Python for ooRexx

Rexx Symposium

May 7, 2025

About me

Kaan Kuzugüdenli

kaan.k@outlook.com

linkedin.com/in/kaank

github.com/kaan1212

Student at the Vienna University of Economics and Business

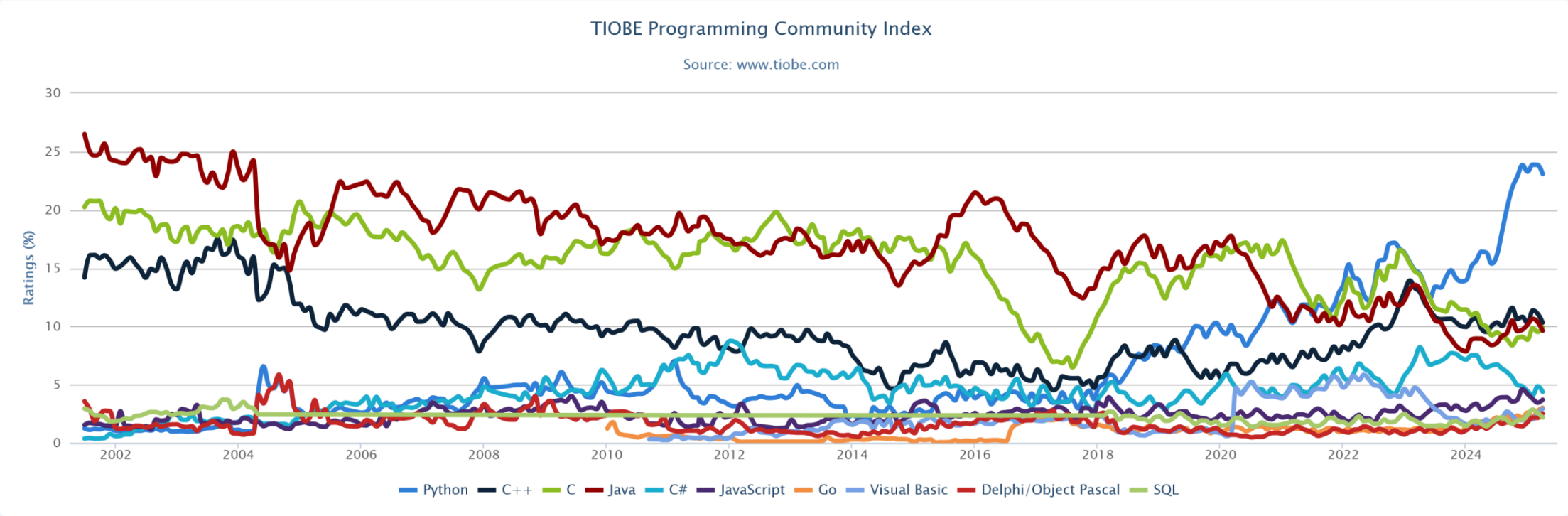
Thesis supervised by Prof. Dr. Rony G. Flatscher

Software engineer: Java, C#, Python, PHP, Go, Pascal, JavaScript etc.

Python

Part I

Popularity

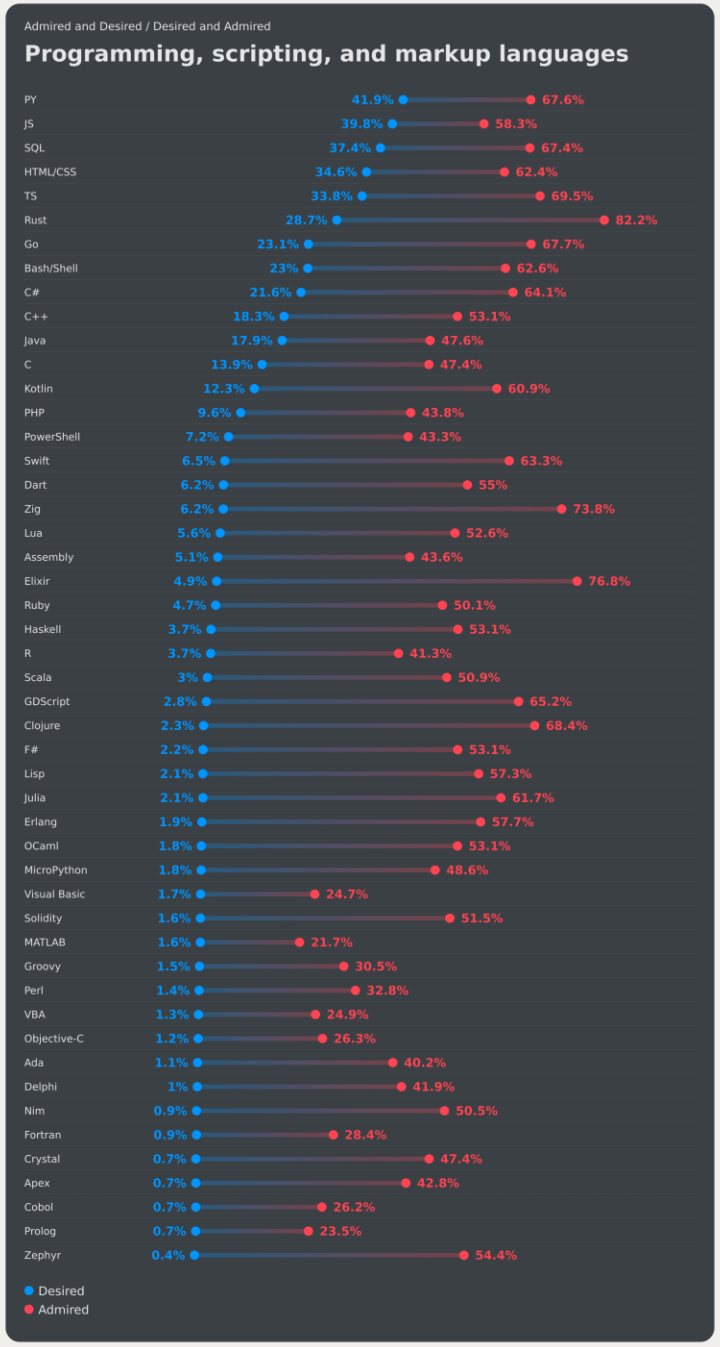

TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
2024
Python
C++
C
Java
C#
JavaScript
Go
Visual Basic
Delphi/Object Pascal
SQL

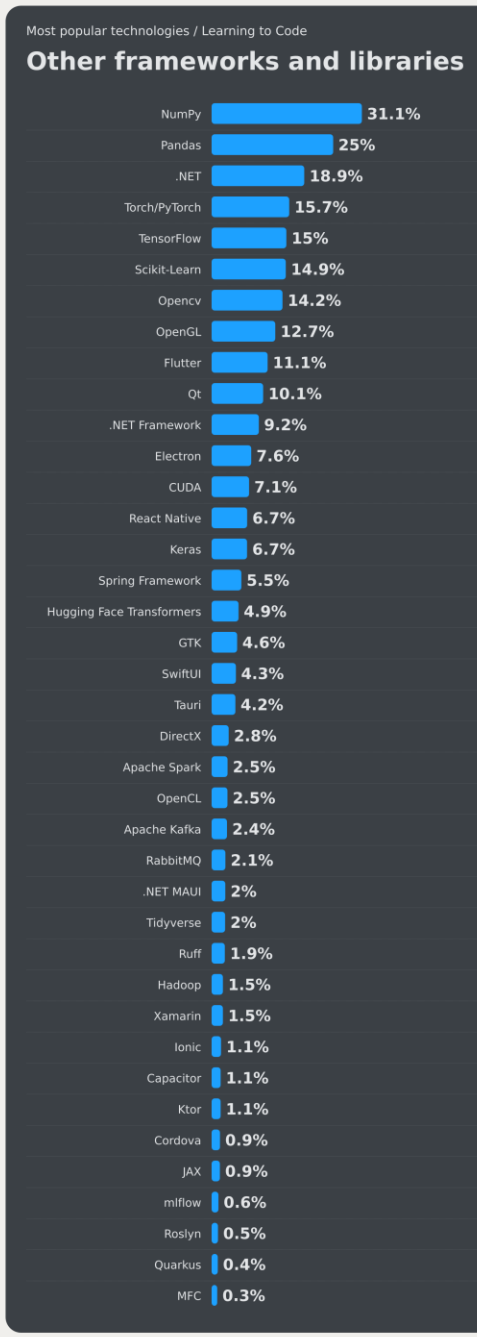
Popularity

2024 Stack Overflow Developer Survey

"JavaScript, **Python** and SQL are all highly-desired and admired programming languages, but Rust continues to be the most-admired programming language with an 83% score this year."

".NET is the most used among other frameworks and libraries again this year for all developers. Those learning to code are using **NumPy and Pandas** the most (as they were last year)."





History

Guido van Rossum

https://gvanrossum.github.io

Python 0.9 (1991)

Python 1.0 (1994)

Python 2.0 (2000)

Python 3.0 (2008)

Features

High-level

General-purpose

Interpreted

Interactive

Object-oriented

Multi-paradigm

Typing: duck, dynamic, strong

Garbage-collected

Very high-level dynamic data types (list, tuple, set, dict)

Large standard library ("batteries included")

Implementations

CPython	reference implementation
PyPy	just-in-time (JIT) compiler
Jython	Java Virtual Machine (JVM)
IronPython	Common Language Runtime (CLR)
MicroPython, CircuitPython	microcontrollers
Stackless Python	microthreads

Many more …

Tooling

IDE	Microsoft Visual Studio Code
	JetBrains PyCharm
	Jupyter Notebook
Package manager	pip
Virtual environments	venv
Unit testing	unittest
Static code analysis	pylint
Formatter	autopep8

Libraries

Data science	NumPy, SciPy, Pandas, Matplotlib
Machine learning	scikit-learn, TensorFlow, Keras, PyTorch
Image processing	Pillow, OpenCV
Web frameworks	Django, Flask, FastAPI
Web scraping	Beautiful Soup, Scrapy, Selenium
Databases	SQLAlchemy
Networking	Requests

Example: Hello, World!

```
from getpass import getuser


def make_greeting(name):
    if name:
        return 'Hello, ' + name + '!'
    else:
        return f'Hello, {getuser()}!'


name = input('Name: ')
greeting = make_greeting(name)
print(greeting)
```

```
C:\>py hello.py
Name: ooRexx
Hello, ooRexx!


C:\>py hello.py
Name:
Hello, Kaan!
```

Example: Control flow

```
for i in range(3):
    print(i)

names = ['Alice', 'Bob', 'Charlie']

for name in names:
    print(name)

for i in range(len(names)):
    print(str(i) + ': ' + names[i])

i = 0
while i < 3:
    print(i)
    i += 1
```

```
C:\>py control_flow.py
0
1
2
Alice
Bob
Charlie
0: Alice
1: Bob
2: Charlie
0
1
2
```

Example: Data structures

```
mylist = [1, 3.14, True, None]
mytuple = (1, 2, 3)
myset = {1, 2, 2, 3, 3, 3}
mydict = {'a': 1, 'b': 2, 'c': 3}

mylist.append('Hello')
print(mylist)

print(mytuple[-1])

print(len(myset))

print(mydict['a'])
print('d' in mydict)
```

```
C:\>py data_structures.py
[1, 3.14, True, None, 'Hello']
3
3
1
False
```

Example: Multiple assignment

```
a, b = 1, 2
print(f'a={a}, b={b}')

b, a = a, b
print('a=' + str(a) + ', b=' + str(b))
```

```
C:\>py multiple_assignment.py
a=1, b=2
a=2, b=1
```

Example: Sequence unpacking

```
mylist = [1, 2, 3, 4, 5]

a, b, *other = mylist

print(f'a={a}, b={b}, other={other}')
```

```
C:\>py sequence_unpacking.py
a=1, b=2, other=[3, 4, 5]
```

Example: Slicing

```
mylist = [1, 2, 3, 4, 5, 6, 7, 8, 9]

print(mylist[:])
print(mylist[:4])
print(mylist[5:])
print(mylist[4:5])
print(mylist[::2])
```

```
C:\>py slicing.py
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4]
[6, 7, 8, 9]
[5]
[1, 3, 5, 7, 9]
```

Resources

https://www.python.org/

https://www.python.org/downloads/

https://docs.python.org/3/faq/general.html

https://docs.python.org/3/tutorial/index.html

https://docs.python.org/3/library/index.html

ooRexx

Part II

Bridges

	BSF4ooRexx	Python for ooRexx
Bridges to	Java	Python
Direction	bidirectional	unidirectional
Supports Rexx	y	n
Supports ooRexx	y	y
Status	general availability (GA)	work in progress

Python for ooRexx

https://github.com/kaan1212/python-for-oorexx

Requirements

- Windows 10 or later
- ooRexx 5.1.0
- Python 3.12.7

Required files

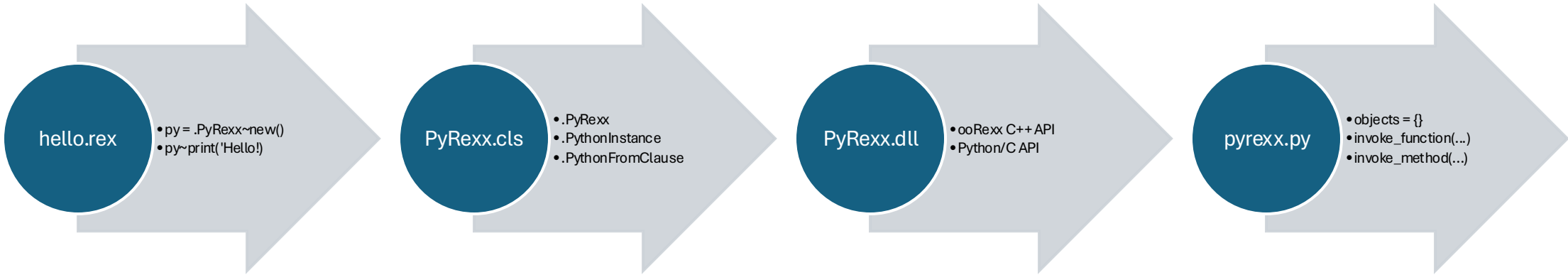
- PyRexx.cls
- PyRexx.dll
- pyrexx.py

Example: Hello, World!

```
py = .PyRexx~new()
py~from('getpass')~import('getuser')

name = py~input('Name: ')

if name = '' then
    name = py~getuser()

py~print('Hello, ', name, '!', py~kwd('sep', ''))

::requires 'PyRexx.cls'
```

```
C:\>rexx hello.rex
Name: ooRexx
Hello, ooRexx!

C:\>rexx hello.rex
Name:
Hello, Kaan!
```

Python for ooRexx



Example: List

```
py = .PyRexx~new()

list = py~list()
list~append(py~int(1))
list~append(py~float(3.14))
list~append(py~bool(1))
list~append(py~str('Hello'))

say list

list = py[py~0, py~True, py~False, py~None, 'Hello']

say list

::requires 'PyRexx.cls'
```

```
C:\>rexx list.rex
[1, 3.14, True, 'Hello']
[0, True, False, None, 'Hello']
```

Example: List iteration

```
py = .PyRexx~new()

list = py[]

do i=1 to 3
   list~append(py~int(i))
end

do i=0 to py~len(list)~makeString - 1
   say list[py~int(i)]
end

do item over list
   say item
end

::requires 'PyRexx.cls'
```

```
C:\>rexx list_iteration.rex
1
2
3
1
2
3
```

Example: Dictionary

```
py = .PyRexx~new()

dict = py~dict()
dict['a'] = py~1
dict['b'] = py~2
dict['c'] = py~3

say dict['a']
say py~len(dict)
say py~operator.contains(dict, 'a')
say py~operator.contains(dict, 'd')

::requires 'PyRexx.cls'
```

```
C:\>rexx dict.rex
1
3
1
0
```

Example: Slicing

```
py = .PyRexx~new()

list = py[]

do i=1 to 9
   list~append(py~int(i))
end

say list
say list[,]
say list[, py~4]
say list[py~5, py~None]
say list[py~4, py~5]
say list[, , py~2]

::requires 'PyRexx.cls'
```

```
C:\>rexx slicing.rex
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4]
[6, 7, 8, 9]
[5]
[1, 3, 5, 7, 9]
```

Example: UUID

```
py = .PyRexx~new()
py~import('uuid')

say py~uuid.uuid1()
say py~uuid.uuid4()

::requires 'PyRexx.cls'
```

```
C:\>rexx uuid.rex
16abeef9-2ad7-11f0-add8-7008942fabec
ac6c6dd7-da5e-4a80-8488-cd6e85b2323a
```

Example: Random

```
py = .PyRexx~new()
py~import('random')

say py~random.choice(py['win', 'lose', 'draw'])

deck = py['ace', 'two', 'three', 'four']
py~random.shuffle(deck)
say deck

::requires 'PyRexx.cls'
```

```
C:\>rexx random.rex
win
['three', 'two', 'ace', 'four']
```

Example: Logging

```
py = .PyRexx~new()
py~import('logging')

logger = py~logging.getLogger(py~__name__)
logger~info('This is a information.')
logger~warning('This is a warning.')
logger~error('This is an error.')

::requires 'PyRexx.cls'
```

```
C:\>rexx logging.rex
This is a warning.
This is an error.
```

Example: SQLite

```
py = .PyRexx~new()
py~import('sqlite3')

con = py~sqlite3.connect('imdb.db')
cur = con~cursor()

cur~execute('CREATE TABLE movie(title)')
cur~execute("INSERT INTO movie VALUES ('E.T.')")
con~commit()

res = cur~execute('SELECT title FROM movie')
say res~fetchall()

con~close()

::requires 'PyRexx.cls'
```

```
C:\>rexx sqlite3.rex
[('E.T.',)]
```

Example: Extending a Python class

```
py = .PyRexx~new()
py~from('html.parser')~import('HTMLParser')

py~defineClass(.MyHTMLParser, 'HTMLParser')
parser = py~MyHTMLParser()
parser~feed('<html><body>Hello</body></html>')

::class MyHTMLParser
   ::method handle_starttag
      use arg pself, tag, attrs
      say 'Encountered a start tag:' tag

::requires 'PyRexx.cls'
```

```
C:\>rexx html.parser.rex
Encountered a start tag: html
Encountered a start tag: body
```

Example: scikit-learn

https://en.wikipedia.org/wiki/Iris_flower_data_set

```
py = .PyRexx~new()
py~import("numpy")
py~from("sklearn.datasets")~import("load_iris")
py~from("sklearn.tree")~import("DecisionTreeClassifier")

iris = py~load_iris()

-- train test split
test_idx = py[py~0, py~50, py~100]

train_target = py~numpy~delete(iris~target, test_idx)
train_data = py~numpy~delete(iris~data, test_idx, py~kwd("axis", py~0))

test_target = iris~target[test_idx]
test_data = iris~data[test_idx]

-- train the classifier
clf = py~DecisionTreeClassifier()
clf~fit(train_data, train_target)

-- classify test data
say test_target
say clf~predict(test_data)

::requires 'PyRexx.cls'
```

```
C:\>rexx scikit-learn.rex
[0 1 2]
[0 1 2]
```

LLM comparison

	OpenAI API compatible	Deployment	Notes
OpenAI API	y	SaaS	
Google Gemini API	y	SaaS	
DeepSeek API	y	SaaS	
Anthropic API	y	SaaS	
llama.cpp	y	local, open source	
Ollama	y	local, open source	based on llama.cpp
Hugging Face	y	local, open source	popular repository
vLLM	y	local, open source	

Example: LLM text output

```
py = .PyRexx~new()
py~import('json')
py~from("openai")~import("OpenAI")

jsonString = .resources~entry("json_string")~makeString
pythonObject = py~json~loads(jsonString)
kwargs = py~kwargs(pythonObject)

client = py~OpenAI()
completion = client~chat~completions~create(kwargs)
say completion~choices[py~0]~message~content

::resource json_string
{
   "model": "gpt-4.1-nano",
   "messages": [
      {
         "role": "user",
         "content": "Write a one-sentence bedtime story about a unicorn."
      }
   ]
}
::END

::requires 'PyRexx.cls'
```

C:\>rexx openai_text.rex

Under the glow of the moon, the gentle unicorn drifted into a peaceful sleep, dreaming of enchanted forests and sparkling stars.

C:\>rexx openai_text.rex

Tonight, the gentle unicorn drifted into a peaceful sleep beneath a sky filled with shimmering stars, dreaming of magical adventures yet to come.

C:\>rexx openai_text.rex

Under the glow of a twinkling moon, the brave unicorn gently explored the enchanted forest, dreaming sweetly of magical adventures to come.

Example: LLM image analysis

```
::resource json_string
{
   "model": "gpt-4.1-nano",
   "messages": [
      {
         "role": "user",
         "content": [
            {
               "type": "text",
               "text": "What's in this image?"
            },
            {
               "type": "image_url",
               "image_url": {
                  "url":
"https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Gfp-
wisconsin-madison-the-nature-boardwalk.jpg/120px-Gfp-wisconsin-
madison-the-nature-boardwalk.jpg"
               }
            }
         ]
      }
   ]
}
::END
```

C:\>rexx openai_image.rex

The image shows a wooden pathway or boardwalk running through a lush, green landscape with tall grass on either side. In the background, there are trees and a partly cloudy blue sky above.

Example: LLM structured output

```
jsonString = completion~choices[py~0]~message~content

pythonObject = py~json~loads(jsonString)
say pythonObject
say pythonObject["participants"][py~0]

::resource json_string
{
   "model": "gpt-4.1-nano",
   "messages": [
      {
         "role": "system",
         "content": "You are a helpful assistant designed to output JSON.
Please respond in the format {name: ..., date: ..., participants: ...}"
      },
      {
         "role": "user",
         "content": "Alice and Bob are going to a science fair on Friday."
      }
   ],
   "response_format": { "type": "json_object" }
}
::END

::requires 'PyRexx.cls'
```

```
C:\>rexx openai_structured.rex

{'name': 'Science Fair', 'date': 'Friday', 'participants':
   ['Alice', 'Bob']}

Alice
```

Roadmap

1. Enhanced compatibility with OpenAI API
2. Improved compatibility between ooRexx and Python types
3. Better error handling, and debugging facilities
4. Support for Linux and macOS
5. Garbage collection
6. CI/CD pipeline

Get started!

https://github.com/kaan1212/python-for-oorexx

- README
- Unit tests (150 tests with 283 assertions available)
  - Built-in functions
  - Built-in types
  - Standard library
- Report any issues
- Follow the GitHub repository to get notified of updates

Thank you!