Generate a PDF and send it by Email

Introduction

You work for a company that sells second hand cars. Management wants to get a summary of the amounts of vehicles that have been sold at the end of every month. The company already has a web service which serves sales data at the end of every month but management wants an email to be sent out with an attached PDF so that data is more easily readable.

What you'll do

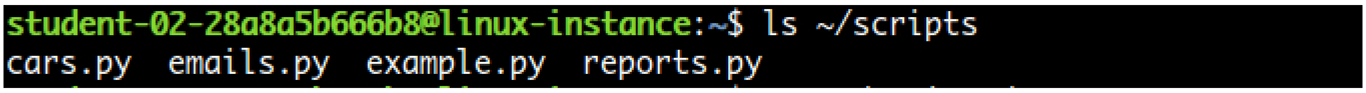
- Write a script that summarizes and processes sales data into different categories
- Generate a PDF using Python
- Automatically send a PDF by email

Sample report

In this section, you will be creating a PDF report named "**A Complete Inventory of My Fruit**". The script to generate this report and send it by email is already pre-done. You can have a look at the script in the `scripts/` directory.

```
ls ~/scripts
```

Output:



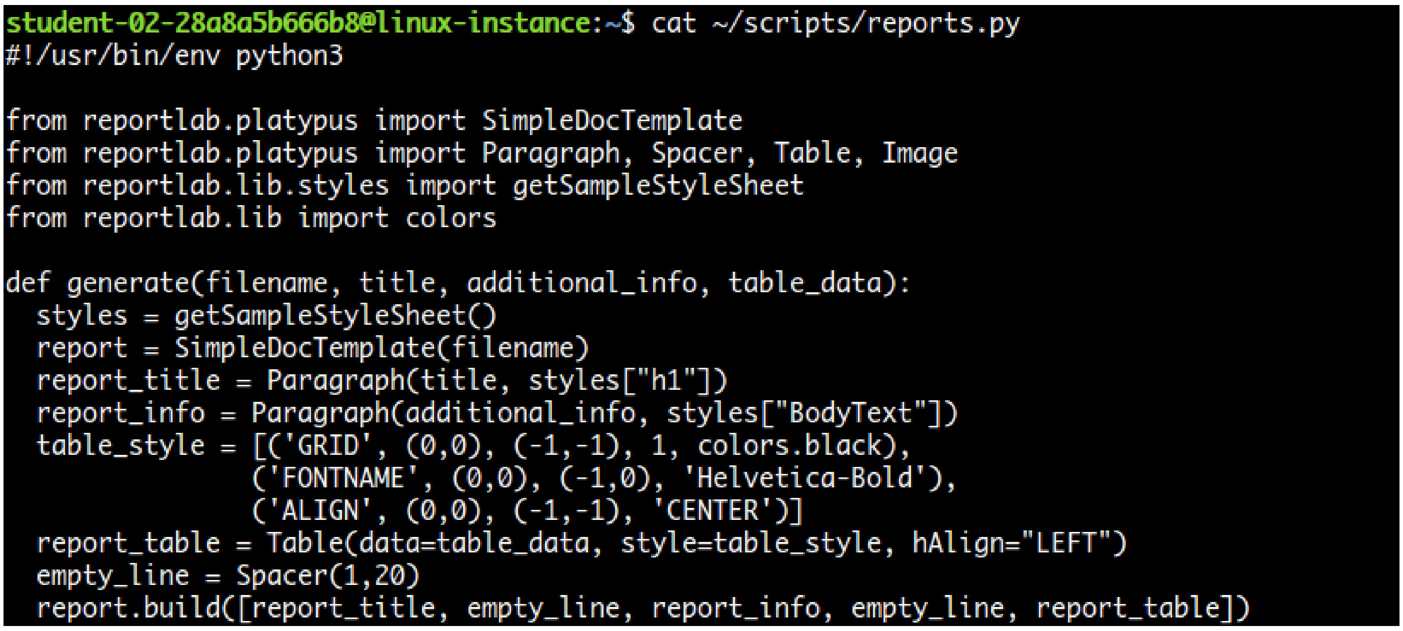
```
student-02-28a8a5b666b8@linux-instance:~$ ls ~/scripts
cars.py  emails.py  example.py  reports.py
```

In the `scripts/` directory, you will find `reports.py` and `emails.py` files. These files are used to **generate PDF files** and **send emails** respectively.

Take a look at these files using `cat` command.

```
cat ~/scripts/reports.py
```

Output:



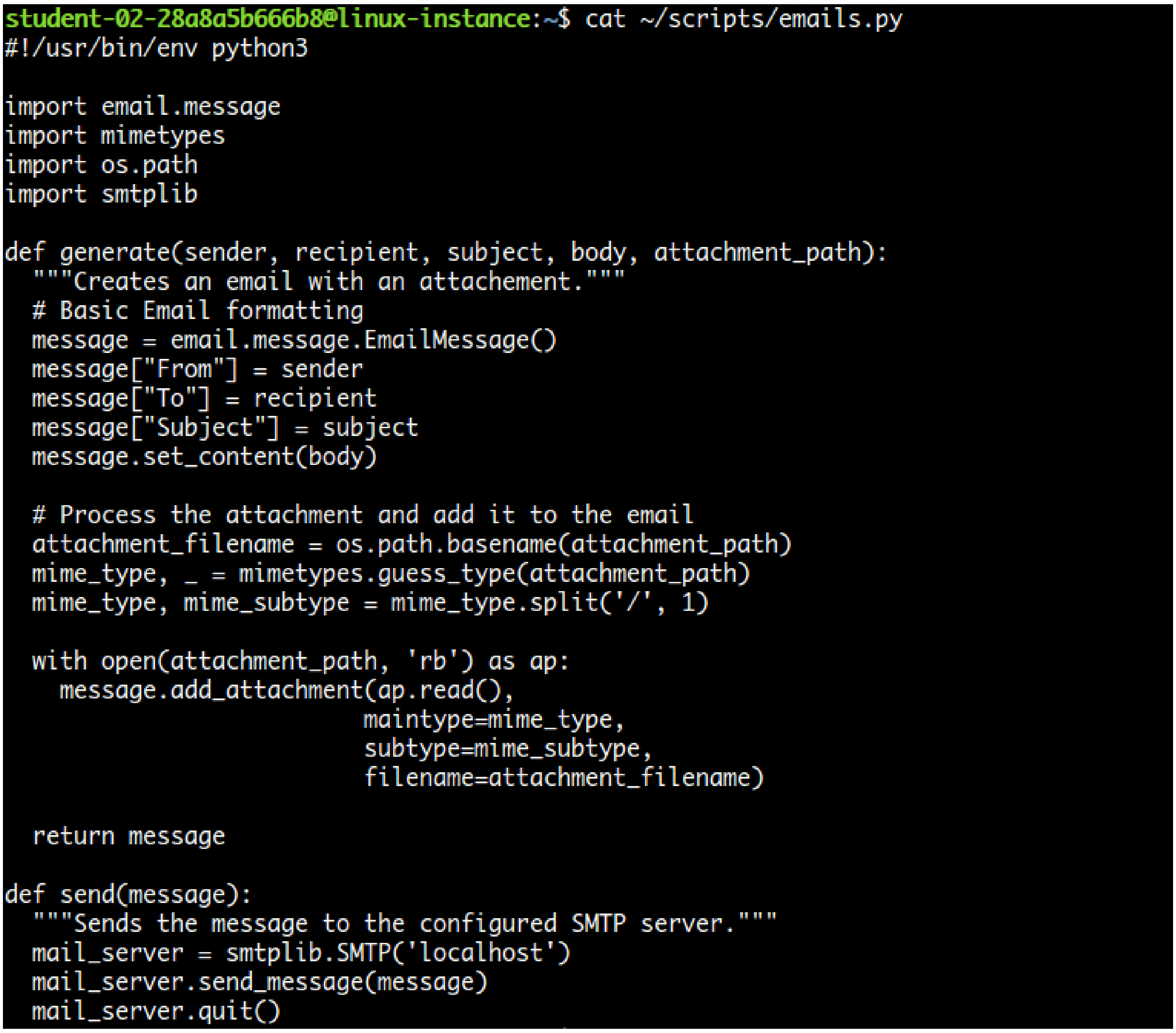
```
student-02-28a8a5b666b8@linux-instance:~$ cat ~/scripts/reports.py
#!/usr/bin/env python3

from reportlab.platypus import SimpleDocTemplate
from reportlab.platypus import Paragraph, Spacer, Table, Image
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib import colors

def generate(filename, title, additional_info, table_data):
  styles = getSampleStyleSheet()
  report = SimpleDocTemplate(filename)
  report_title = Paragraph(title, styles["h1"])
  report_info = Paragraph(additional_info, styles["BodyText"])
  table_style = [('GRID', (0,0), (-1,-1), 1, colors.black),
                 ('FONTNAME', (0,0), (-1,0), 'Helvetica-Bold'),
                 ('ALIGN', (0,0), (-1,-1), 'CENTER')]
  report_table = Table(data=table_data, style=table_style, hAlign="LEFT")
  empty_line = Spacer(1,20)
  report.build([report_title, empty_line, report_info, empty_line, report_table])
```

```
cat ~/scripts/emails.py
```

Output:



```
student-02-28a8a5b666b8@linux-instance:~$ cat ~/scripts/emails.py
#!/usr/bin/env python3

import email.message
import mimetypes
import os.path
import smtplib

def generate(sender, recipient, subject, body, attachment_path):
  """Creates an email with an attachement."""
  # Basic Email formatting
  message = email.message.EmailMessage()
  message["From"] = sender
  message["To"] = recipient
  message["Subject"] = subject
  message.set_content(body)

  # Process the attachment and add it to the email
  attachment_filename = os.path.basename(attachment_path)
  mime_type, _ = mimetypes.guess_type(attachment_path)
  mime_type, mime_subtype = mime_type.split('/', 1)

  with open(attachment_path, 'rb') as ap:
    message.add_attachment(ap.read(),
                           maintype=mime_type,
                           subtype=mime_subtype,
                           filename=attachment_filename)

  return message

def send(message):
  """Sends the message to the configured SMTP server."""
  mail_server = smtplib.SMTP('localhost')
  mail_server.send_message(message)
  mail_server.quit()
```

Now, take a look at `example.py`, which uses these two modules **reports** and **emails** to create a report and then send it by email.

```
cat ~/scripts/example.py
```

Grant executable permission to the `example.py` script.

```
sudo chmod o+wx ~/scripts/example.py
```

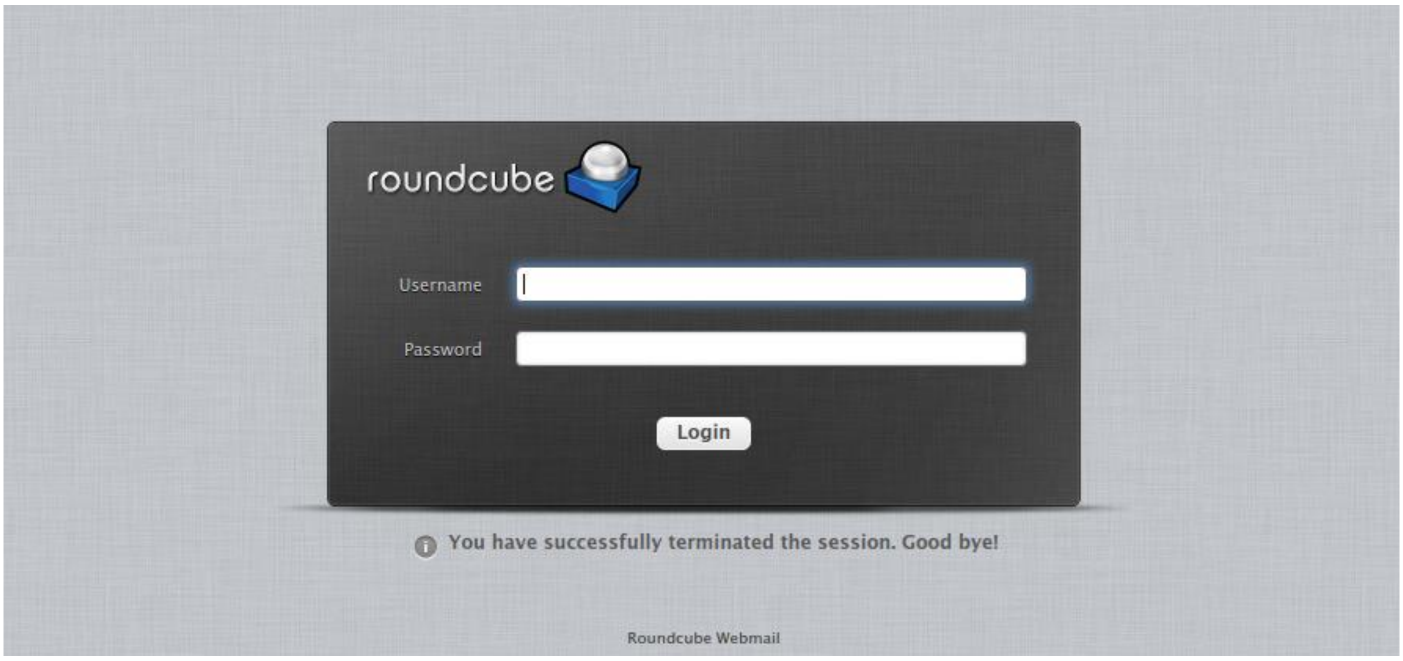
Run the `example.py` script, which will generate mail to you.

```
./scripts/example.py
```

A mail should now be successfully sent.

Copy the `external IP address` of your instance from the Connection Details Panel on the left side and open a new web browser tab and enter the IP address. The Roundcube Webmail login page appears.

Here, you'll need a login to **roundcube** using the username and password mentioned in the Connection Details Panel on the left hand side, followed by clicking **Login**.

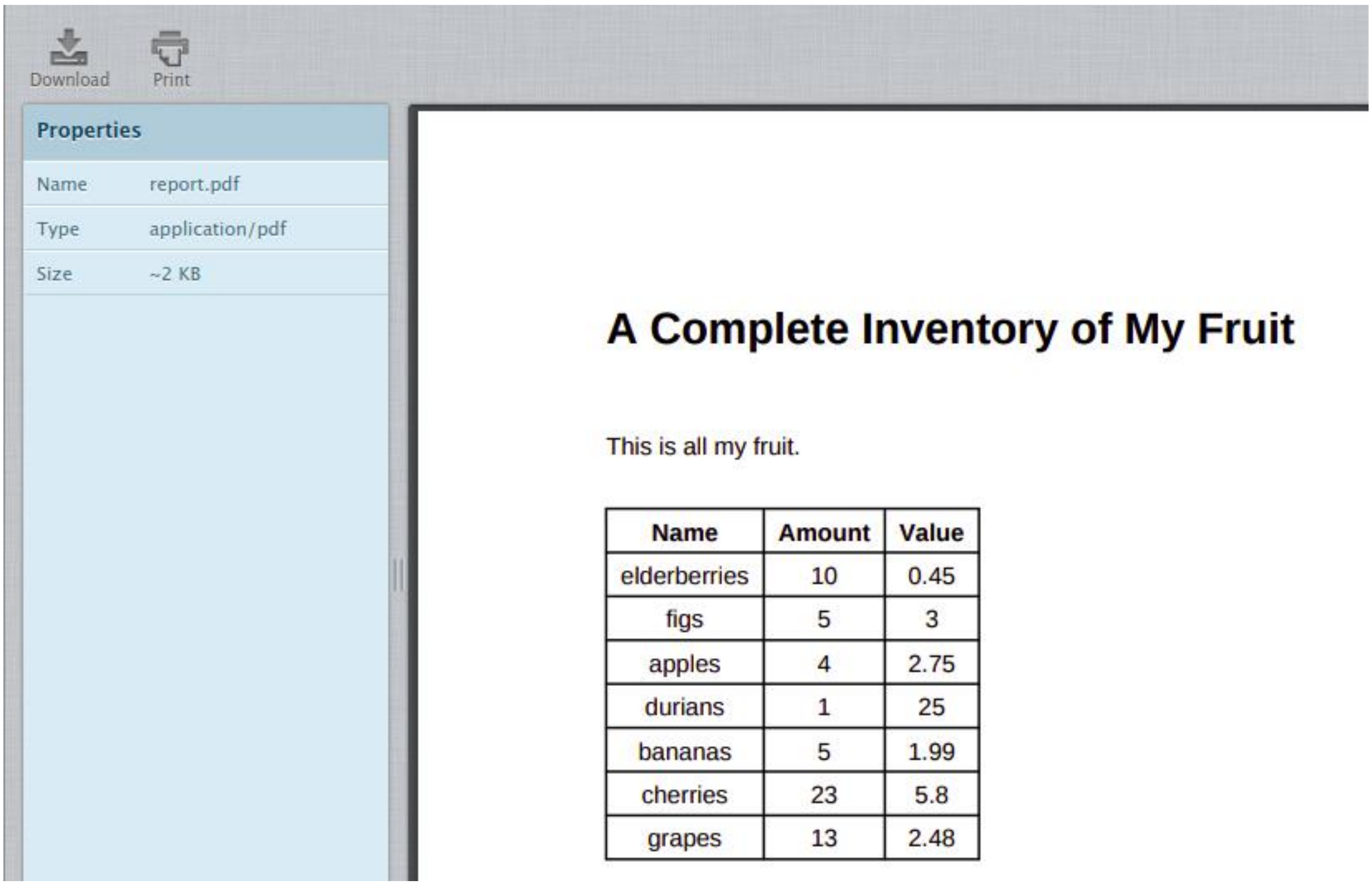


You have successfully terminated the session. Good bye!

Roundcube Webmail

Now you should be able to see your inbox, with one unread email. Open the mail by double clicking on it. There should be a report in PDF format attached to the mail. View the report by opening it.

Output:



Name report.pdf

Type application/pdf

Size ~2 KB

A Complete Inventory of My Fruit

This is all my fruit.

Name	Amount	Value
elderberries	10	0.45
figs	5	3
apples	4	2.75
durians	1	25
bananas	5	1.99
cherries	23	5.8
grapes	13	2.48

Generate report

Now, let's make a couple of changes in the `example.py` file to add a new fruit and change the sender followed by granting editor permission. Open `example.py` file using the following command:

```
nano ~/scripts/example.py
```

And update the following variables:

variable_name	value
sender	Replace **sender@example.com** with **automation@example.com**
table_data	Add another entry into the list: `['kiwi', 4, 0.49]`

The file should now look similar to:

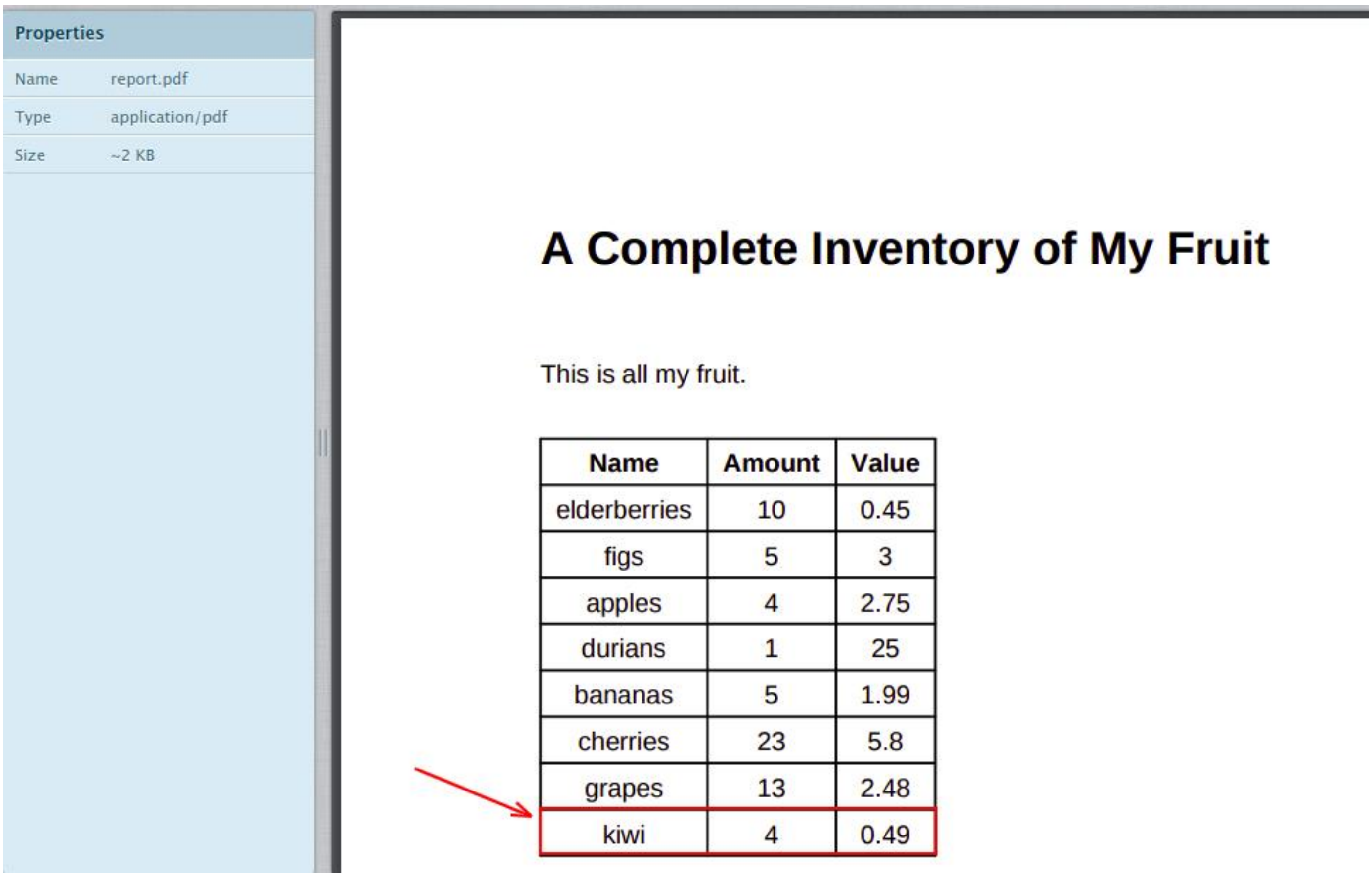
```
#!/usr/bin/env python3
import emails
import os
import reports
table_data=[
  ['Name', 'Amount', 'Value'],
  ['elderberries', 10, 0.45],
  ['figs', 5, 3],
  ['apples', 4, 2.75],
  ['durians', 1, 25],
  ['bananas', 5, 1.99],
  ['cherries', 23, 5.80],
  ['grapes', 13, 2.48],
  ['kiwi', 4, 0.49]]
reports.generate("/tmp/report.pdf", "A Complete Inventory of My Fruit", "This is all
my fruit.", table_data)
sender = "automation@example.com"
receiver = "{}@example.com".format(os.environ.get('USER'))
subject = "List of Fruits"
body = "Hi\n\nI'm sending an attachment with all my fruit."
message = emails.generate(sender, receiver, subject, body, "/tmp/report.pdf")
emails.send(message)
```

Once you've made the changes in the `example.py` script, save the file by typing **Ctrl-o**, **Enter** key and **Ctrl-x**.

Now execute the example script again.

```
./scripts/example.py
```

Now, check the webmail for any new mail. You can click on the **Refresh** button to refresh your inbox.



Name	report.pdf
Type	application/pdf
Size	~2 KB

A Complete Inventory of My Fruit

This is all my fruit.

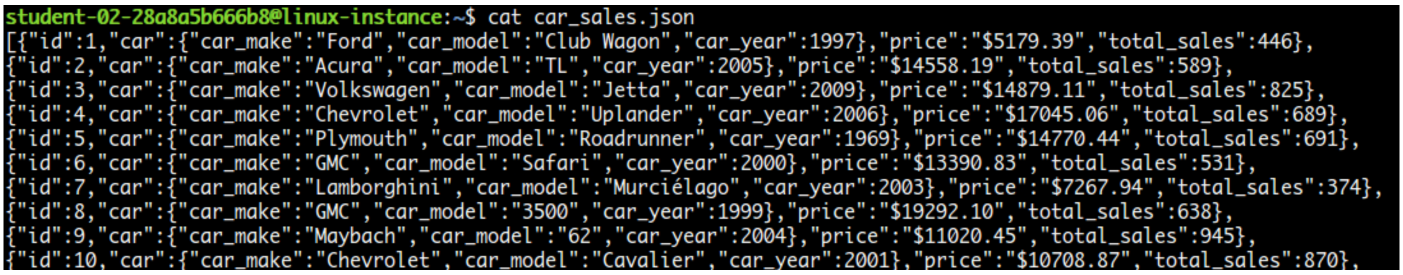
Name	Amount	Value
elderberries	10	0.45
figs	5	3
apples	4	2.75
durians	1	25
bananas	5	1.99
cherries	23	5.8
grapes	13	2.48
kiwi	4	0.49

Sales summary

In this section, let's view the summary of last month's sales for all the models offered by the company. This data is in a JSON file named `car_sales.json`. Let's have a look at it.

```
cat car_sales.json
```

Output:



```
student-02-28a8a5b666b8@linux-instance:~$ cat car_sales.json
[{"id":1,"car":{"car_make":"Ford","car_model":"Club Wagon","car_year":1997},"price":"$5179.39","total_sales":446},
{"id":2,"car":{"car_make":"Acura","car_model":"TL","car_year":2005},"price":"$14558.19","total_sales":589},
{"id":3,"car":{"car_make":"Volkswagen","car_model":"Jetta","car_year":2009},"price":"$14879.11","total_sales":825},
{"id":4,"car":{"car_make":"Chevrolet","car_model":"Uplander","car_year":2006},"price":"$17045.06","total_sales":689},
{"id":5,"car":{"car_make":"Plymouth","car_model":"Roadrunner","car_year":1969},"price":"$14770.44","total_sales":691},
{"id":6,"car":{"car_make":"GMC","car_model":"Safari","car_year":2000},"price":"$13390.83","total_sales":531},
{"id":7,"car":{"car_make":"Lamborghini","car_model":"Murciélago","car_year":2003},"price":"$7267.94","total_sales":374},
{"id":8,"car":{"car_make":"GMC","car_model":"3500","car_year":1999},"price":"$19292.10","total_sales":638},
{"id":9,"car":{"car_make":"Maybach","car_model":"62","car_year":2004},"price":"$11020.45","total_sales":945},
{"id":10,"car":{"car_make":"Chevrolet","car_model":"Cavalier","car_year":2001},"price":"$10708.87","total_sales":870},
```

To simplify the JSON structure, here is an example of one of the JSON objects among the list.

```
{
        "id": 47,
        "car": {
                "car_make": "Lamborghini",
                "car_model": "Murciélago",
                "car_year": 2002
        },
        "price": "$13724.05",
        "total_sales": 149
}
```

Here `id, car, price and total_sales` are the field names (key).

The script `cars.py` already contains part of the work, but learners need to complete the task by writing the remaining pieces. The script already calculates the car model with the most revenue (price * total_sales) in the `process_data` method. Learners need to add the following:

1. Calculate the car model which had the most sales by completing the process_data method, and then appending a formatted string to the `summary` list in the below format:

- "The {car model} had the most sales: {total sales}"

2. Calculate the most popular car_year across all car make/models (in other words, find the total count of cars with the car_year equal to 2005, equal to 2006, etc. and then figure out the most popular year) by completing the `process_data` method, and append a formatted string to the `summary` list in the below format:

- "The most popular year was {year} with {total sales in that year} sales."

The challenge

Here, you are going to update the script `cars.py`. You will be using the above JSON data to process information. A part of the script is already done for you, where it calculates the car model with the most revenue (price * total_sales). You should now fulfil the following objectives with the script:

1. Calculate the car model which had the most sales.

a. Call `format_car` method for the car model.

2. Calculate the most popular car_year across all car make/models.

Hint: Find the total count of cars with the car_year equal to 2005, equal to 2006, etc. and then figure out the most popular year.
Grant required permissions to the file `cars.py` and open it using nano editor.

```
sudo chmod o+wx ~/scripts/cars.py
```

```
nano ~/scripts/cars.py
```

The code is well commented including the TODO sections for you to understand and fulfill the objectives.

Generate PDF and send Email

Once the data is collected, you will also need to further update the script to generate a PDF report and automatically send it through email.

To generate a PDF:

- Use the `reports.generate()` function within the main function.
- The report should be named as **cars.pdf**, and placed in the folder **/tmp/**.
- The PDF should contain:
  - A summary paragraph which contains the most sales/most revenue/most popular year values worked out in the previous step.

  Note: To add line breaks in the PDF, use: <br/> between the lines.
  - A table which contains all the information parsed from the JSON file, organised by id_number. The car details should be combined into one column in the form <car_make> <car_model> (<car_year>).

  Note: You can use the **cars_dict_to_table** function for the above task.

Example:

ID	Car	Price	Total Sales

47	Acura TL (2007)	€14459,15	1192
73	Porsche 911 (2010)	€6057,74	882
85	Mercury Sable (2005)	€45660,46	874

To send the PDF through email:

Once the PDF is generated, you need to send the email, using the `emails.generate()` and `emails.send()` methods.

Use the following details to pass the parameters to `emails.generate()`:

- **From:** automation@example.com
- **To:** <user>@example.com
- **Subject line**: Sales summary for last month
- **E-mail Body:** The same summary from the PDF, but using `\n` between the lines
- **Attachment:** Attach the PDF path i.e. generated in the previous step

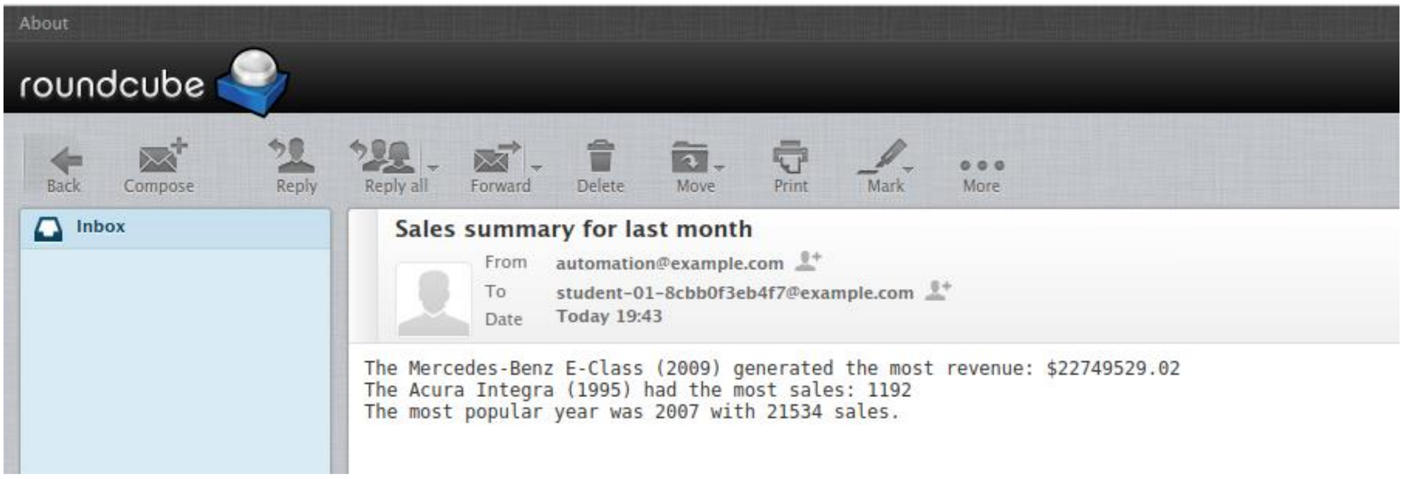
Once you have completed editing `cars.py` script, save the file by typing **Ctrl-o**, **Enter** key, and **Ctrl-x**.

Run the `cars.py` script, which will generate mail to their user.

```
./scripts/cars.py
```

Now, check the webmail for any new mail. You can click on the **Refresh** button to refresh your inbox.

Output:

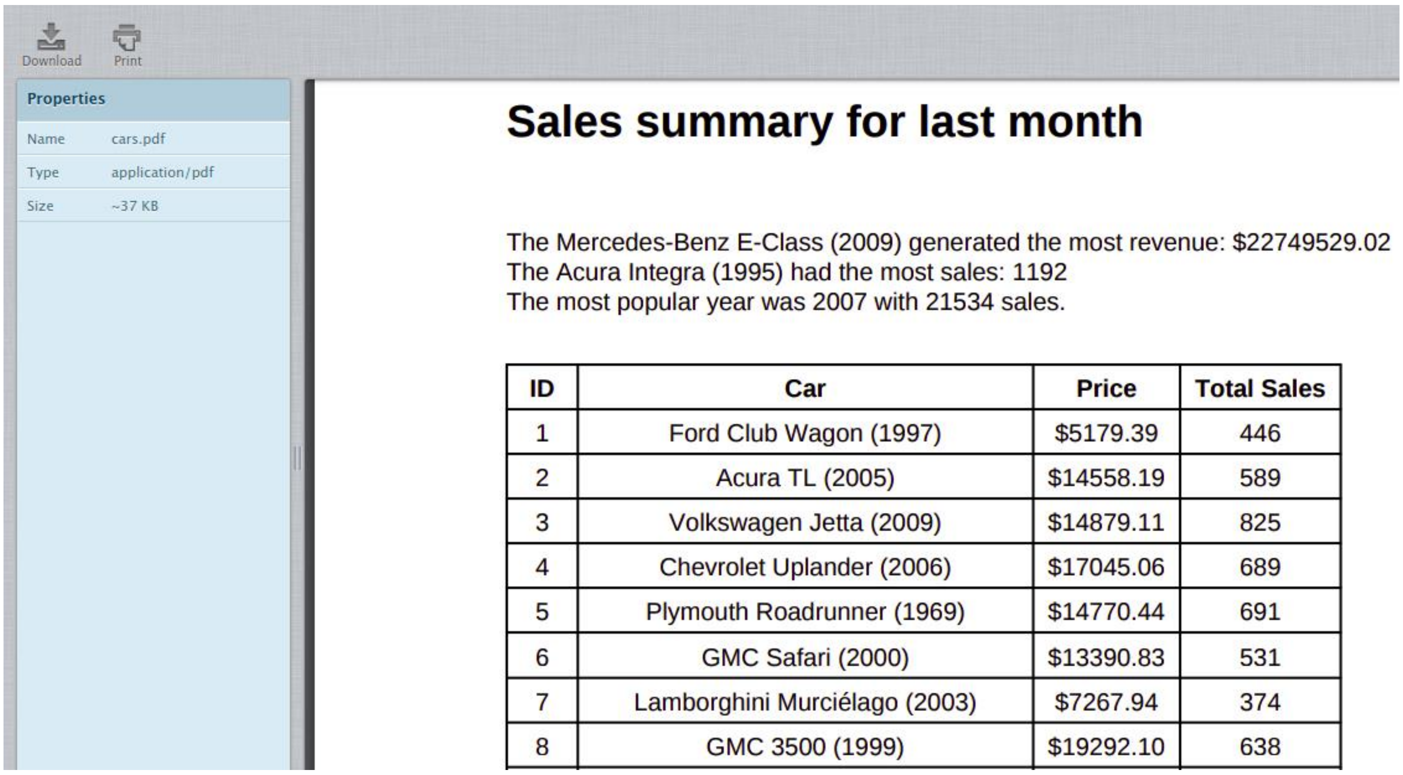


Sales summary for last month

From automation@example.com
To student-01-8cbb0f3eb4f7@example.com
Date Today 19:43

```
The Mercedes-Benz E-Class (2009) generated the most revenue: $22749529.02
The Acura Integra (1995) had the most sales: 1192
The most popular year was 2007 with 21534 sales.
```

Open `cars.pdf` that's located on the right most side.



Sales summary for last month

The Mercedes-Benz E-Class (2009) generated the most revenue: $22749529.02
The Acura Integra (1995) had the most sales: 1192
The most popular year was 2007 with 21534 sales.

ID	Car	Price	Total Sales
1	Ford Club Wagon (1997)	$5179.39	446
2	Acura TL (2005)	$14558.19	589
3	Volkswagen Jetta (2009)	$14879.11	825
4	Chevrolet Uplander (2006)	$17045.06	689
5	Plymouth Roadrunner (1969)	$14770.44	691
6	GMC Safari (2000)	$13390.83	531
7	Lamborghini Murciélago (2003)	$7267.94	374
8	GMC 3500 (1999)	$19292.10	638

Optional challenge

As **optional** challenges, you could try some of the following functionalities:

1. Sort the list of cars in the PDF by total sales.
2. Create a pie chart for the total sales of each car made.
3. Create a bar chart showing total sales for the top 10 best selling vehicles using the ReportLab Diagra library. Put the vehicle name on the X-axis and **total revenue** (remember, price * total sales!) along the Y-axis.