```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "# Assignment0 : Function as a Service (FaaS)\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Description\n",
    "The objective of this assignment is to gain proficiency in building, deploying, and running code in a cloud environment using an event-driven serverless architecture. You will be setting up an Azure environment and creating, deploying, and testing an Azure Function on a cloud platform.\n",
    "\n",
    "To complete this assignment, you will begin by running and testing a Python program on your local machine and verifying the output. Next, you will proceed to deploy and test the same function on Azure utilizing Azure Functions, and compare the results with those obtained from local execution. Please adhere to the provided instructions to finish the assignment."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## Run and test a python program on a local machine\n",
    "For simplicity, a python script has been provided for you in this assignment. The Python script example reads a paragraph of text, counts the number of occurrences of each word, and dumps the result in a JSON format.
"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "import json\n",
    "from collections import Counter\n",
```

```
    "\n",
    "\n",
    "def word_count(paragraph):\n",
    "    \n",
    "    # Remove punctuation from input paragraph and convert to
lowercase\n",
    "    paragraph = paragraph.lower().replace('.', '').replace(',',
'')\n",
    "    \n",
    "    # Split the paragraph into a list of words\n",
    "    words = paragraph.split()\n",
    "\n",
    "    # Count the number of occurrences of each word\n",
    "   # word_counts = dict(Counter(words))\n",
    "    word_count = Counter(words)\n",
    "\n",
    "    # Sort the words in descending order of frequency\n",
    "    sorted_word_count = dict(sorted(word_count.items(), key=lambda
item: item[1], reverse=True))\n",
    "    print (sorted_word_count)\n",
    "    return sorted_word_count\n",
    "    "
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "You can call this function from your local machine with a paragraph
string as follows:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "\n",
    "# Define the input paragraph\n",
    "paragraph = \"\"\"\n",
    "ChatGPT is a cutting-edge language model that can perform a wide range
of natural language processing tasks with remarkable accuracy and fluency.
From answering general knowledge questions and assisting with research, to
generating human-like text and even engaging in creative writing, ChatGPT's
capabilities are truly impressive. With its vast knowledge base and ability
```

```
to learn and adapt over time, ChatGPT can serve as a valuable resource for anyone looking to harness the power of natural language processing in their work or personal projects. Whether you're a researcher, a writer, a business professional, or simply someone looking to engage in interesting conversations, ChatGPT has the skills and knowledge to help you achieve your goals.\"\"\"\n",
    "\n",
    "\n",
    "result = word_count(paragraph)\n",
    "\n",
    "print(\"the most frequent word is \", max(result, key=result.get), \" and its frequency is \", result[max(result, key=result.get)])\n",
    " "
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    " - What are three most frequent words? \n",
    " - What are their frequency?\n",
    "\n",
    "\n",
    " "
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## How to go serverless?\n",
    "Serverless computing is a cloud computing model that allows developers to build and run applications without having to manage servers or infrastructure. In a serverless architecture, the cloud provider is responsible for managing the underlying infrastructure, including servers, storage, and networking. Developers only need to write code for their applications and upload it to the serverless platform, which automatically handles the deployment, scaling, and management of the code. To deploy the above function as a serverless function, you need to walk through the following steps:\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
```

```
    "### 1- Choose a Cloud platform: \n",
    "You have the option to select from the leading cloud providers: Amazon Web Services, Microsoft Azure, or Google Cloud Platform. Depending on your choice, you will be working with a serverless computing service like AWS Lambda, Azure Functions, or Google Cloud Functions to fulfill the requirements of this assignment.\n",
    " \n",
    "### 2- Sign up for an account: \n",
    " - Set up an account in one cloud platform (e.g., Azure). \n",
    " - Log in to your account (e.g., Azure portal) and navigate to FaaS service (e.g., Azure Functions)\n",
    "\n",
    "### 3- Create a new Function App Resource: \n",
    " Click on the \"Create Function App\" button. Choose an existing resource group or create a new one. Give it a name, and choose \"Code\" as development type. You can create a function in your preferred programming language, such as Python, Node.js or Java. Alternatively, you can use the above python script that has been provided for you. Select Python 3.8 as the runtime, and choose an appropriate storage account (This can later be used as file upload trigger as well). Additionally feel free to explore the Github actions features for CI/CD."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### 4- Justify the function code\n",
    " You need to apply slight modification to the function through event definition. \n",
    " - When you deploy a function to a serverless platform like Azure Functions, the platform handles the invocation of the function by passing an HTTP request and a context object as arguments to the function. The HTTP request provides information about the trigger that caused the function to be invoked, such as an HTTP request, a message from a queue or an input data. The context object provides information about the runtime environment, such as the Azure region, function name, and time remaining before the function times out. In the context of our example function, the HTTP request represents the input paragraph string that the function processes. The context object isn't used in this particular function, but it's included in the function signature because it's a required argument for Azure Functions.\n",
    " \n",
    " - In serverless computing, a serverless function typically sends its response back to the client or service that invoked it by returning a JSON object that includes a statusCode and a body. The statusCode indicates the
```

```
status of the response, such as whether the request was successful or not, and the body contains the response data in JSON format. In the example function, the sorted_word_count dictionary contains the word frequencies that the function has computed. To return this data to the client that invoked the function, we use the json.dumps() method to convert the dictionary to a JSON-formatted string and include it as the value of the body key in the response object. We set the statusCode to 200 to indicate that the request was successful."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Now we can start creating functions with different triggers and bindings. Click on \"Create function\". For now, you will be using HTTP trigger and HTTP output binding. For the authorization level, choose \"Anonymous\". You can later change it to \"Function\" or \"Admin\" level."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "import azure.functions as func\n",
    "import logging\n",
    "from collections import Counter\n",
    "import json\n",
    "\n",
    "app = func.FunctionApp(http_auth_level=func.AuthLevel.ANONYMOUS)\n",
    "\n",
    "@app.route(route=\"assignment0\")\n",
    "def assignment0(req: func.HttpRequest) -> func.HttpResponse:\n",
    "    logging.info('Python HTTP trigger function processed a request.')\n",
    "    try:\n",
    "        req_body = req.get_json()\n",
    "    except ValueError:\n",
    "        return func.HttpResponse(\n",
    "             \"Invalid input. Please provide a valid JSON with a 'paragraph' field.\",\n",
    "             status_code=400\n",
    "        )\n",
    "\n",
```

```
    "    paragraph = req_body.get('paragraph')\n",
    "    if not paragraph:\n",
    "        return func.HttpResponse(\n",
    "             \"Please provide a paragraph in the request body for
processing.\",\n",
    "             status_code=400\n",
    "        )\n",
    "\n",
    "    word_count = process_paragraph(paragraph)\n",
    "\n",
    "    return func.HttpResponse(\n",
    "        json.dumps(word_count),\n",
    "        status_code=200,\n",
    "        mimetype=\"application/json\"\n",
    "    )\n",
    "\n",
    "def process_paragraph(paragraph: str) -> dict:\n",
    "    # Remove punctuation from input paragraph and convert to
lowercase\n",
    "    paragraph = paragraph.lower().replace('.', '').replace(',',
'')\n",
    "\n",
    "    # Split the paragraph into a list of words\n",
    "    words = paragraph.split()\n",
    "\n",
    "    # Count the number of occurrences of each word\n",
    "    word_count = Counter(words)\n",
    "\n",
    "    # Sort the words in descending order of frequency\n",
    "    sorted_word_count = dict(sorted(word_count.items(), key=lambda
item: item[1], reverse=True))\n",
    "    \n",
    "    return sorted_word_count"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "\n",
    "In the function code editor, copy and paste the word_count function
definition we defined earlier. \n",
    "This version of the function sorts the frequency of words in
descending order using the sorted() method with a key function that sorts
by the second element of each tuple in the dictionary. The resulting
dictionary is then converted to a JSON object using the json.dumps() method
```

```
and returned in the response body."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### 5- Define the input event\n",
    "You can invoke your Azure Function directly by creating a JSON event object and passing it as an argument to the function. Here's an example of how to define an event for the word_count function that we defined earlier:"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "{\n",
    "    \"paragraph\": \"ChatGPT is a cutting-edge language model that can perform a wide range of natural language processing tasks with remarkable accuracy and fluency. From answering general knowledge questions and assisting with research, to generating human-like text and even engaging in creative writing, ChatGPT's capabilities are truly impressive. With its vast knowledge base and ability to learn and adapt over time, ChatGPT can serve as a valuable resource for anyone looking to harness the power of natural language processing in their work or personal projects. Whether you're a researcher, a writer, a business professional, or simply someone looking to engage in interesting conversations, ChatGPT has the skills and knowledge to help you achieve your goals.\"\n",
    "}"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "In this example, we define a JSON object with a \"paragraph\" key that contains the input paragraph to be analyzed. We then call the 'word_count' function and pass this object as the argument."
   ]
  },
  {
   "cell_type": "markdown",
```

```
   "metadata": {},
   "source": [
    "### 6- Deploy and test the function\n",
    " You can test the function by invoking it with test data as event. Click on the \"Test\" button in the Azure web console to test the function.\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### 7- Serverless Word Count with Azure Functions and Blob Storage Triggers\n",
    "\n",
    "In this step, you need to create and deploy a serverless function for a Word Count application, triggered by Blob Storage file upload event. Once triggered, it should parse the uploaded file to get the word count.\n",
    "\n",
    "1. **Navigate to Azure Storage Account**\n",
    "    - Search and navigate to the Azure Storage Account.\n",
    "2. **Create a new Storage Account**\n",
    "    - Use the previous storage account that you created while initializing function App or click on \"Create Storage Account\" to create a new one.\n",
    "    - Navigate to the \"Storage Accounts\" tab.\n",
    "    - Click on \"Container\" and then click on \"+ Container\" to create a new container.\n",
    "    - Select the access type as \"Private (no anonymous access)\".\n",
    "    - Click on \"OK\".\n",
    "\n",
    "3. **Edit Storage Account Permissions**\n",
    "    - Now you can see your container in the list for that storage account.\n",
    "    - Navigate to the container and click on \"Shared access tokens\".\n",
    "    - Then click the \"Generate SAS token and URL\" button.\n",
    "    - Copy your containers SAS token and URL. This will be used to access the container from the Azure Function or external python scripts.\n",
    "\n",
    "```\n",

"https://<STORAGE_ACCOUNT>.blob.core.windows.net/<CONTAINER_NAME>?<BLOB_SAS_TOKEN>\n",
    "```\n",
```

```
    "\n",
    "4. **Add Trigger to Azure Function**\n",
    "    - Go to the Azure Functions dashboard, and click on \"New
Function\".\n",
    "    - In job type, you can either append a new route for your existing
function, or create a route in a new file altogether.\n",
    "    - Select Blob Trigger from the list of triggers.\n",
    "    - Choose the storage account name from the dropdown list.\n",
    "    - Provide the container name as path.\n",
    "    - Select the event type as \"Blob Created\" and click on
\"Add\".\n",
    "\n",
    "5. **File to upload to Blob Storage**\n",
    "   - Create a txt file called `upload.txt` in the same directory as
this notebook and add some text to it (copy 'paragraph' from above)\n",
    "\n",
    "6. **Upload File to Blob Storage**\n",
    "   - Now running the following code will upload a file to the Blob
Storage (you may need to install azure-storage-blob package and restart the
notebook kernel)."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "from azure.storage.blob import BlobClient\n",
    "\n",
    "def upload_file_to_blob(file_name, account_endpoint, container_name,
blob_name, sas_token):\n",
    "    \"\"\"\n",
    "    Upload a file to Azure Blob Storage using the account endpoint,
container name, blob name, and SAS token.\n",
    "\n",
    "    :param file_name: Local file to upload\n",
    "    :param account_endpoint: Endpoint URL of the Azure Storage
account\n",
    "    :param container_name: Name of the container\n",
    "    :param blob_name: Name of the blob to create or overwrite\n",
    "    :param sas_token: SAS token for authentication\n",
    "    :return: True if file was uploaded, else False\n",
    "    \"\"\"\n",
    "    try:\n",
    "        # Construct the full Blob SAS URL\n",
```

```
    "        blob_sas_url = f\"{account_endpoint}/{container_name}/{blob_name}?{sas_token}\"\n",
    "\n",
    "        # Create a BlobClient using the Blob SAS URL\n",
    "        blob_client = BlobClient.from_blob_url(blob_sas_url)\n",
    "\n",
    "        # Upload the file\n",
    "        with open(file_name, \"rb\") as data:\n",
    "            blob_client.upload_blob(data, overwrite=True)\n",
    "        return True\n",
    "    except Exception as e:\n",
    "        print(f\"Error occurred: {e}\")\n",
    "        return False\n",
    "\n",
    "# Example usage\n",
    "file_name = 'upload.txt'\n",
    "account_endpoint = \"https://<STORAGE_ACCOUNT_NAME>.blob.core.windows.net\"\n",
    "container_name = \"<CONTAINER_NAME>\"\n",
    "blob_name = 'upload.txt'  # Name of the blob to create or overwrite\n",
    "sas_token = \"<SAS_TOKEN>\"\n",
    "\n",
    "# Upload the file\n",
    "upload_successful = upload_file_to_blob(file_name, account_endpoint, container_name, blob_name, sas_token)\n",
    "if upload_successful:\n",
    "    print(\"File uploaded successfully.\")\n",
    "else:\n",
    "    print(\"Failed to upload the file.\")\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "Update the serverless function to parse the uploaded .txt file when called by the S3 bucket event :"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
```

```
    "import azure.functions as func\n",
    "import logging\n",
    "from collections import Counter\n",
    "import json\n",
    "\n",
    "app = func.FunctionApp()\n",
    "\n",
    "@app.blob_trigger(arg_name=\"myblob\", path=\"<CONTAINER_NAME>\",\n",
    "
connection=\"<CONNECTION_NAME_TO_STORAGE_ACCOUNT>\") \n",
    "def BlobTrigger(myblob: func.InputStream):\n",
    "    # Read the blob content\n",
    "    blob_content = myblob.read().decode('utf-8')\n",
    "\n",
    "    # Process the content (for example, counting words)\n",
    "    word_count = process_paragraph(blob_content)\n",
    "    \n",
    "    print(word_count)\n",
    "    # Log or further process the word count\n",
    "    logging.info(json.dumps(word_count))\n",
    "    return None\n",
    "\n",
    "def process_paragraph(paragraph: str) -> dict:\n",
    "    # Remove punctuation from input paragraph and convert to
lowercase\n",
    "    paragraph = paragraph.lower().replace('.', '').replace(',',
'')\n",
    "\n",
    "    # Split the paragraph into a list of words\n",
    "    words = paragraph.split()\n",
    "\n",
    "    # Count the number of occurrences of each word\n",
    "    word_count = Counter(words)\n",
    "\n",
    "    # Sort the words in descending order of frequency\n",
    "    sorted_word_count = dict(sorted(word_count.items(), key=lambda
item: item[1], reverse=True))\n",
    "    \n",
    "    return sorted_word_count"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "After you run the file upload function, you can see the uploaded file
```

```
in the Blob Storage. This will also trigger the Azure Function and you can see the output in the Azure portal by navigating to the function and clicking on “Monitor”->“Logs”."
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "## 8 Create an API Managment Service\n",
    "In this step, you will create an API Managment Service so you can send various paragraphs to be processed by the function apps. \n",
    "\n",
    "An API Managment Service serves as a centralized entry point for managing, securing, and optimizing APIs. Its main purposes include routing requests to backend services, enforcing security measures, providing monitoring and analytics, transforming data formats, implementing caching, load balancing, handling cross-cutting concerns, and managing API versions. It simplifies API management, enhances security, and improves overall system performance.\n",
    "\n",
    "0. **Create a New Serverless Function**\n",
    "    - Navigate to function app console\n",
    "    - Create a new Function App called \"WordCountRestFunction\" with similiar settings as used previously\n",
    "    - Create a Function inside the function app called \"count\" \n",
    "    - Copy the following code into the serverless function\n",
    "    <br/><br/>"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "import azure.functions as func\n",
    "import logging\n",
    "from collections import Counter\n",
    "import json\n",
    "\n",
    "app = func.FunctionApp(http_auth_level=func.AuthLevel.ANONYMOUS)\n",
    "\n",
    "@app.route(route=\"assignment0\")\n",
    "def assignment0(req: func.HttpRequest) -> func.HttpResponse:\n",
    "    logging.info('Python HTTP trigger function processed a
```

```
request.')\n",
    "    try:\n",
    "        req_body = req.get_json()\n",
    "    except ValueError:\n",
    "        return func.HttpResponse(\n",
    "             \"Invalid input. Please provide a valid JSON with a
'paragraph' field.\",\n",
    "             status_code=400\n",
    "        )\n",
    "\n",
    "    paragraph = req_body.get('paragraph')\n",
    "    if not paragraph:\n",
    "        return func.HttpResponse(\n",
    "             \"Please provide a paragraph in the request body for
processing.\",\n",
    "             status_code=400\n",
    "        )\n",
    "\n",
    "    word_count = process_paragraph(paragraph)\n",
    "\n",
    "    return func.HttpResponse(\n",
    "        json.dumps(word_count),\n",
    "        status_code=200,\n",
    "        mimetype=\"application/json\"\n",
    "    )\n",
    "\n",
    "def process_paragraph(paragraph: str) -> dict:\n",
    "    # Remove punctuation from input paragraph and convert to lowercase
\n",
    "    paragraph = paragraph.lower().replace('.', '').replace(',',
'')\n",
    "\n",
    "    # Split the paragraph into a list of words\n",
    "    words = paragraph.split()\n",
    "\n",
    "    # Count the number of occurrences of each word\n",
    "    word_count = Counter(words)\n",
    "\n",
    "    # Sort the words in descending order of frequency\n",
    "    sorted_word_count = dict(sorted(word_count.items(), key=lambda
item: item[1], reverse=True))\n",
    "    \n",
    "    return sorted_word_count"
   ]
  },
  {
```

```
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "\n",
    "1. **Navigate to API Managment Services console**\n",
    "    - Navigate to API Managment Services.\n",
    "    <br/><br/>\n",
    "\n",
    "2. **Create a new API Managment Service**\n",
    "    - Click on \"+ Create\"\n",
    "    - Choose the same resource group and region as the function 
app\n",
    "    - Name your resource name, organization name, and administrator 
email\n",
    "    - Click \"Review + create\"\n",
    "    - The deployment process may take more than 10 minutes\n",
    "    <br/><br/>\n",
    "\n",
    "3. **While in your New API Managment Service Click on \"APIs\" in the 
side menu**<br/><br/>\n",
    "\n",
    "4. **Connect Function App To API Managment Service**\n",
    "    - Once you have clicked on APIs (in the side bar) numerous options 
will popup. If you have already created an API click \"Add API\".\n",
    "    - Scroll down until you see the title \"Create from Azure 
Resource\"\n",
    "    - Below this click on \"Function App\"\n",
    "    - A pop up will appear that says \"Create from Function App\"\n",
    "    - Click Browse to select your function app - this will navigate 
you to another page\n",
    "    - On this next page press the \"Select\" button (far right side) 
to choose your function app \"WordCountRestFunction\" that you made in the 
previous steps. Then press \"Select\" in the bottom left hand side of the 
page to confirm your choice\n",
    "    - Name your display name \"WordCountRestAPI\" \n",
    "    - <b>Name your API URL suffix: test</b>\n",
    "    - Click \"create\"\n",
    "    - Once the API is created you will see your API 
\"WordCountRestAPI\" in your side panel\n",
    "    <br/><br/>\n",
    "\n",
    "5. **Turn Off Security Settings**\n",
    "    - Click on your \"WordCountRestAPI\" in your side panel\n",
    "    - Click on \"Settings\" in the top tab for the 
\"WordCountRestAPI\" api\n",
    "    - <b>Scroll down and make sure \"Subscription Required\" is
```

```
unchecked </b>(allows access to your endpoint without a security token)\n",
    "    - At the bottom of the page press \"Save\"\n",
    "    <br/><br/>\n",
    "\n",
    "7. **Test REST API endpoint url**\n",
    "    - Click on your \"WordCountRestAPI\" in your side panel   \n",
    "    - Click on the \"Test\" Tab\n",
    "    - Click on \"POST xxxxxx\" in the side panel\n",
    "    - In \"Request Body\" paste the following JSON below and press
\"Send\". You should see a count of all your words in your HTTP response
message like the previous part of the assignment"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "{\n",
    "    \"paragraph\": \"ChatGPT is a cutting-edge language model that can
perform a wide range of natural language processing tasks with remarkable
accuracy and fluency. From answering general knowledge questions and
assisting with research, to generating human-like text and even engaging in
creative writing, ChatGPT's capabilities are truly impressive. With its
vast knowledge base and ability to learn and adapt over time, ChatGPT can
serve as a valuable resource for anyone looking to harness the power of
natural language processing in their work or personal projects. Whether
you're a researcher, a writer, a business professional, or simply someone
looking to engage in interesting conversations, ChatGPT has the skills and
knowledge to help you achieve your goals.\"\n",
    "}"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "8. **Get REST API endpoint url**\n",
    "    - Scroll down to the bottom of the \"Test Tab\"\n",
    "    - Copy the \"Request URL\" (should similiar to this
https://xyz.azure-api.net/WordCountRestAPI/test/count)<br/><br/>\n",
    "\n",
    "9. **Run the following python script to hit the \"Request URL\"
endpoint**\n",
    "    - Utilize the python script below to hit the api endpoint\n",
```

```
    "    - Ensure to replace \"REQUEST_URL\" variable with your request url
from the previous step<br/><br/>\n",
    "\n",
    "10. **If you run into \"Access denied due to missing subscription
key\"**\n",
    "    - Go back to Step 5 and turn off the security settings"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "import requests\n",
    "import json\n",
    "\n",
    "# REPLACE with your REQUEST URL\n",
    "REQUEST_URL =
\"https://xyz.azure-api.net/WordCountRestAPI/test/count\"\n",
    "\n",
    "# JSON body to be sent in the POST request\n",
    "data = {\n",
    "    \"paragraph\": \"We were both young when I first saw you I close
my eyes and the flashback starts I'm standin' there On a balcony in summer
air See the lights, see the party, the ball gowns See you make your way
through the crowd And say, Hello Little did I know That you were Romeo, you
were throwin' pebbles And my daddy said, Stay away from Juliet And I was
cryin' on the staircase Beggin' you, Please don't go,  and I said Romeo,
take me somewhere we can be alone I'll be waiting, all there's left to do
is run You'll be the prince and I'll be the princess It's a love story,
baby, just say, Yes So I sneak out to the garden to see you We keep quiet,
'cause we're dead if they knew So close your eyes Escape this town for a
little while, oh oh 'Cause you were Romeo, I was a scarlet letter And my
daddy said, Stay away from Juliet But you were everything to me I was
beggin' you, Please don't go,  and I said Romeo, take me somewhere we can
be alone I'll be waiting, all there's left to do is run You'll be the
prince and I'll be the princess It's a love story, baby, just say, Yes
Romeo, save me, they're tryna tell me how to feel This love is difficult,
but it's real Don't be afraid, we'll make it out of this mess It's a love
story, baby, just say, Yes Oh, oh I got tired of waiting Wonderin' if you
were ever comin' around My faith in you was fading When I met you on the
outskirts of town, and I said Romeo, save me, I've been feeling so alone I
keep waiting for you, but you never come Is this in my head? I don't know
what to think He knelt to the ground and pulled out a ring And said, Marry
me, Juliet You'll never have to be alone I love you and that's all I really
```

```
know I talked to your dad, go pick out a white dress It's a love story, baby, just say, Yes Oh, oh, oh Oh, oh, oh, oh 'Cause we were both young when I first saw you\"\n",
    "}\n",
    "\n",
    "# Convert the Python dictionary to a JSON string\n",
    "json_data = json.dumps(data)\n",
    "\n",
    "# Specify the headers if needed (e.g., content type as JSON)\n",
    "headers = {\n",
    "    \"Content-Type\": \"application/json\"\n",
    "}\n",
    "\n",
    "# Send the POST request\n",
    "response = requests.post(REQUEST_URL, data=json_data, headers=headers)\n",
    "\n",
    "# Print the response\n",
    "print(\"Response Code:\", response.status_code)\n",
    "print(\"Response Body:\", response.text)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### 9-Create and Deploy your own code\n",
    "\n",
    "You learnt how to create a Function as a Service (FaaS) through steps 1-7. Now, you need to choose/develop a simple code and attempt to deploy it as an Azure Function.\n",
    "\n",
    "To deploy your code as an Azure Function, you need to modify it to conform to the platform's requirements. \n",
    "\n",
    "To trigger the execution of your function, you must specify an event. The event could be anything that the Azure Functions platform supports, such as a new file upload to Blob Storage, a new database entry, or an HTTP request. Once the event is detected, the platform will invoke your function, passing it the necessary input parameters. The function will then execute and return the output to the platform, which can be further processed or returned to the user.\n",
    "\n",
    "- What modification is required in your code?\n",
    "- What could be defined as an event to trigger your function execution?\n",
```

```
    "- Did you face any dependency problem?"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Deliverables:\n",
    "\n",
    "In a word document please show:\n",
    "- Step 6 - 7:\n",
    "    - Pictures of \"Execution Result\" of word_count serverless function. \n",
    "    - The brief explanation of how you configured the event to trigger your function app. \n",
    "- Step 8:\n",
    "    - A Picture of your \"Test Tab\" showing your Request URL in your API Managment Service\n",
    "    - A Picture of your severless function\n",
    "    - A Picture of the \"Execution result\" of running your serverless function\n",
    "    - Your request url that you used to process words with your API Managment Service for TA testing\n",
    "- Step 9:\n",
    "    - The serverless function code you created for step 9.\n",
    "    - The brief explanation of how you configured the event to trigger your function app\n",
    "    - The event code\n",
    "    - The \"Execution result\" of running your serverless function"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "### Good Luck!"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
```

```
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.11.1"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```