Big Data Analysis

META project data:

- **Project customer:** IT School "GoIT"
- **Project name:** Stack overflow software developer survey analysis (The final project of the Python block)
- **Project goal:** Working with large datasets, descriptive statistics and data visualization.
- **DataSet Name:** Stack Overflow Developer Survey 2025
- **Project contractor:** Isachenko Andrii Junior Data Analyst
- **Tools:** Python (Pandas, NumPy, MatPlotLib, SeaBorn), Jupyter Notebook, Goole Colab
- **Data received date:** 2026-XX-XX
- **Analysis completion date:** 2026-XX-XX

1. Importing python libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Data acquisition

```
# Loading data from CSV files (pre-upload the dataset to Google Drive and provide access for Google Colab):
# Adding links to data files
srp_df_url = '/content/drive/MyDrive/ISAO/data/survey_results_public.csv'


# srp - shortened from survey_results_public
srs_df_url = '/content/drive/MyDrive/ISAO/data/survey_results_schema.csv'
# srs - shortened from survey_results_schema

# Downloading data for further analysis
srp_df = pd.read_csv(srp_df_url, encoding='utf-8', low_memory=False)
# srp - shortened from survey_results_public
srs_df = pd.read_csv(srs_df_url, encoding='utf-8')
# srs - shortened from survey_results_schema
```

```
# Checking the result
# Retrieve data from survey_results_schema
srs_df.head()
```

	qid	qname	question	type	sub	sq_id
0	QID18	TechEndorse_1	What attracts you to a technology or causes yo...	RO	AI integration or AI Agent capabilities	1.0
1	QID18	TechEndorse_2	What attracts you to a technology or causes yo...	RO	Easy-to-use API	2.0
2	QID18	TechEndorse_3	What attracts you to a technology or causes yo...	RO	Robust and complete API	3.0
3	QID18	TechEndorse_4	What attracts you to a technology or causes yo...	RO	Customizable and manageable codebase	4.0
4	QID18	TechEndorse_5	What attracts you to a technology or causes yo...	RO	Reputation for quality	5.0

Подальші дії: New interactive sheet

```
# Checking the result
# Retrieve data from survey_results_public
srp_df.head()
```

	ResponseId	MainBranch	Age	EdLevel	Employment	EmploymentAddl	WorkExp	LearnCodeChoose	LearnCode	LearnCodeAI	...
0	1	I am a developer by profession	25-34 years old	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	Employed	Caring for dependents (children, elderly, etc.)	8.0	Yes, I am not new to coding but am learning ne...	Online Courses or Certification (includes all ...	Yes, I learned how to use AI-enabled tools for...	...
1	2	I am a developer by profession	25-34 years old	Associate degree (A.A., A.S., etc.)	Employed	NaN	2.0	Yes, I am not new to coding but am learning ne...	Online Courses or Certification (includes all ...	Yes, I learned how to use AI-enabled tools for...	...
2	3	I am a developer by profession	35-44 years old	Bachelor's degree (B.A., B.S., B.Eng., etc.)	Independent contractor, freelancer, or self-em...	None of the above	10.0	Yes, I am not new to coding but am learning ne...	Online Courses or Certification (includes all ...	Yes, I learned how to use AI-enabled tools for...	...
3	4	I am a developer by profession	35-44 years old	Bachelor's degree (B.A., B.S., B.Eng., etc.)	Employed	None of the above	4.0	Yes, I am not new to coding but am learning ne...	Other online resources (e.g. standard search, ...	Yes, I learned how to use AI-enabled tools for...	...
4	5	I am a developer by profession	35-44 years old	Master's degree (M.A., M.S., M.Eng., MBA, etc.)	Independent contractor, freelancer, or self-em...	Caring for dependents (children, elderly, etc.)	21.0	No, I am not new to coding and did not learn n...	NaN	Yes, I learned how to use AI-enabled tools for...	...

5 rows × 172 columns

```
# We conclude that the dataset was loaded successfully.
# We take into account the fact that the received data is already prepared and pre-cleaned.
```

3. We are analyzing the data

3.1 We cover the total number of respondents

```
# The number of respondents is covered and the result is displayed
num_respondents = srp_df['ResponseId'].nunique()
print("Number of respondents:", num_respondents)
```

```
Number of respondents: 49191
```

3.2 Analysis of the recurrence of respondents' responses

```
# We can extract the food list from the file
qnames = set(srs_df["qname"].dropna().unique())

# We know the span of the columns by 'qnames', both in the scheme and in the sample itself
available_questions = qnames.intersection(set(srp_df.columns))

# You can see the rows with blanks in the selected columns
filtered = srp_df[list(available_questions)].dropna()

# We appreciate the number of respondents without gaps
result = filtered.shape[0]
print("Number of respondents who contributed to all meals:", result)
```

```
Number of respondents who contributed to all meals: 0
```

3.3 Statistical analysis of respondents' information

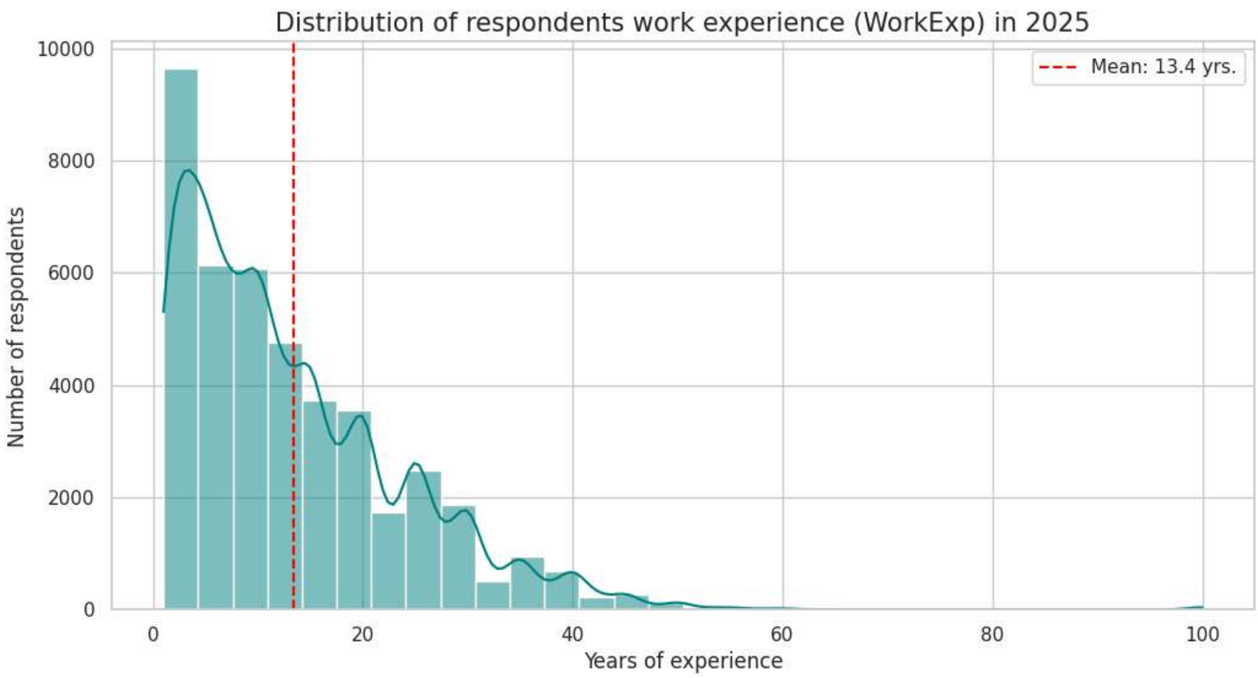
```
# Let's remove gaps
workexp = srp_df["WorkExp"].dropna()
# The number format is translated and re-checked to ensure that no gaps appear after conversion
workexp = pd.to_numeric(workexp, errors="coerce").dropna()

# Calculable statistical indicators
mean = round(workexp.mean(), 2) #immediately rounded up
median = workexp.median()
mode = workexp.mode().iloc[0]
```

```
# The result is displayed
print("Statistical analysis of respondents' data:")
print(f'Mean: {mean}')
print(f'Median: {median}')
print(f'Mode: {mode}')
```

```
Statistical analysis of respondents' data:
Mean: 13.37
Median: 10.0
Mode: 10.0
```

Let's create a graph of the distribution of respondents' experience.

```
# Style settings
sns.set_theme(style="whitegrid")
plt.figure(figsize=(12, 6))

# Graph construction
sns.histplot(srp_df['WorkExp'].dropna(), bins=30, kde=True, color='teal')

plt.title('Distribution of respondents work experience (WorkExp) in 2025', fontsize=15)
plt.xlabel('Years of experience', fontsize=12)
plt.ylabel('Number of respondents', fontsize=12)

mean_exp = srp_df['WorkExp'].mean()
plt.axvline(mean_exp, color='red', linestyle='--', label=f'Mean: {mean_exp:.1f} yrs.')
plt.legend()

# Save the graph as a file
plt.savefig('work_experience_dist.png', dpi=300, bbox_inches='tight')

# Display (called after saving)
plt.show()
```



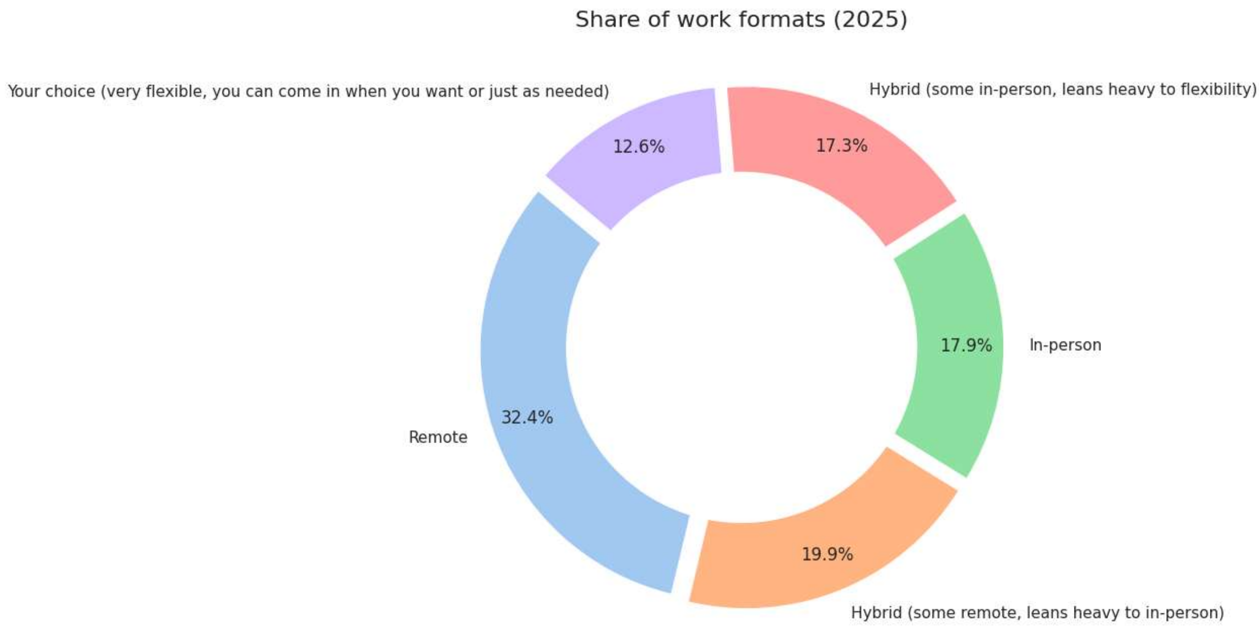
3.4 Remote work analysis

```
# We filter only those who work remotely
remote_workers = srp_df[srp_df["RemoteWork"].str.contains("remote", case=False, na=False)]
count_remote = remote_workers.shape[0]

# We output the result
print(f'Number of respondents who work remotely: {count_remote}')
```

```
Number of respondents who work remotely: 17663
```

3.5 We are building a schedule for the distribution of work formats among respondents

```
# Data preparation
work_format_counts = srp_df['RemoteWork'].value_counts()

# Checking whether there is data for construction
if work_format_counts.empty:
    print("There is no data in the 'RemoteWork' column.")
else:
    labels = work_format_counts.index
    values = work_format_counts.values

    # DYNAMIC SETTINGS:
    # Create an 'explode' list of the same length as the number of categories
    explode = [0.05] * len(values)

    # We create a color palette exactly according to the number of values
    colors = sns.color_palette('pastel', len(values))

    plt.figure(figsize=(8, 8))

    # Building a diagram
    plt.pie(values,
            labels=labels,
            autopct='%1.1f%%',
            startangle=140,
            colors=colors,
            pctdistance=0.85,
            explode=explode) # Now the length is always correct

    # Transforming Pie y Donut
    centre_circle = plt.Circle((0,0), 0.70, fc='white')
    fig = plt.gcf()
    fig.gca().add_artist(centre_circle)

    plt.title('Share of work formats (2025)', fontsize=16)

    # Preservation
    plt.savefig('remote_work_pie_chart.png', dpi=300, bbox_inches='tight')
    plt.show()
```



3.6 Determining Python popularity

```
# Identify respondents who have Python in their list of programming languages
python_langs = srp_df["LanguageHaveWorkedWith"].str.contains("Python", case=False, na=False)

# Calculating the percentage
```

```
percent_python = round((python_langs.sum() / len(srp_df)) * 100, 1)

# We output the result
print(f"Percentage of respondents who program in Python: {percent_python}%")
```

```
Percentage of respondents who program in Python: 37.5%
```

3.7 Analysis of ways to learn programming

```
# Applying a mask
mask_online = srp_df["LearnCode"].str.contains("Online course", case=False, na=False)

# We filter only those who studied through online courses
online_learners = srp_df[mask_online]

# We are counting the number of such respondents.
count_learners = online_learners.shape[0]

# We output the result
print(f"Number of respondents who studied through online courses: {count_learners}")
```

```
Number of respondents who studied through online courses: 21212
```

3.8 Geographic analysis of Python developer compensation

```
# We filter out respondents who program in Python
python_srp_df = srp_df[srp_df["LanguageHaveWorkedWith"].str.contains("Python", case=False, na=False)].copy()

subset = python_srp_df[["Country", "ConvertedCompYearly"]].copy()

# Convert to numeric format and remove spaces
subset["ConvertedCompYearly"] = pd.to_numeric(subset["ConvertedCompYearly"], errors="coerce")
subset = subset.dropna(subset=["ConvertedCompYearly"])

# Group by country and calculate the mean and median
comp_stats = (
    subset
    .groupby("Country")["ConvertedCompYearly"]
    .agg(mean_compensation="mean", median_compensation="median")
    .reset_index()
)

# Rounding the value
comp_stats["mean_compensation"] = comp_stats["mean_compensation"].round(2)
comp_stats["median_compensation"] = comp_stats["median_compensation"].round(2)

# Виводимо результат
print("Geographic analysis of Python developer compensation:")
comp_stats
```

```
Geographic analysis of Python developer compensation:
```

	Country	mean_compensation	median_compensation
0	Afghanistan	22328.67	1000.0
1	Albania	47217.60	50000.0
2	Algeria	20187.29	7088.0
3	Andorra	226103.50	226103.5
4	Antigua and Barbuda	1.00	1.0
...	...	...	...
148	Venezuela, Bolivarian Republic of...	9908.65	3000.0
149	Viet Nam	218837.17	8254.0
150	Yemen	32929.50	23672.0
151	Zambia	5424.25	3206.0
152	Zimbabwe	34000.00	25500.0

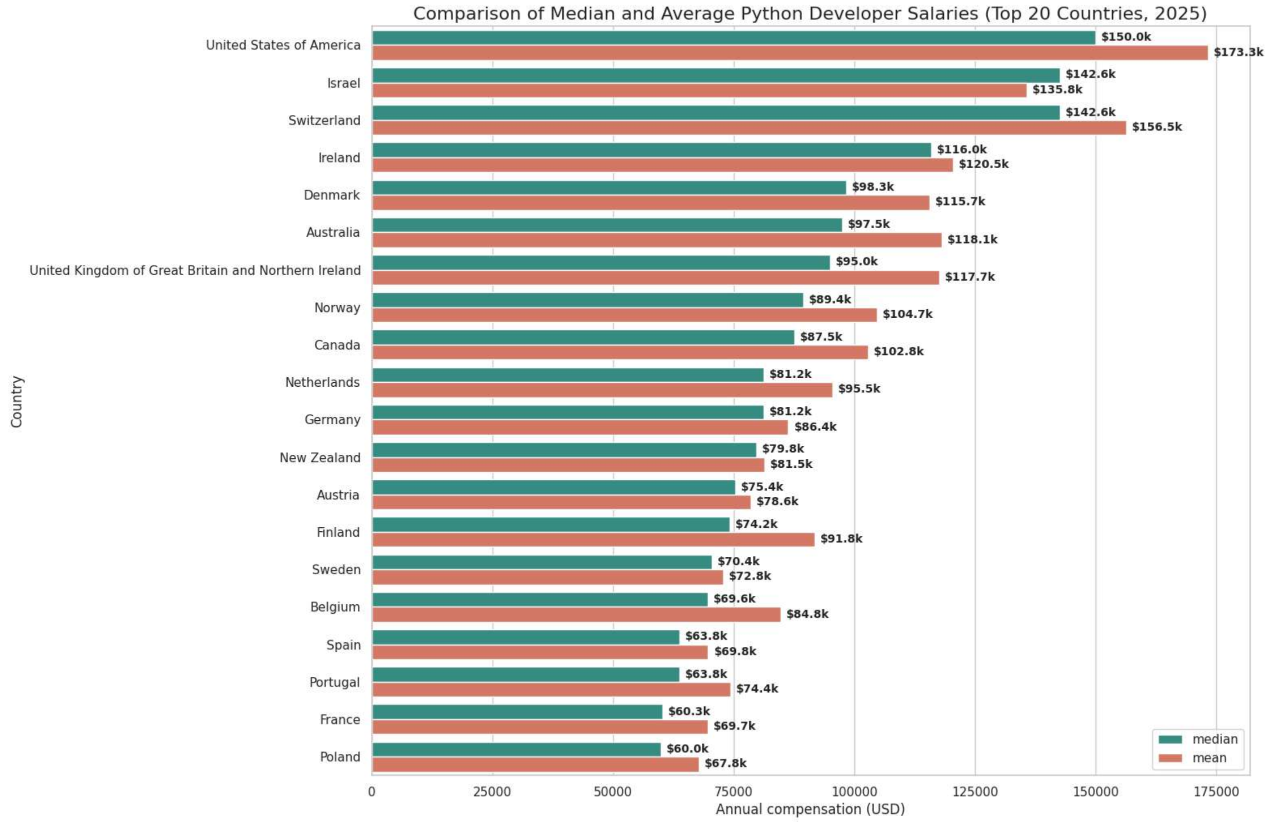
153 rows × 3 columns

Подальші дії: New interactive sheet

```
# Save the result to a separate CSV file
comp_stats.to_csv("python_comp_stats.csv", index=False, encoding="utf-8")
print("File successfully saved as python_comp_stats.csv")
```

Let's build a graph comparing the median and average salaries of Python developers by country

```
# Data preparation
python_devs = srp_df[
    (srp_df['LanguageHaveWorkedWith'].str.contains('Python', na=False)) &
    (srp_df['ConvertedCompYearly'].notnull())
].copy()

# Filter by number of respondents (>50 for validity)
country_counts = python_devs['Country'].value_counts()
top_countries = country_counts[country_counts > 50].index
python_filtered = python_devs[python_devs['Country'].isin(top_countries)]

# Aggregation: calculate the Median and Mean simultaneously
stats = python_filtered.groupby('Country')['ConvertedCompYearly'].agg(['median', 'mean'])

# Sort by median and take the TOP-20
stats_sorted = stats.sort_values(by='median', ascending=False).head(20)

# Preparing for visualization (transforming the table for Seaborn)
# Convert Indices (Countries) into a column and expand metrics (median/mean) into a single column
stats_plot = stats_sorted.reset_index().melt(id_vars='Country', var_name='Metric', value_name='Salary')

# Graph construction
plt.figure(figsize=(14, 12))
sns.set_theme(style="whitegrid")

# Creating grouped columns
ax = sns.barplot(data=stats_plot,
                 y='Country',
                 x='Salary',
                 hue='Metric',
                 palette={'median': '#2a9d8f', 'mean': '#e76f51'})

# Adding value labels
for p in ax.patches:
    width = p.get_width()
    if width > 0: # Add text only if the column exists
        ax.annotate(f'${width/1000:.1f}k',
                    (width, p.get_y() + p.get_height() / 2.),
                    ha='left', va='center',
                    xytext=(5, 0),
                    textcoords='offset points',
                    fontsize=10, fontweight='bold')

plt.title('Comparison of Median and Average Python Developer Salaries (Top 20 Countries, 2025)', fontsize=16)
plt.xlabel('Annual compensation (USD)', fontsize=12)
plt.ylabel('Country', fontsize=12)
plt.legend()

# Preservation
plt.savefig('python_median_vs_mean.png', dpi=300, bbox_inches='tight')
plt.show()
```



3.9 Analysis of the education of the highest paid specialists

```
# 2. We are closing gaps in compensation and education
srp_df_clean = srp_df.dropna(subset=["ConvertedCompYearly", "EdLevel"])

# Sort by compensation
srp_df_sorted = srp_df_clean.sort_values("ConvertedCompYearly", ascending=False)

# We select the top 5 respondents
srp_top_5 = srp_df_sorted.head(5)

# We output the result
srp_top_5_education = srp_top_5[["ConvertedCompYearly", "EdLevel", ]]
print("Education levels of TOP 5 respondents with the highest compensation:")
srp_top_5_education
```

Education levels of TOP 5 respondents with the highest compensation:

	ConvertedCompYearly	EdLevel
34267	50000000.0	Associate degree (A.A., A.S., etc.)
28700	33552715.0	Master’s degree (M.A., M.S., M.Eng., MBA, etc.)
43143	18387548.0	Associate degree (A.A., A.S., etc.)
35353	15430267.0	Bachelor’s degree (B.A., B.S., B.Eng., etc.)
45971	13921760.0	Master’s degree (M.A., M.S., M.Eng., MBA, etc.)

Подальші дії: New interactive sheet

4.0* Analysis of Python popularity by age category

```
# Creating a Boolean column: Does the respondent program in Python
srp_df["python_user"] = srp_df["LanguageHaveWorkedWith"].str.contains("Python", case=False, na=False)

# We are removing gaps in the age category
srp_df_clean = srp_df.dropna(subset=["Age"])

# We group by age categories.
srp_result = (
    srp_df_clean.groupby("Age")["python_user"]
    .mean() * 100
)

# Round and tabulate
srp_result = srp_result.round(1).reset_index(name="Percent_Python")

# We output the result
srp_result
```

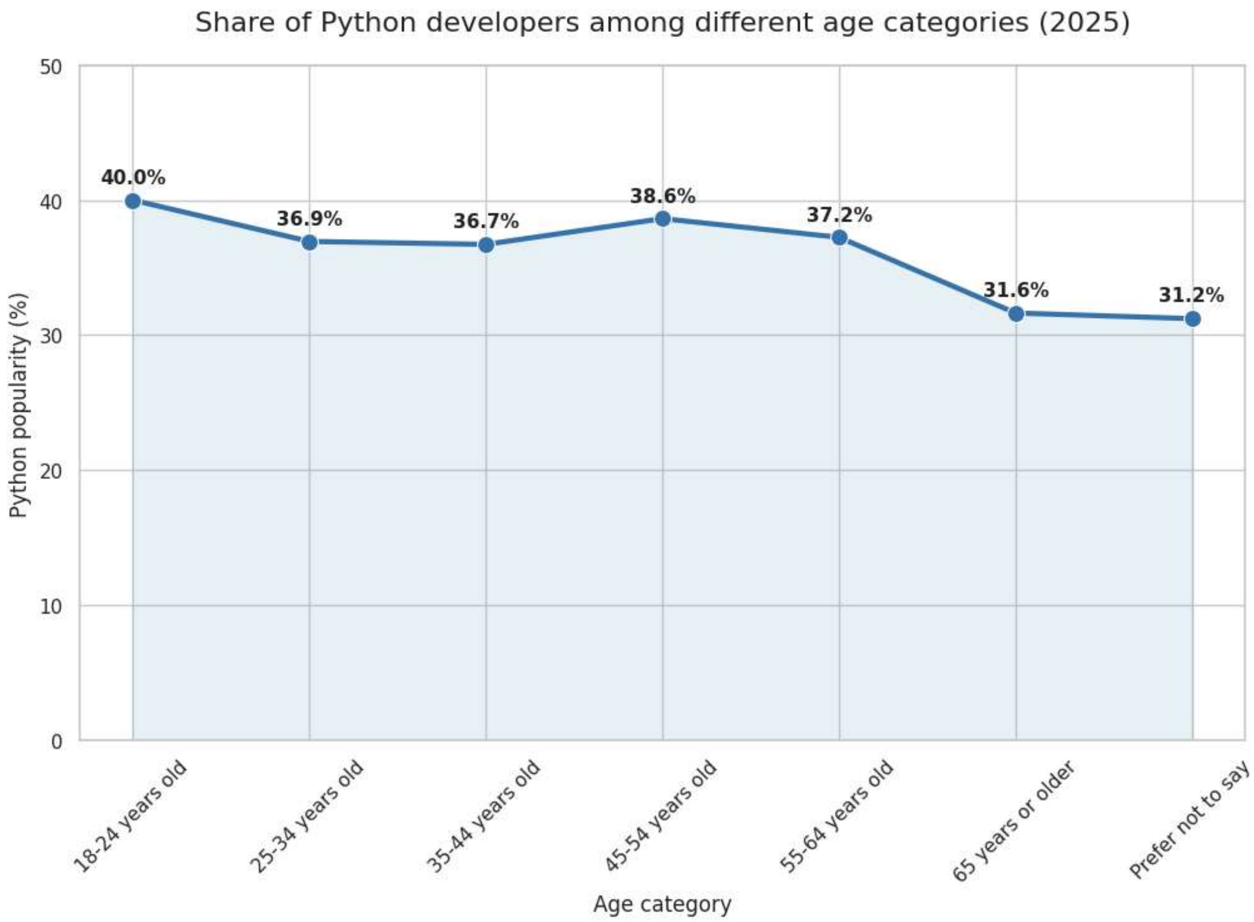
	Age	Percent_Python
0	18-24 years old	40.0
1	25-34 years old	36.9
2	35-44 years old	36.7
3	45-54 years old	38.6
4	55-64 years old	37.2
5	65 years or older	31.6
6	Prefer not to say	31.2

Подальші дії: New interactive sheet

Let's build a graph of Python popularity among age categories

```
# Create a checkbox column: True if Python is in the language list
srp_df['UsesPython'] = srp_df['LanguageHaveWorkedWith'].str.contains('Python', na=False)

# Group by age and calculate the percentage (average of Boolean values * 100)
age_python_stats = srp_df.groupby('Age')['UsesPython'].mean().sort_index() * 100

# Determine the correct order of age categories (optional if they are not sorted)
# Usually in a survey they go from youngest to oldest.
age_order = [
    'Under 18 years old',
    '18-24 years old',
    '25-34 years old',
    '35-44 years old',
    '45-54 years old',
    '55-64 years old',
    '65 years or older',
    'Prefer not to say'
]

# Filter existing categories to avoid errors if one does not exist
age_order = [age for age in age_order if age in age_python_stats.index]
age_python_stats = age_python_stats.reindex(age_order)

# Visualization
plt.figure(figsize=(12, 7))
sns.set_theme(style="whitegrid")

# Create a line graph with markers (it shows the trend better)
ax = sns.lineplot(x=age_python_stats.index, y=age_python_stats.values,
                  marker='o', markersize=10, linewidth=3, color='#3776AB') # Python logo color

# Add a fill under the line for effect
plt.fill_between(age_python_stats.index, age_python_stats.values, color='#3776AB', alpha=0.1)

# Add value labels near the points
for x, y in zip(age_python_stats.index, age_python_stats.values):
    plt.text(x, y + 1, f'{y:.1f}%', ha='center', va='bottom', fontsize=11, fontweight='bold')

plt.title('Share of Python developers among different age categories (2025)', fontsize=16, pad=20)
plt.xlabel('Age category', fontsize=12)
plt.ylabel('Python popularity (%)', fontsize=12)
plt.ylim(0, max(age_python_stats.values) + 10) # Add margin at the top for signatures
plt.xticks(rotation=45)
```

```
plt.savefig('python_popularity_by_age.png', dpi=300, bbox_inches='tight')
plt.show()
```



5.0* Analysis of industries among high-paid remote workers

```
# Convert to numeric format and calculate the 75th percentile
srp_df["ConvertedCompYearly"] = pd.to_numeric(srp_df["ConvertedCompYearly"], errors="coerce")

# Calculating the 75th percentile of compensation
perc75 = srp_df["ConvertedCompYearly"].dropna().quantile(0.75)

# We filter respondents by high compensation and remote work format
high_paid_remote = srp_df[
    (srp_df["ConvertedCompYearly"] > perc75) &
    (srp_df["RemoteWork"].str.contains("remote", case=False, na=False))
]

# We count industries (we allow for multiple values because of ;)
srp_industries = (
    high_paid_remote["Industry"]
    .dropna()
    .str.split(";")
    .explode()
    .str.strip()
    .value_counts()
)

# We output the result
print("The most common industries among high-paid remote workers:")
srp_industries_df = srp_industries.reset_index()
srp_industries_df.columns = ["Industry", "Quantity"]
srp_industries_df
```

```
The most common industries among high-paid remote workers:
```

	Industry	Quantity
0	Software Development	1503
1	Other:	267
2	Fintech	255
3	Healthcare	236
4	Internet, Telecomm or Information Services	192
5	Banking/Financial Services	156
6	Government	118
7	Media & Advertising Services	97
8	Retail and Consumer Services	95
9	Computer Systems Design and Services	92
10	Transportation, or Supply Chain	88
11	Manufacturing	81

Подальші дії: New interactive sheet

```
# Saving the result to a CSV file
srp_industries_df.to_csv('high_paid_remote_industries.csv', index=False, encoding='utf-8-sig')

print("File successfully saved as 'high_paid_remote_industries.csv'")
```

```
File successfully saved as 'high_paid_remote_industries.csv'
```

Main conclusions of the analysis

1. Data quality and completeness of respondents' answers

Result: 0 respondents answered all questions.

Conclusion: This is a completely normal phenomenon for large surveys. This confirms that the survey has a complex structure with logical branches. This also indicates "survey fatigue" due to its considerable length.

2. Work experience (WorkExp)

Statistics: The average (13.37 years) is higher than the median (10.0 years).

Conclusion: We see a positive asymmetry. Most respondents have about 10 years of experience (this is the core of the community), but the presence of a small group of "veterans" with 30–40+ years of experience pulls the average up. The market looks experienced and mature.

3. Work format and mobility

Result: 17,663 people work remotely (-36% of the total).

Conclusion: Despite the trend towards returning to offices, remote work remains a strong standard. Considering that some of the other respondents work in a hybrid format, it can be argued that flexibility is a key requirement of the modern developer.

4. Python ecosystem

Result: $37.5 of respondents use Python.

Conclusion: Python in 2025 is not just a language, but a universal tool. Every third developer uses it, which is supported by the development of AI (LLMs, AI Agents) and Data Science. High popularity guarantees a large number of libraries and community support