000000	500694.000000	36746.000000	499999.000000	187743.000000	0.073493	5.207316	0.375486	54.000000
75%	554.000000	756850.000000	55817.000000	500001.000000	391790.000000	0.111634	11.630131	0.783572	72.000000
max	730.000000	999565.000000	98379.000000	500007.000000	984705.000000	0.196758	462.814233	1.969410	90.000000






Target Gender	Clicks				Impressions				Spent	
	A	F	M	A	F	M	A	F	M	
	Target Age									
20-25	2460345	2355790	2954169	28499957	29999964	34999963	12387854	16271204	14605751	
20-30	2254458	1742729	2133740	28999969	21999963	25999959	14631175	11435369	13184984	
20-35	1323341	1735301	2926626	19499944	22499974	34999942	8530330	10987452	18383305	
20-40	2304325	1987013	1975200	30499967	28499973	26999950	15591491	15490069	13806347	
20-45	2402785	1667405	1790037	26999977	22499993	21999968	11692494	9064229	9623006	

## Chapter 14: Introduction to API Development







Admin			^
DELETE	/admin/stations/{station_id}	Admin Delete Station	▼
Stations			^
GET	/stations	Get Stations	▼
POST	/stations	Create Station	▼
GET	/stations/{station_id}	Get Station	▼
PUT	/stations/{station_id}	Update Station	▼
DELETE	/stations/{station_id}	Delete Station	 ▼
Trains			^
GET	/stations/{station_id}/departures	Get Station Departures	▼

## Chapter 16: Packaging Python Applications

⚠ You are using TestPyPI – a separate instance of the Python Package Index that allows you to try distribution tools and processes without affecting the real index.




[Help](#) [Sponsors](#) [Log in](#) [Register](#)

# railway-cli 0.0.1

✓ Latest version

```
pip install -i https://test.pypi.org/simple/ railway-cli==0.0.1
```



Released: less than 10 seconds ago

A CLI client for the railway API

### Navigation

- Project description
- Release history
- Download files

### Project description

#### Railway CLI

A railway CLI application to demonstrate packaging and distribution of a Python project for Chapter 16 of "Learn Python Programming, 4d Edition", by Fabrizio Romano and Heinrich Kruger. At the same time, it acts as an example of an application built around the trains API project from Chapter 14 of the same book.