Capgemini

# Rightcall
# Call Center Monitoring

Bielsko-Biala, 29th of October 2018

# Agenda

1. How it works?
2. Web automation
3. Secure login
4. Speech recognition
5. Customer sentiment analysis
6. Google Sheets
7. GUI automation
8. IA vs RPA

# Problem

Your customers want your agents to follow exceptional call center quality assurance best practices:

- frequent performance checks,
- call script monitoring,
- self-service tools promotion monitoring,
- sentiment analysis.

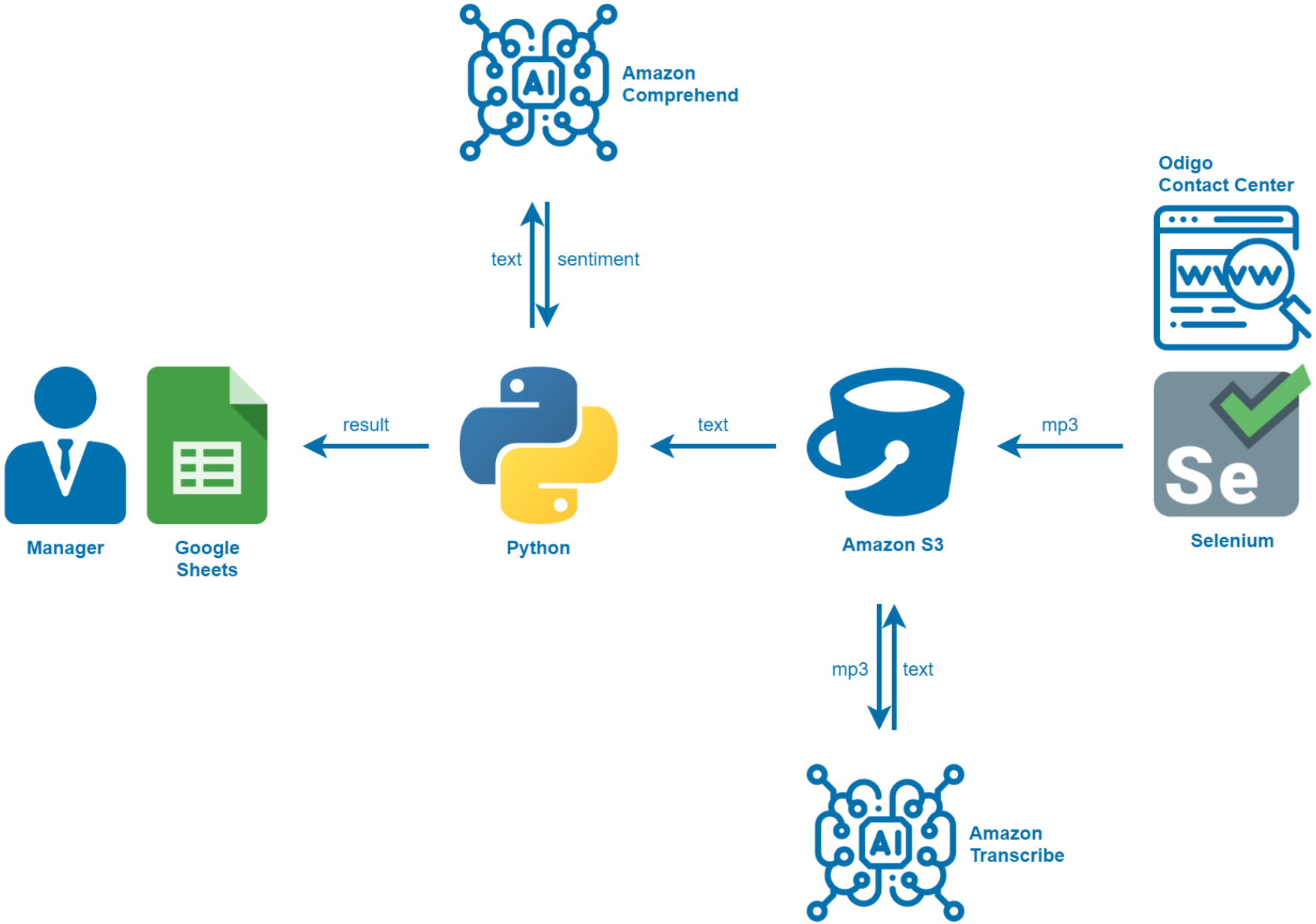
# Solution

**Rightcall** is right call center quality assurance monitoring written in Python. Key Features:

- frequent performance checks,
- call script monitoring,
- self-service tools promotion monitoring,
- sentiment analysis.

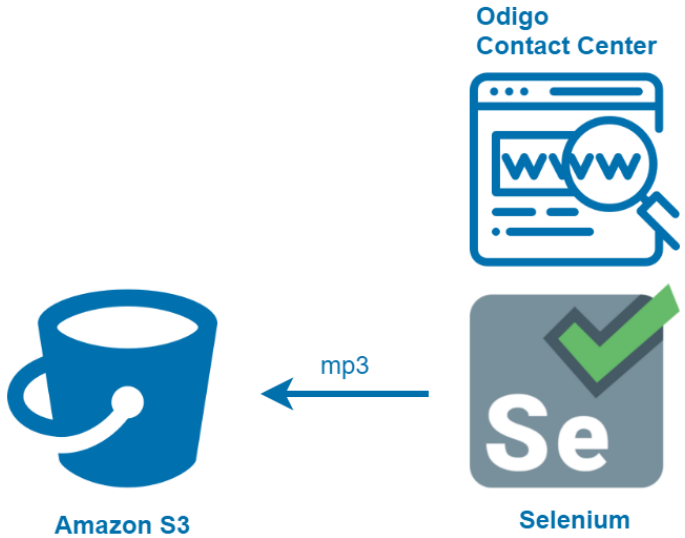
# How it works?



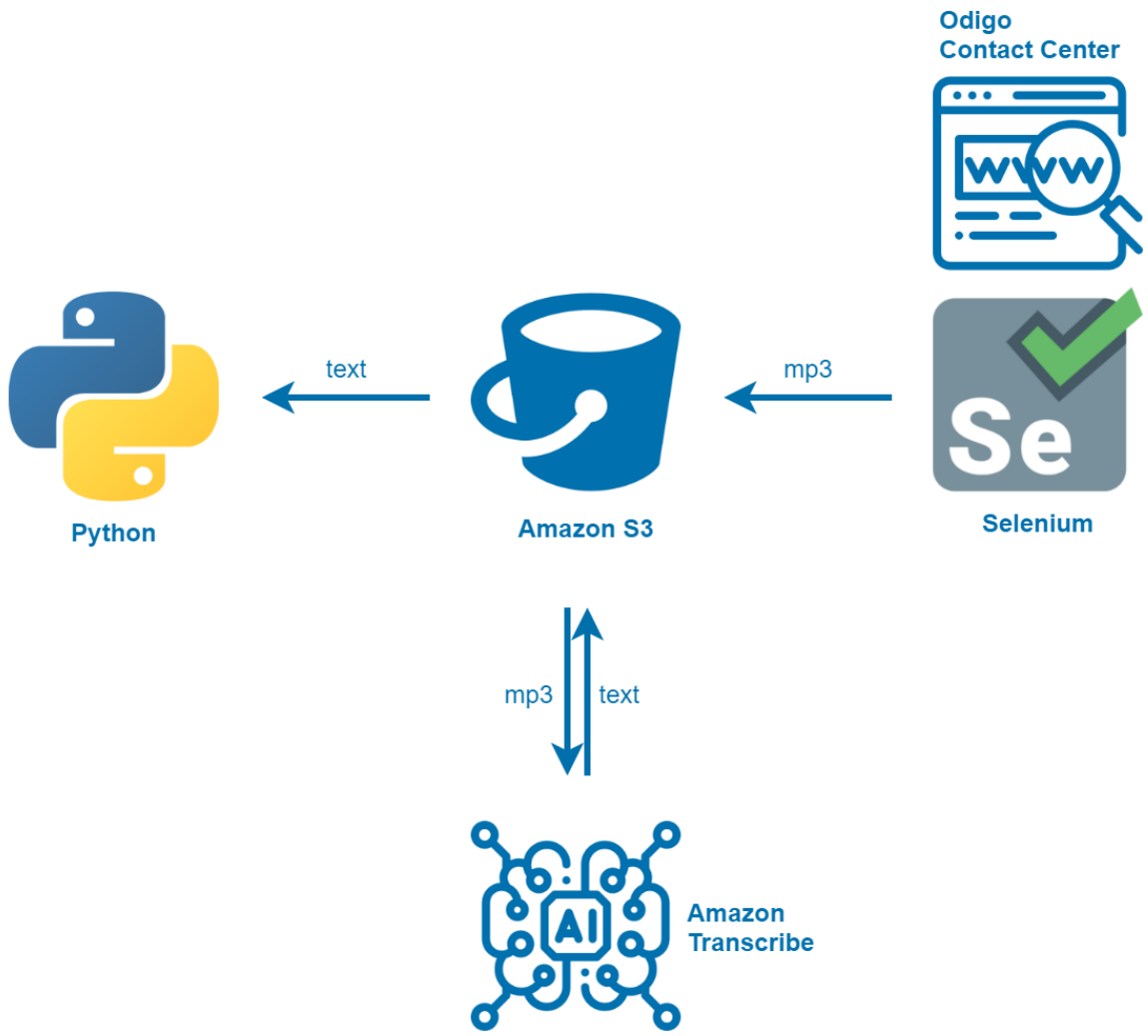




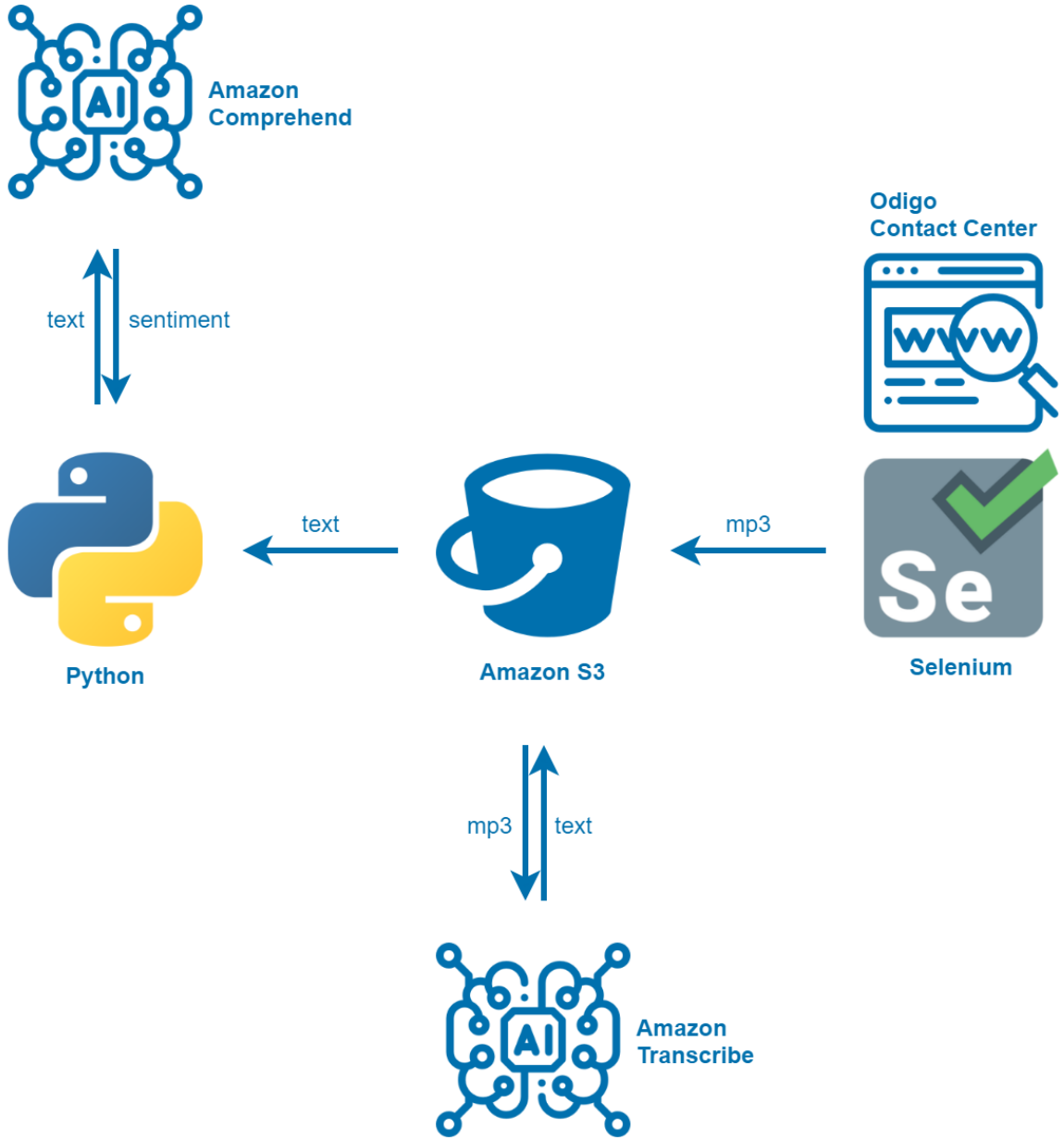




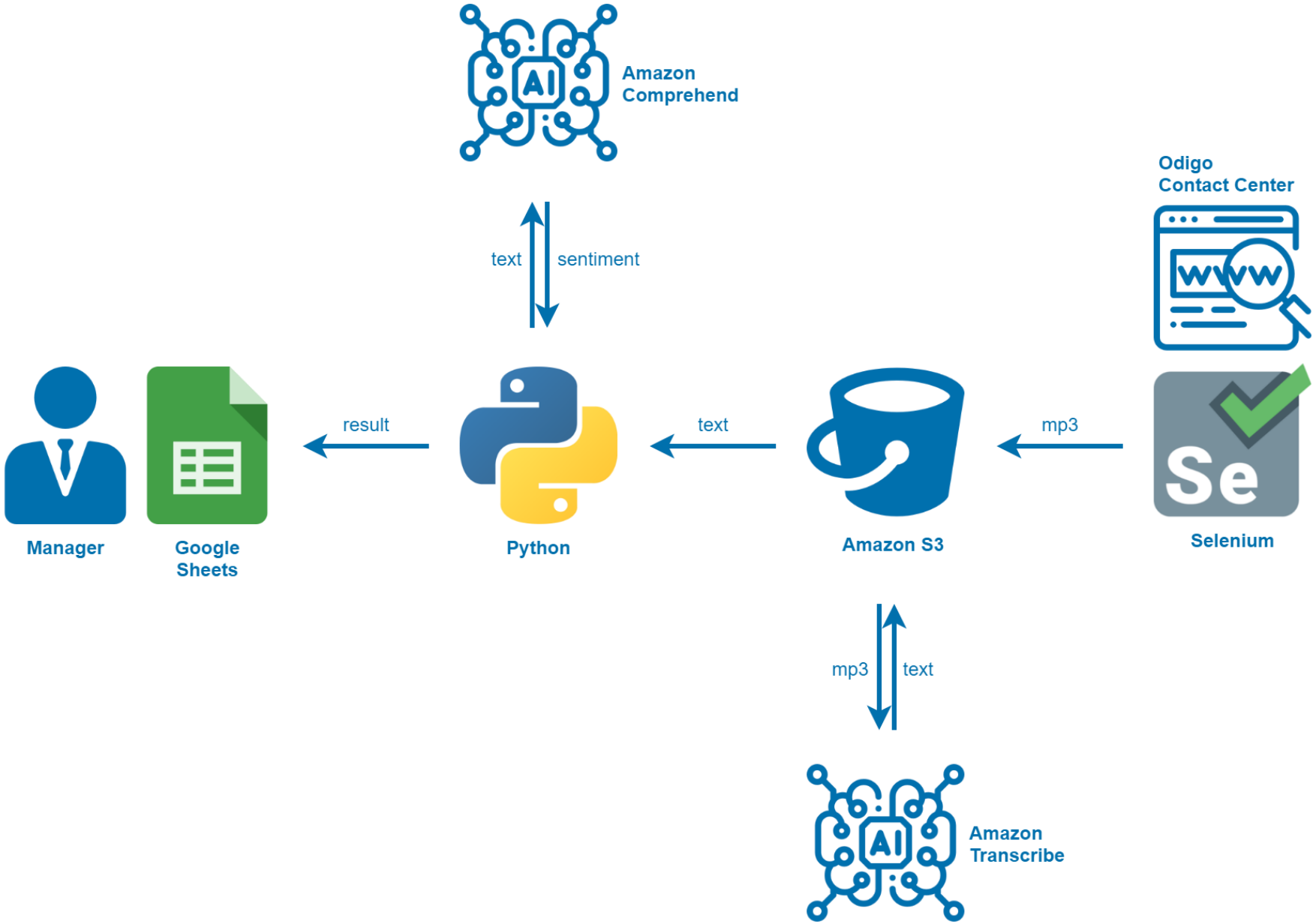










# Web automation

# Requestium Python lib

```
from requestium import Session

s = Session(webdriver_path='/usr/lib/chromium-browser/chromedriver',
            browser='chrome')
```

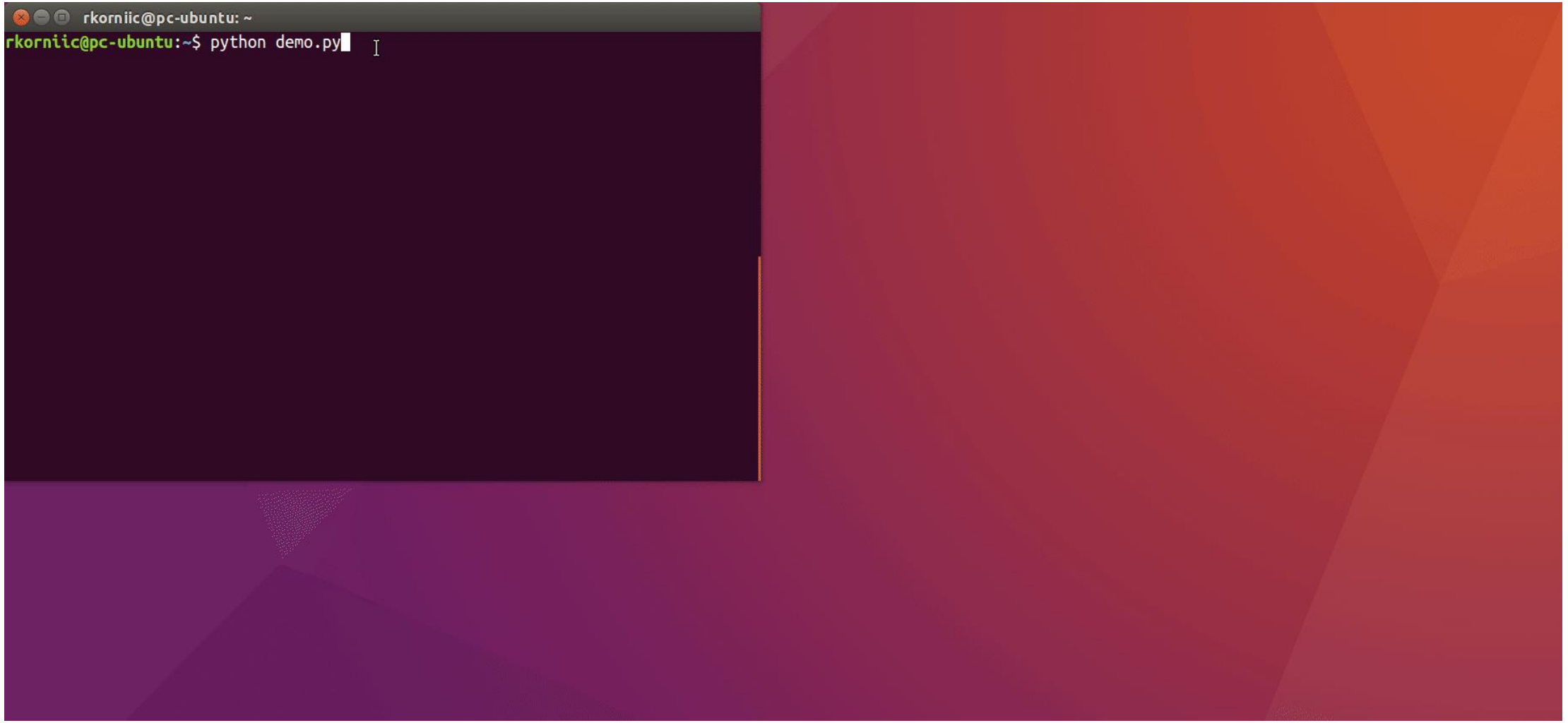

 

# Requestium Python lib
# Login

```
from requestium import Session

s = Session(webdriver_path='/usr/lib/chromium-browser/chromedriver',
            browser='chrome')

url = 'https://profil.wp.pl/login.html?zaloguj=poczta'
s.driver.get(url)
s.driver.ensure_element_by_id('login').send_keys('albert')
s.driver.ensure_element_by_id('password').send_keys('1234')
s.driver.ensure_element_by_id('btnSubmit').click()
```

# Demo





# Requestium Python lib Headless

```
from requestium import Session

s = Session(webdriver_path='/usr/lib/chromium-browser/chromedriver',
            browser='chrome',
            webdriver_options={'arguments': ['headless']})
```

# Requestium Python lib
# Headless. Login

```
from requestium import Session

s = Session(webdriver_path='/usr/lib/chromium-browser/chromedriver',
            browser='chrome',
            webdriver_options={'arguments': ['headless']})

url = 'https://profil.wp.pl/login.html?zaloguj=poczta'
s.driver.get(url)
s.driver.ensure_element_by_id('login').send_keys('albert')
s.driver.ensure_element_by_id('password').send_keys('1234')
s.driver.ensure_element_by_id('btnSubmit').click()
```

# Secure login

# fbi Python lib

## Introduction

The password encryption utility. Do not save your passwords as plaintext!

## GitHub

https://github.com/korniichuk/fbi

## Install fbi Python lib from PyPI

```
$ sudo pip install fbi
```

# fbi Python lib Quickstart

**First**, init the fbi utility:

```
$ fbi init
```

**Second**, encode password in a file:

```
$ fbi encode ~/.key/rightcall.enc
```

**Third**, decode password from a file:

```
from fbi import getpassword
path = '~/.key/rightcall.enc'
passwd = getpassword(path)
```

# Speech recognition

# Speech recognition Part 1

## Amazon Transcribe

**pros**
- conversation mode (with multiple speakers);
- minimum sample rate of 8 KHz;
- custom vocabulary;
- documentation and code samples;
- punctuation.

**cons**
- only asynchronous mode;
- only for audio on Amazon S3 storage;
- need a couple of days for limit increase (10 concurrent transcription jobs).

## Microsoft Azure Speech to Text

**pros**
- real-time;
- conversation mode (with multiple speakers);
- punctuation.

**cons**
- up to 10 s audio with REST API;
- no sample rate of 8 KHz;
- no custom vocabulary (only Custom Speech Service);
- no Python lib;
- not for mp3.

## Google Cloud Speech-to-Text

**pros**
- real-time;
- phone audio mode (beta);
- minimum sample rate of 8 KHz;
- phrase hints;
- documentation and code samples;
- punctuation.

**cons**
- very unstable (Error: None Server unavailable, please try again later);
- Google Cloud Storage for audio longer than 1 minute.

# Speech recognition Part 2

## IBM Watson Speech to Text

**pros**
- real-time;
- conversation mode (with multiple speakers);
- minimum sample rate of 8 KHz;
- punctuation.

**cons**
- no custom vocabulary (only language model customization);
- IBM is competitor of Capgemini.

## Nuance Automatic Speech Recognition

**pros**
- real-time;
- minimum sample rate of 8 KHz.

**cons**
- no conversation mode (with multiple speakers), only Dictation | WebSearch | DTV-search;
- For audio no longer than ~2 minutes;
- not for mp3;
- no punctuation.

## CMU Sphinx

**pros**
- offline mode;
- open-source.

**cons**
- accuracy is very poor;
- out-of-date models;
- not for mp3;
- no punctuation.

# Speech recognition with Amazon Transcribe

```
import boto3

transcribe = boto3.client('transcribe')
```

# Speech recognition with Amazon Transcribe

```
import boto3

transcribe = boto3.client('transcribe')

src = 'https://s3-eu-west-1.amazonaws.com/examplebucket/example.mp3'
r = transcribe.start_transcription_job(
        TranscriptionJobName='example', Media={'MediaFileUri': src},
        MediaFormat='mp3', LanguageCode='en-US')
```

## Transcription jobs

Status: All | Download | Create job

example

1

| Name | Language | Output location | Creation Time | Completion Time | Availability | Status |
|---|---|---|---|---|---|---|

**Empty resources**

No resources to display

Create job

# Customer sentiment analysis

# What is customer sentiment?

**Sentiment** is the emotion behind customer engagement.

When we monitor customer sentiment, we try to measure the tone, context, and feeling from customer actions. Whether a customer completes a purchase, leaves a review, or mentions our company socially, there is always an emotional state connected to their action.

# Sentiment analysis with Amazon Comprehend

```
import boto3

comprehend = boto3.client('comprehend')
```

# Sentiment analysis with Amazon Comprehend

```
import boto3

comprehend = boto3.client('comprehend')

text = 'Hello, World!'
r = comprehend.detect_sentiment(Text=text, LanguageCode='en')
sentiment = r['Sentiment'] # Neutral | Positive | Negative | Mixed
```

Amazon Comprehend

Try Amazon Comprehend
Organization
Analysis



# Try Amazon Comprehend Info

See how Amazon Comprehend recognizes entities, extracts key phrases, detects language, and analyzes sentiment. To analyze your text, type or paste it in the text box.

## Input text

Supported languages

Amazon.com, Inc. is located in Seattle, WA and was founded July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable Seattle - based companies are Starbucks and Boeing.

272 of 5000 characters used.

Clear text Analyze

## Insights Info

Entities | Key phrases | Language | Sentiment | Syntax

**Analyzed text**

Amazon.com, Inc. is located in Seattle, WA and was founded July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable Seattle - based companies are Starbucks and Boeing.

▼ Results

| Entity | Category | Count | Confidence |
|---|---|---|---|
| Amazon.com, Inc | Organization | 1 | 0.90 |
| Boeing | Organization | 1 | 0.99+ |

# Google Sheets

# Google Sheets

```
from apiclient.discovery import build
from httplib2 import Http
from oauth2client import file

store = file.Storage('token.json')
creds = store.get()
service = build('sheets', 'v4', http=creds.authorize(Http()))
```

# Google Sheets
# Append values

```
from apiclient.discovery import build
from httplib2 import Http
from oauth2client import file

store = file.Storage('token.json')
creds = store.get()
service = build('sheets', 'v4', http=creds.authorize(Http()))

spreadsheet_id = '1wsDYxzfdDhikAehg9k0Xp_FgRLUf-0sB79sWO8V21Lw'
body = {'values': [['value1', 'value2', 'value3', 'value4']]}
r = service.spreadsheets().values().append(spreadsheetId=spreadsheet_id,
        range='Sheet1', valueInputOption='RAW', body=body).execute()
```

Untitled spreadsheet

File Edit View Insert Format Data Tools Add-ons Help Last edit was seconds ago

SHARE

100% £ % .0 .00 123 Arial 10

fx value1

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | value1 | value2 | value3 | value4 | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | | | | | | | |
| 23 | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | |
| 25 | | | | | | | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | | | | | |
| 28 | | | | | | | | | | | | | | | | | | |
| 29 | | | | | | | | | | | | | | | | | | |
| 30 | | | | | | | | | | | | | | | | | | |
| 31 | | | | | | | | | | | | | | | | | | |
| 32 | | | | | | | | | | | | | | | | | | |
| 33 | | | | | | | | | | | | | | | | | | |
| 34 | | | | | | | | | | | | | | | | | | |
| 35 | | | | | | | | | | | | | | | | | | |
| 36 | | | | | | | | | | | | | | | | | | |
| 37 | | | | | | | | | | | | | | | | | | |
| 38 | | | | | | | | | | | | | | | | | | |
| 39 | | | | | | | | | | | | | | | | | | |
| 40 | | | | | | | | | | | | | | | | | | |
| 41 | | | | | | | | | | | | | | | | | | |
| 42 | | | | | | | | | | | | | | | | | | |

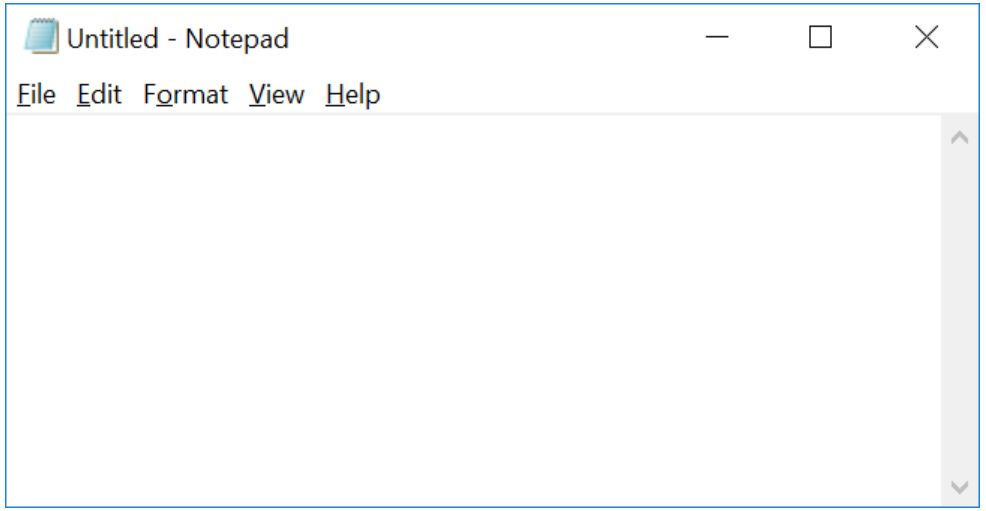
Sheet1

# GUI text-based automation

# Text-based automation

```
from pywinauto.application import Application

app = Application().start('notepad.exe')
```

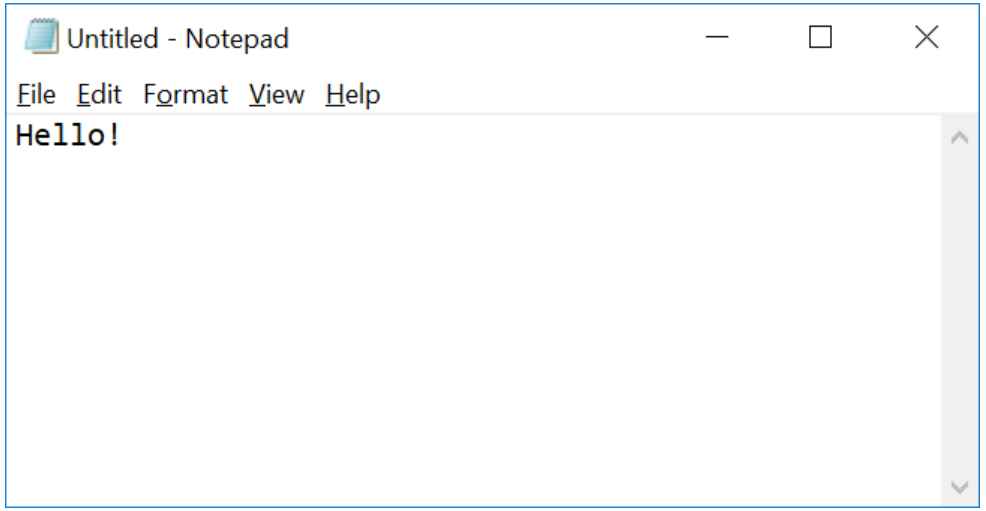




# Text-based automation

```
from pywinauto.application import Application

app = Application().start('notepad.exe')

app.UntitledNotepad.Edit.type_keys('Hello!')
```

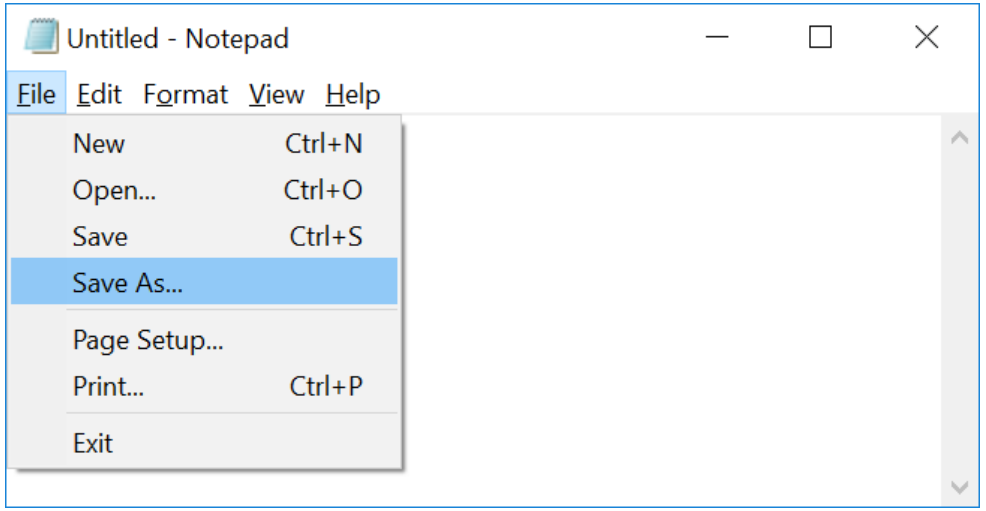




# Text-based automation

```
from pywinauto.application import Application

app = Application().start('notepad.exe')

app.UntitledNotepad.Edit.type_keys('Hello!')

app.UntitledNotepad.menu_select('File->SaveAs')
```

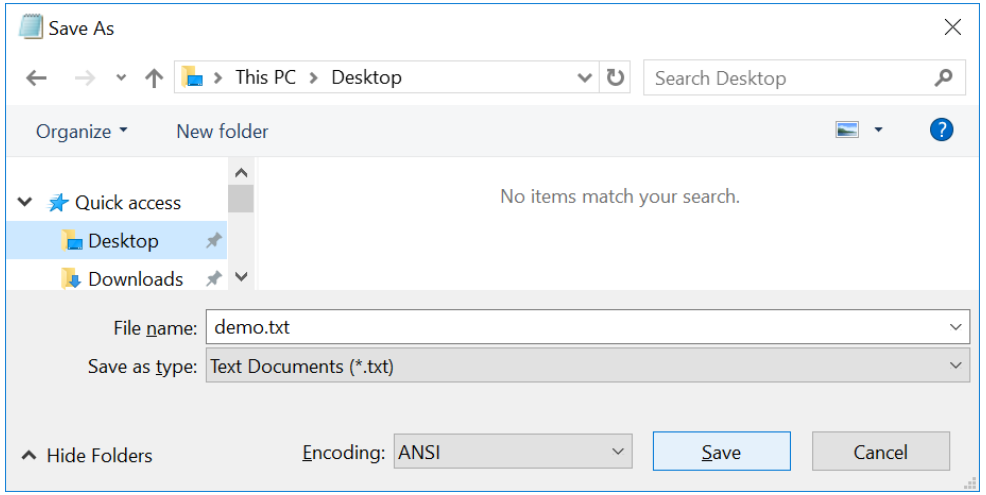




# Text-based automation

```
from pywinauto.application import Application

app = Application().start('notepad.exe')

app.UntitledNotepad.Edit.type_keys('Hello!')

app.UntitledNotepad.menu_select('File->SaveAs')

app.SaveAs.Edit1.set_text('demo.txt')
app.SaveAs.Save.click()
```

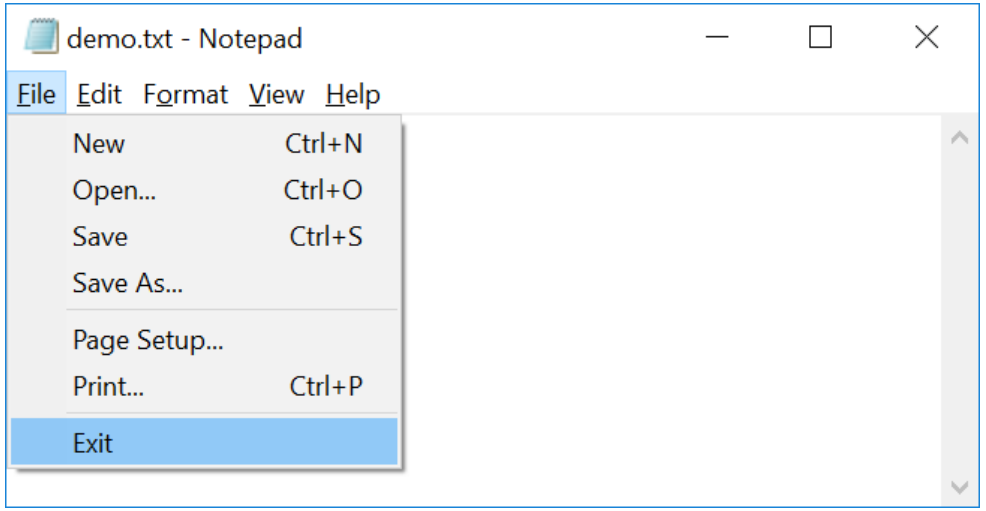




# Text-based automation

```
from pywinauto.application import Application

app = Application().start('notepad.exe')

app.UntitledNotepad.Edit.type_keys('Hello!')

app.UntitledNotepad.menu_select('File->SaveAs')

app.SaveAs.Edit1.set_text('demo.txt')
app.SaveAs.Save.click()

app.demotxtNotepad.menu_select('File->Exit')
```

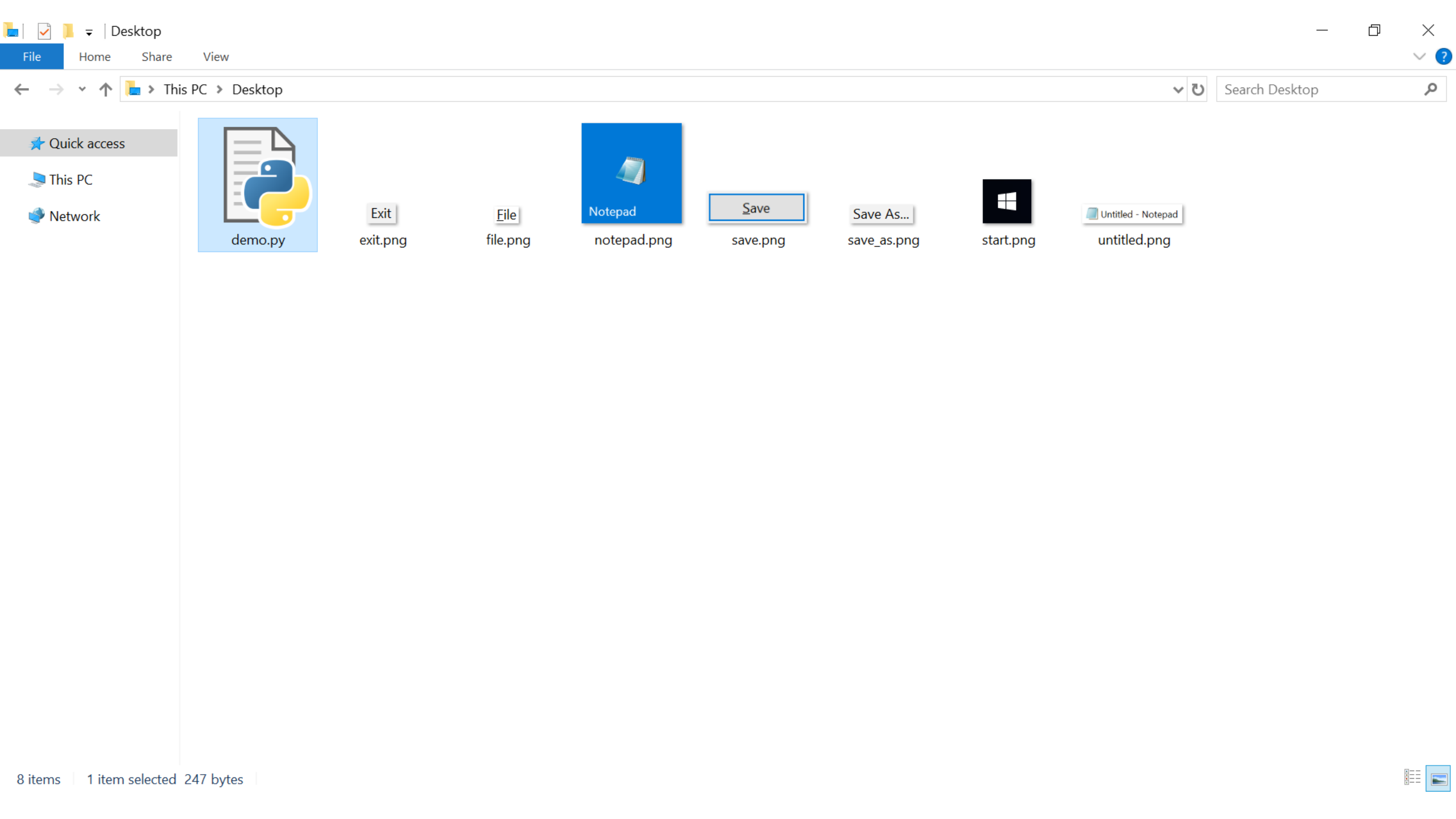




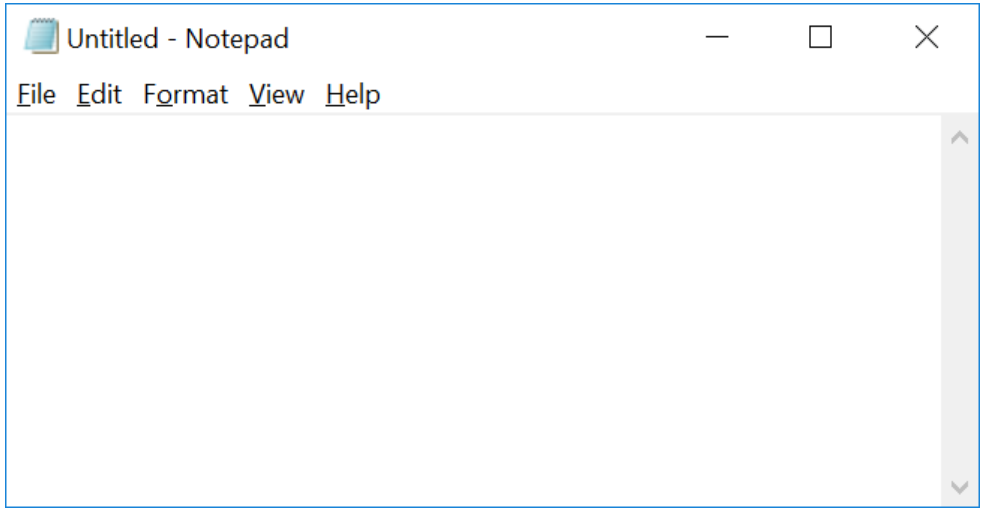
# GUI image-based automation


Desktop
File
Home
Share
View
This PC
Desktop
Search Desktop
Quick access
This PC
Network
demo.py
Exit
exit.png
File
file.png
Notepad
notepad.png
Save
save.png
Save As...
save_as.png
start.png
Untitled - Notepad
untitled.png
8 items
1 item selected 247 bytes

# Image-based automation

```
from lackey import *

click('start.png')
click('notepad.png')
```

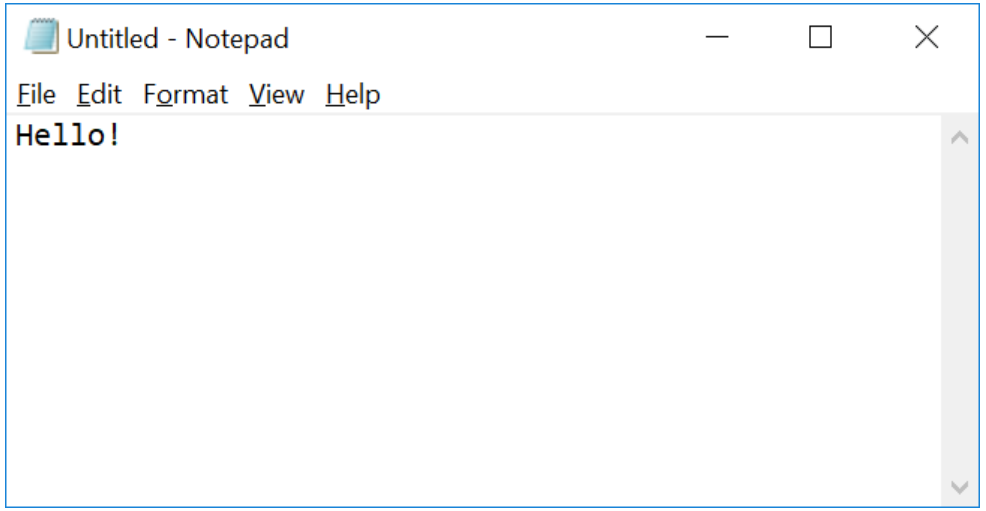




# Image-based automation

```
from lackey import *

click('start.png')
click('notepad.png')

wait('untitled.png')
type('Hello!')
```

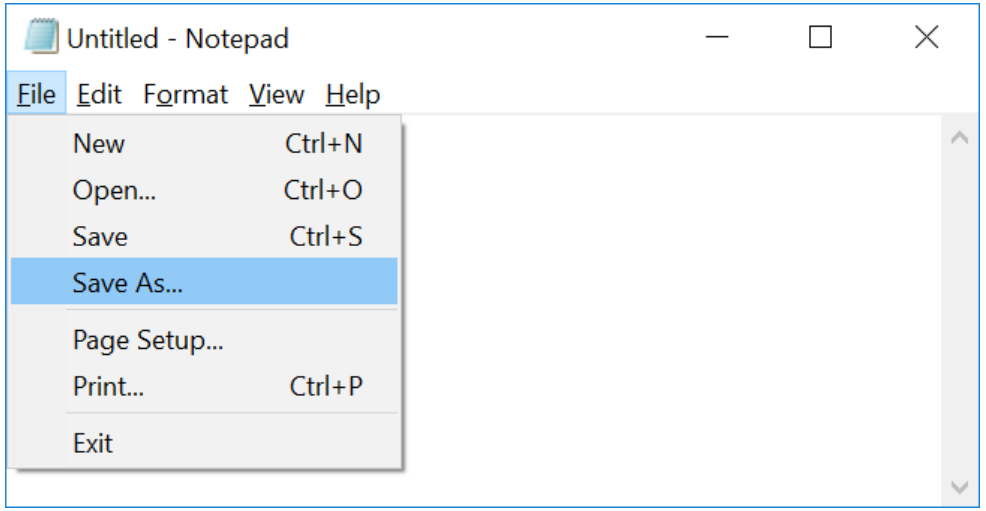




# Image-based automation

```
from lackey import *

click('start.png')
click('notepad.png')

wait('untitled.png')
type('Hello!')

click('file.png')
click('save_as.png')
```

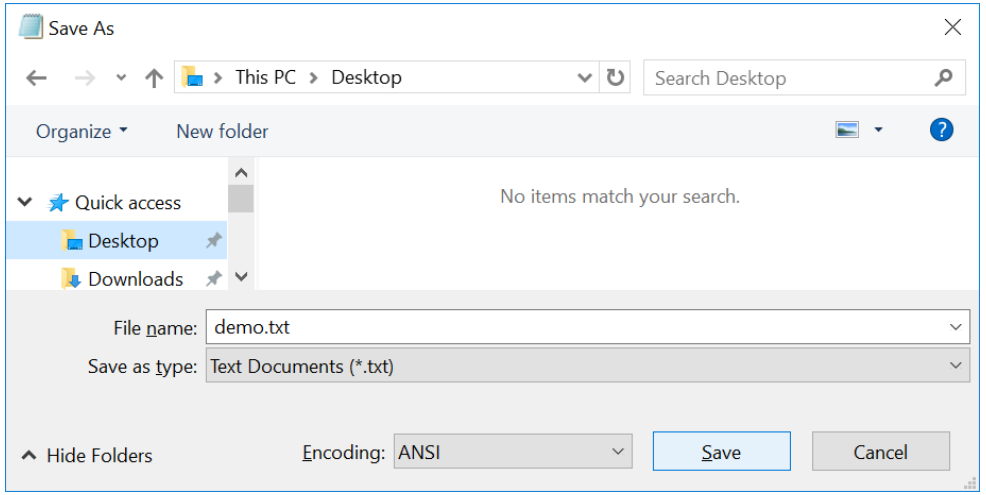




# Image-based automation

```
from lackey import *

click('start.png')
click('notepad.png')

wait('untitled.png')
type('Hello!')

click('file.png')
click('save_as.png')

type('demo.txt')
click('save.png')
```

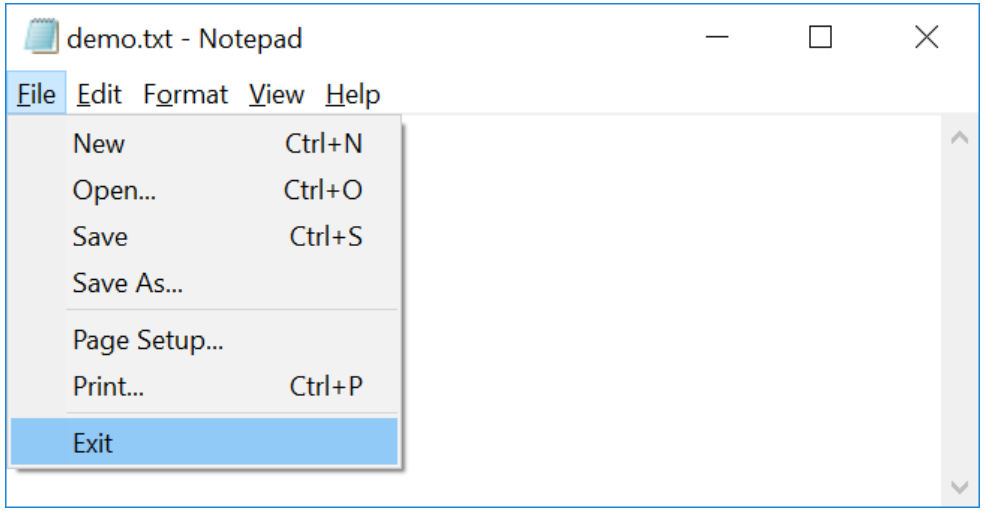




# Image-based automation

```
from lackey import *

click('start.png')
click('notepad.png')

wait('untitled.png')
type('Hello!')

click('file.png')
click('save_as.png')

type('demo.txt')
click('save.png')

click('file.png')
click('exit.png')
```





# IA vs RPA

# RPA

RPA (**Robotic Process Automation**) tools perform "if, then, else" statements on structured data, typically using a combination of user interface (UI) interactions, or by connecting to APIs to drive client servers, mainframes or HTML code.

An RPA tool operates by mapping a process in the RPA tool language for the software "robot" to follow, with runtime allocated to execute the script by a control dashboard.

Source: Gartner

# RPA tool vendors

## Specialist RPA software providers

- AutomationEdge
- **Automation Anywhere**
- **Blue Prism**
- Contextor
- EnableSoft
- Epiance
- Kryon Systems
- OpenConnect
- Softomotive
- **UiPath**

## Software providers with multiple software products, including RPA

- AntWorks
- HelpSystems
- Infosys (EdgeVerve Systems)
- Jacada
- Kofax
- Nice
- Pegasystems
- Perpetuuiti
- Verint
- Redwood Software
- Servicetrace
- SpiceCSM
- Winshuttle
- **WorkFusion**

## IT or BPO service providers that provide proprietary RPA software platforms

- Another Monday
- Cognizant
- Conduent
- Sutherland
- Syntel
- Tech Mahindra

# IA

IA (**Intelligent Automation**) automation accomplishes a task repeatedly without human intervention.

IA includes technologies such as machine learning, natural language processing/understanding, optical character recognition.

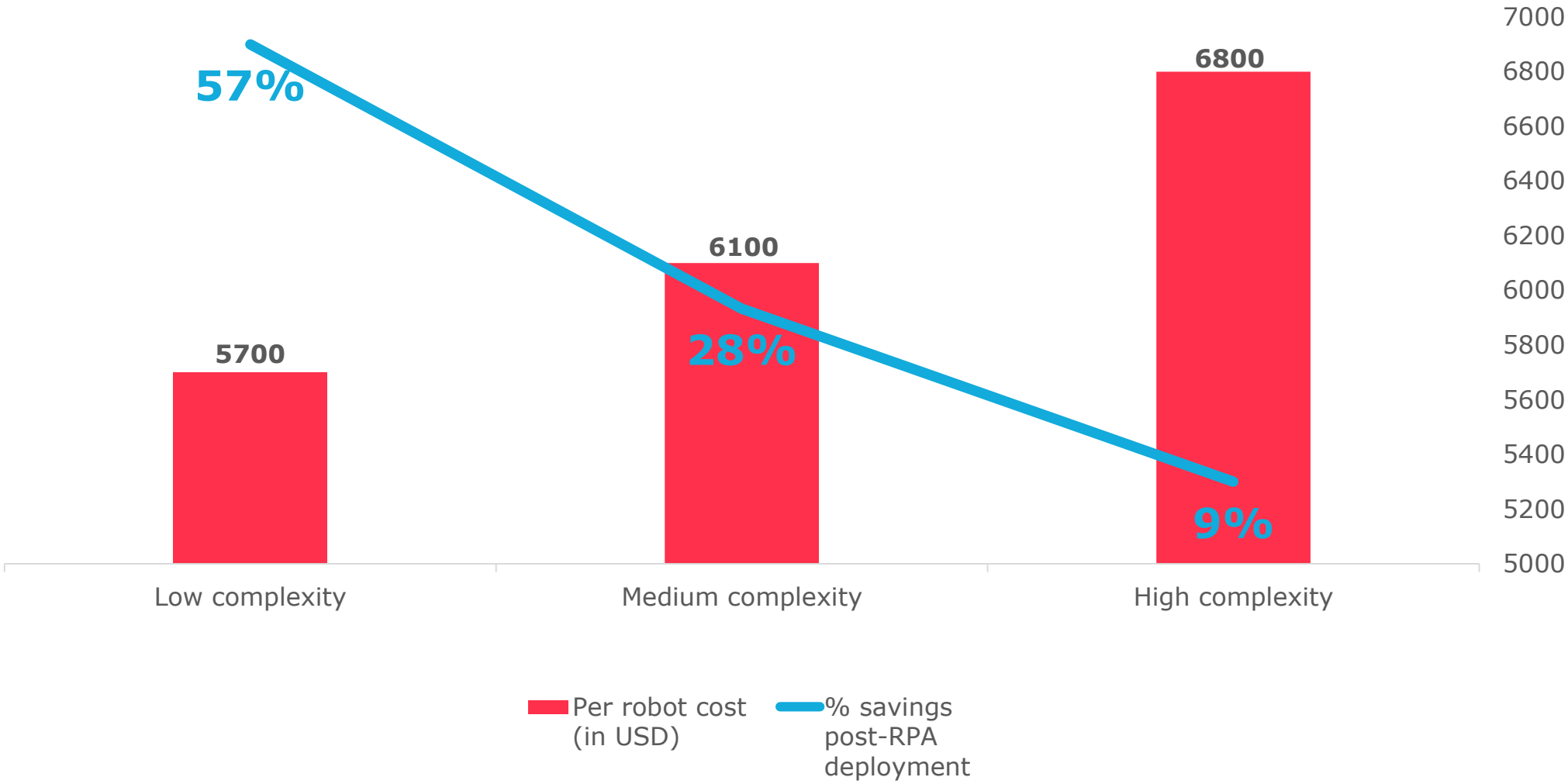
# IA vs RPA

## Process complexity and automation opportunities in financial services

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Payment processing | Tax calculation & filing | Reconciliations | Invoice processing | Trade processing & settlements | Funds transfer & review | Account closure | Loan account & credit line setup |
| Customer account opening | Interest payments | Setting credit limits | Periodic accounting books closure | Loan disbursements | KYC/AML process | Loan applications processing | Updating accounts details |
| Regulatory reporting | Card account interchange & settlement | Trade confirmations & settlements | Transactions authorization | Tax claims management | Document processing | Overdraft/ advances provision management | Credit underwriting |
| **Foreclosure audit review & resolution** | **Collateral review** | **Personalized offers & promotions** | **Risk exposure monitoring** | **Internal comm monitoring** | **Fraudulent transaction detection** | **Customer queries** | **Personalized financial advice** |

# RPA. Variation of cost and savings




**Scenario:** Process with 100 FTEs pre-RPA deployment

# BTW

## Windows

```
> echo Hello! > demo.txt
```

## Linux

```
$ echo Hello! > demo.txt
```

## Python

```
with open ('demo.txt', 'w') as f:
    f.write('Hello!')
```

# bit.ly/RCBIELSKO

## About Capgemini

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of 200,000 team members in over 40 countries. The Group reported 2017 global revenues of EUR 12.8 billion.

Learn more about us at

www.capgemini.com

**People matter, results count.**


**Ruslan Korniichuk**

Senior Consultant
Capgemini
ruslan.korniichuk@capgemini.com
+48 791 065 877

**Lukasz Stefanik**

Cognitive Solutions Lead
Capgemini
lukasz.stefanik@capgemini.com
+48 728 406 311