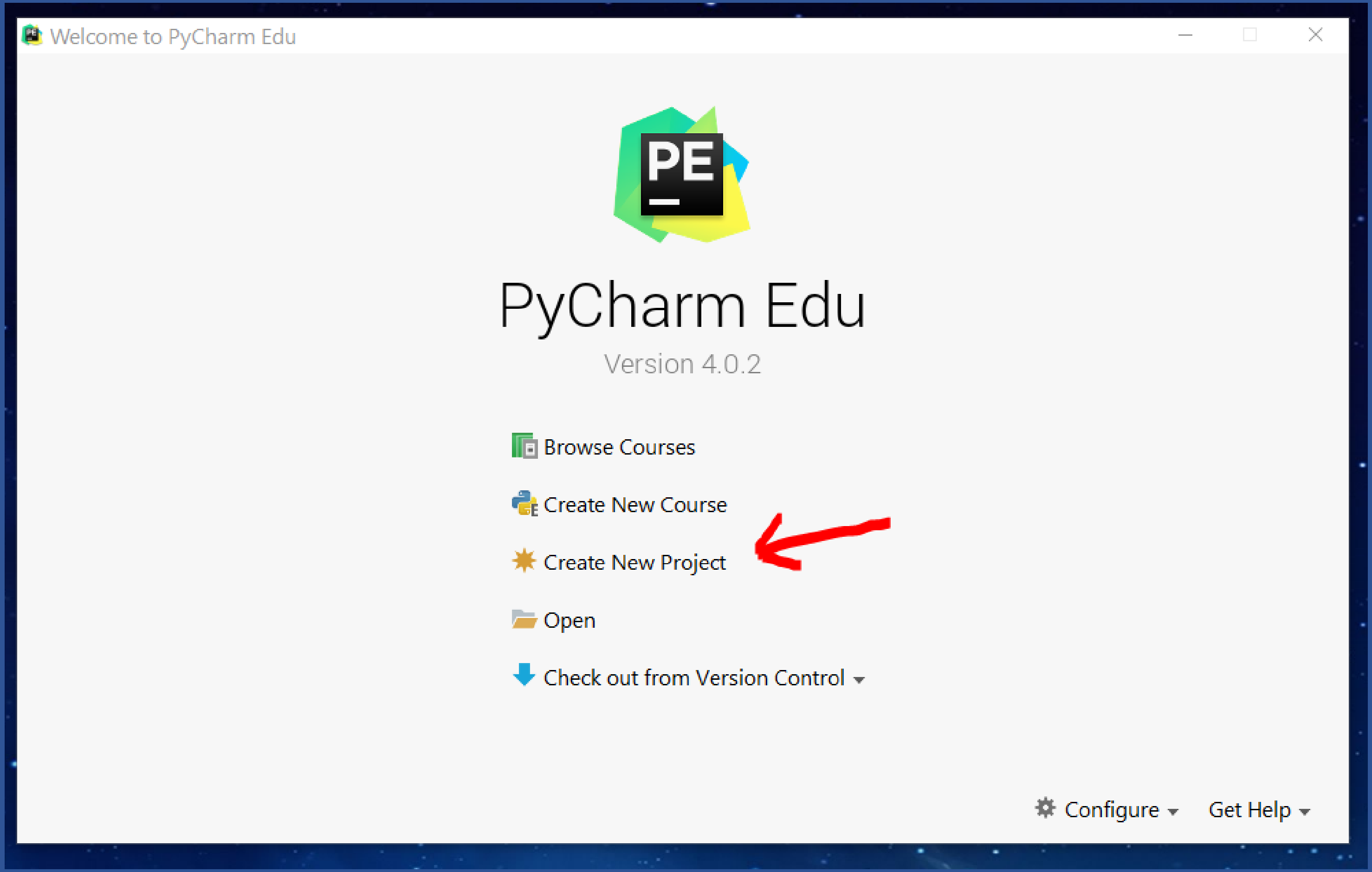
Consume Web Service Batch Execution in Python

Create BAS test project in python

- Expeiment = Titanic 1 [Predictive Exp.]
- Python version = 2.7
- IDE = Pycharm

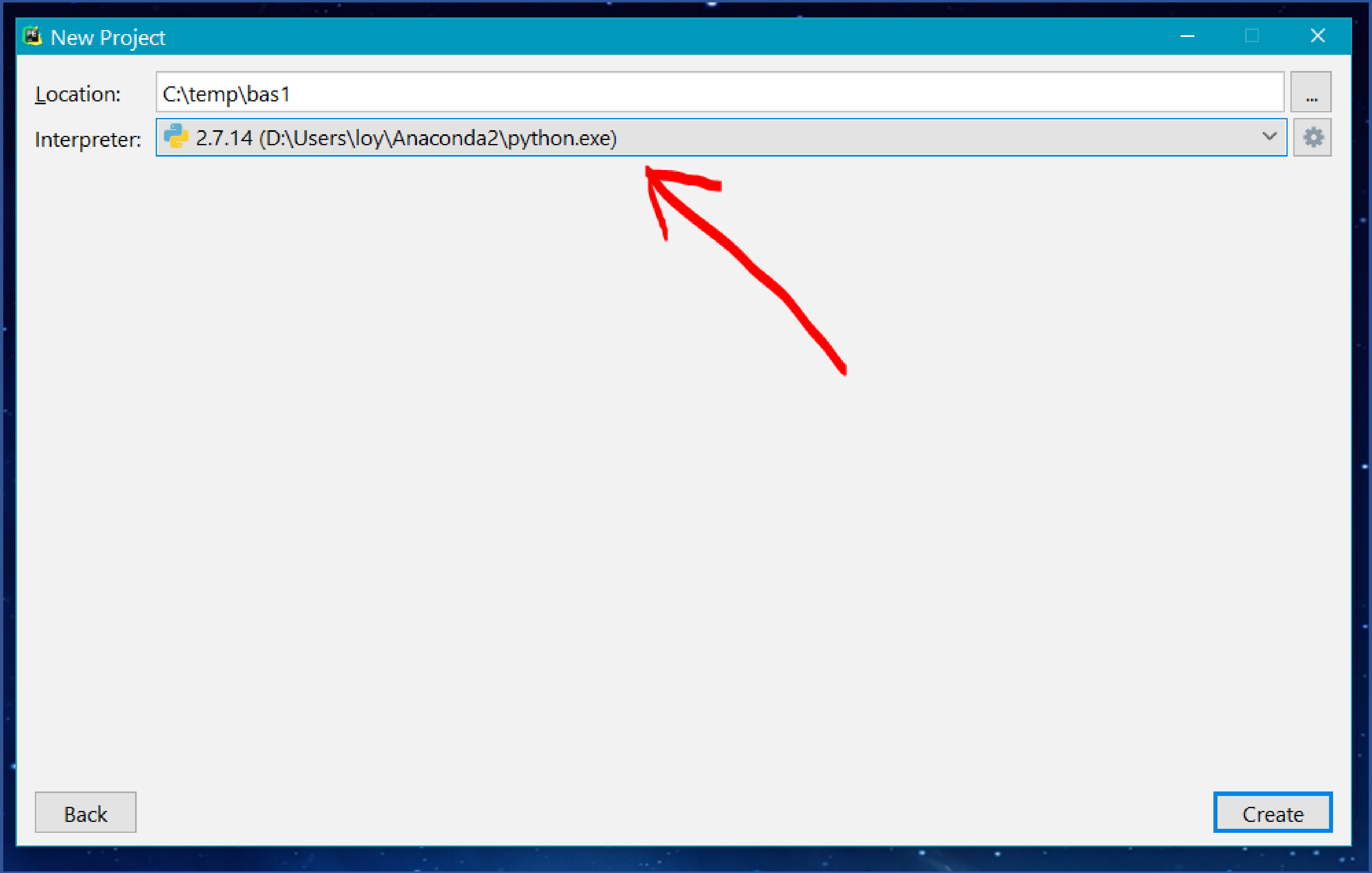
11610 Python BAS Titanic

Create new project



11610 Python BAS Titanic

Create project bas1 in python 2.7

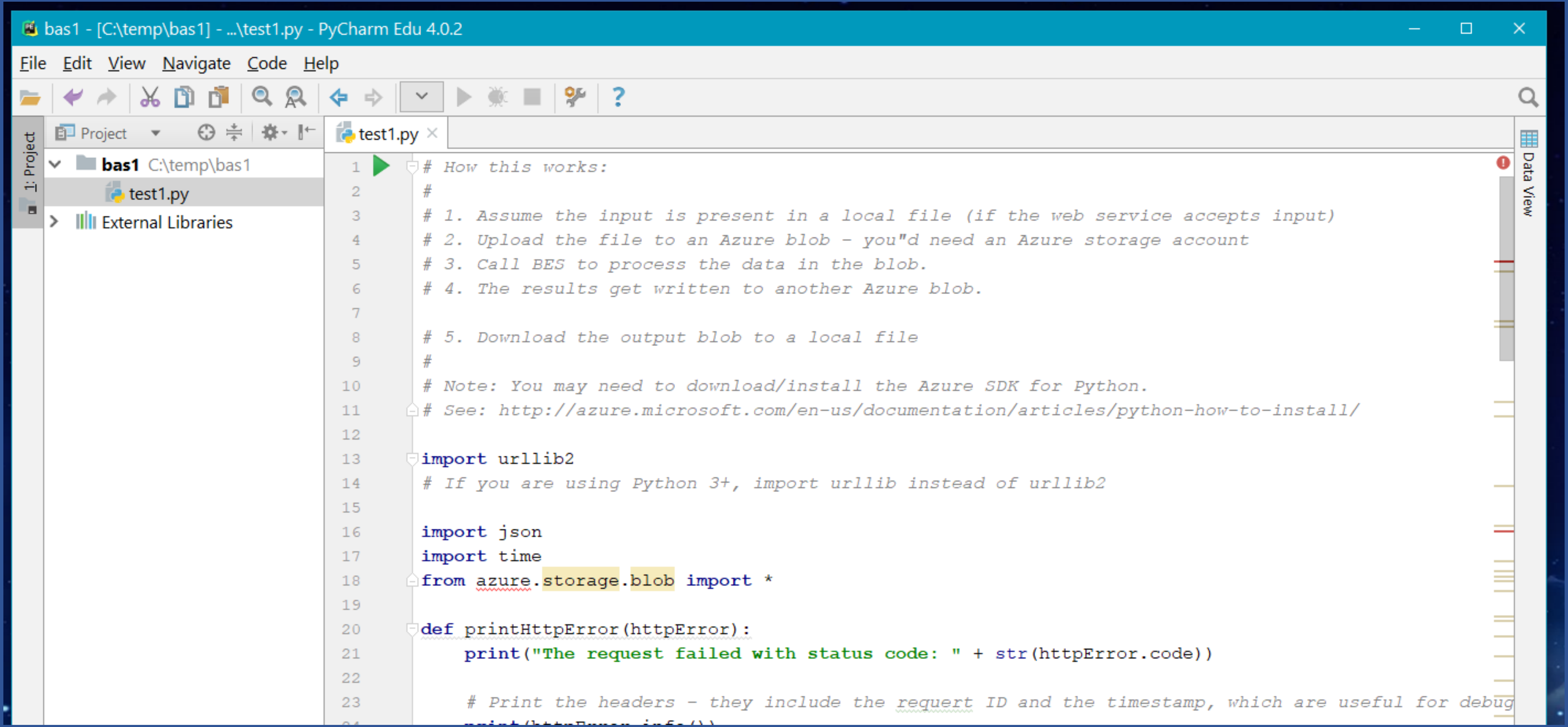


Location: C:\temp\bas1

Interpreter: 2.7.14 (D:\Users\loy\Anaconda2\python.exe)

11610 Python BAS Titanic

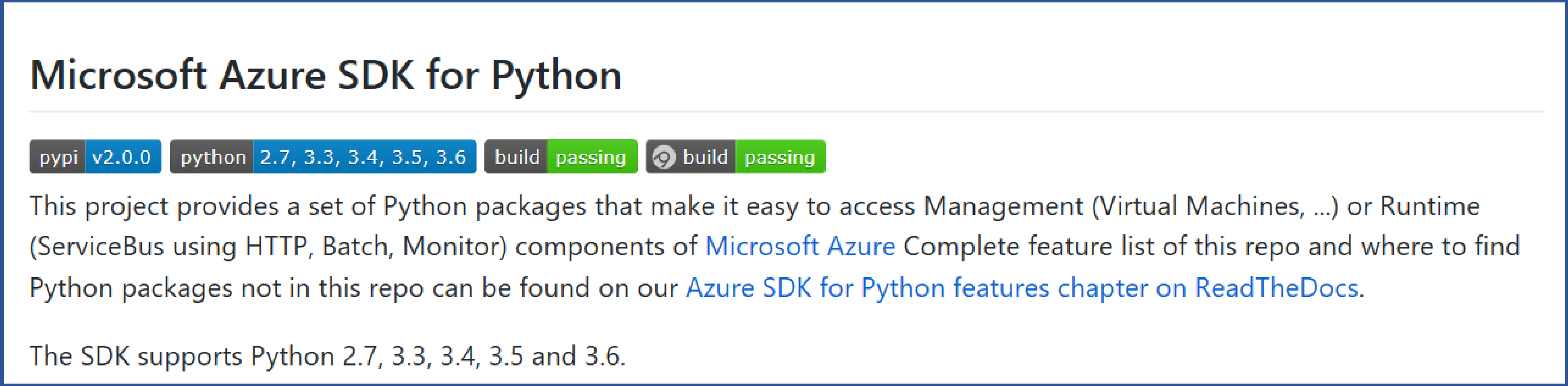
Add python file / add python code from BAS api sample page



```
# How this works:
#
# 1. Assume the input is present in a local file (if the web service accepts input)
# 2. Upload the file to an Azure blob - you"d need an Azure storage account
# 3. Call BES to process the data in the blob.
# 4. The results get written to another Azure blob.

# 5. Download the output blob to a local file
#
# Note: You may need to download/install the Azure SDK for Python.
# See: http://azure.microsoft.com/en-us/documentation/articles/python-how-to-install/

import urllib2
# If you are using Python 3+, import urllib instead of urllib2

import json
import time
from azure.storage.blob import *

def printHttpError(httpError):
    print("The request failed with status code: " + str(httpError.code))

    # Print the headers - they include the requert ID and the timestamp, which are useful for debug
```

11610 Python BAS Titanic

We need azure.storage.blob



Microsoft Azure SDK for Python

This project provides a set of Python packages that make it easy to access Management (Virtual Machines, ...) or Runtime (ServiceBus using HTTP, Batch, Monitor) components of Microsoft Azure Complete feature list of this repo and where to find Python packages not in this repo can be found on our Azure SDK for Python features chapter on ReadTheDocs.

The SDK supports Python 2.7, 3.3, 3.4, 3.5 and 3.6.

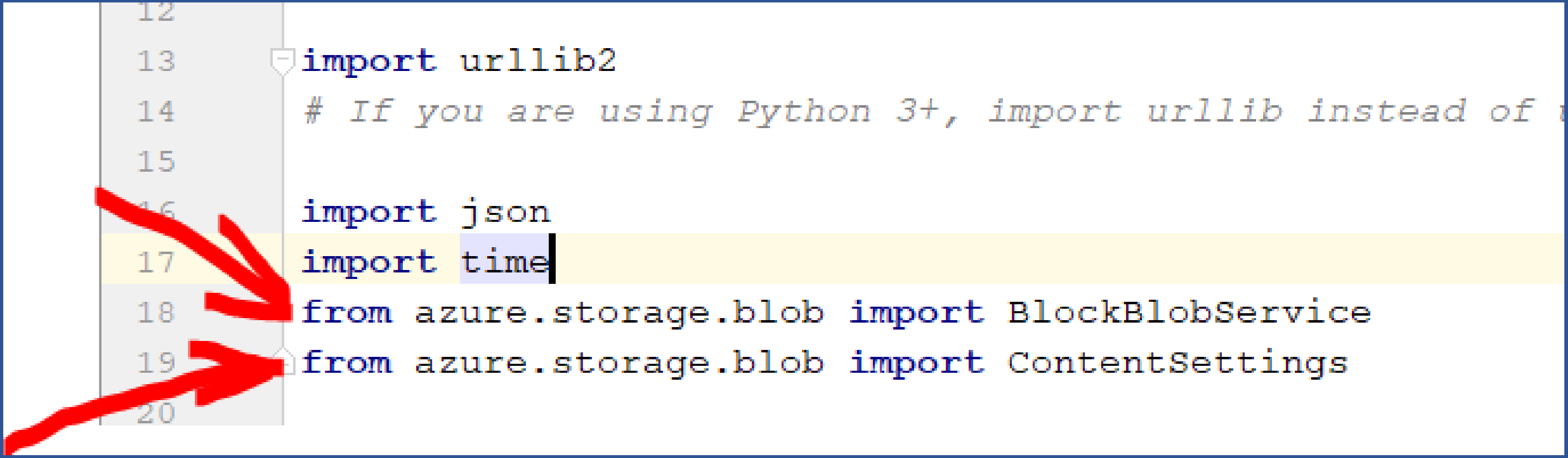
Install Azure SDK using PIP

- Go to interpreter folder
- Open CMD or Power shell
- pip install azure-storage-blob # Install the latest Storage management library
- OR
- pip install azure # install all package, take longer
- https://github.com/Azure/azure-sdk-for-python

- *** if package went to wrong folder, just copy all azure pages to the correct folder
- D:\Users\loy\Anaconda2\Lib\site-packages

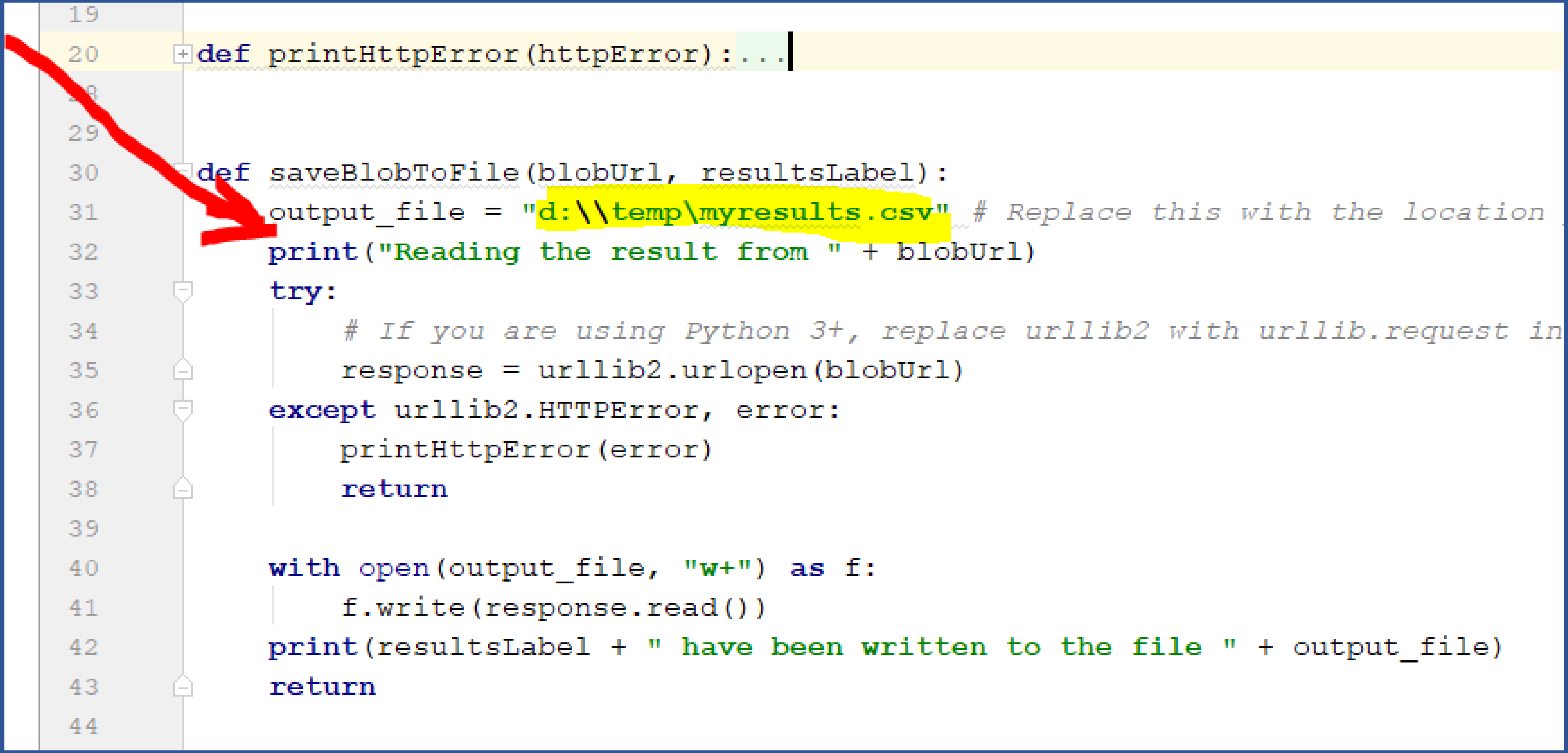
11610 Python BAS Titanic

Change from blob to BlockBlobService



```
import urllib2
# If you are using Python 3+, import urllib instead of u

import json
import time
from azure.storage.blob import BlockBlobService
from azure.storage.blob import ContentSettings
```

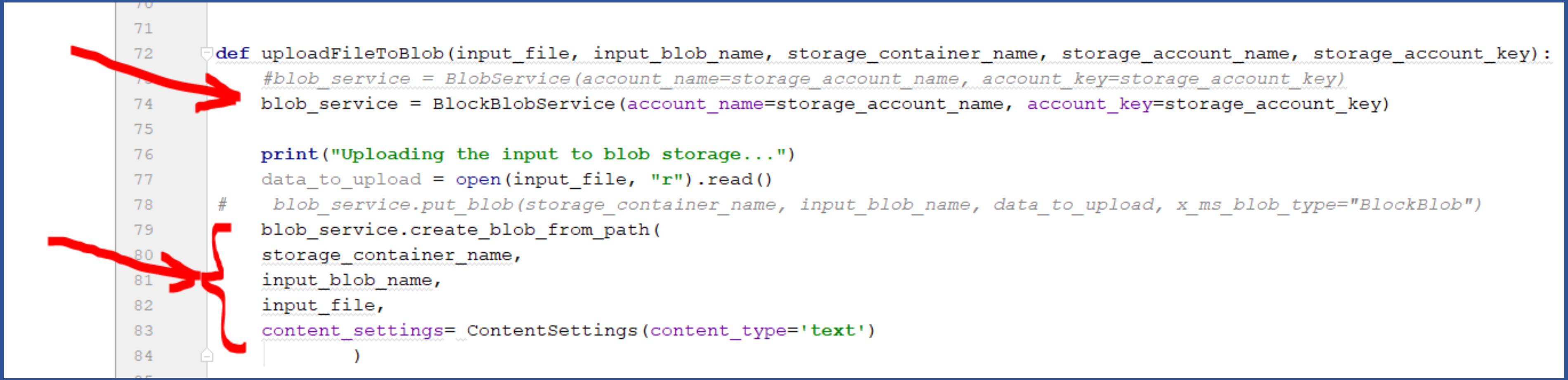
Set output file



```

def printHttpError(httpError):...


def saveBlobToFile(blobUrl, resultsLabel):
    output_file = "d:\\temp\myresults.csv" # Replace this with the location
    print("Reading the result from " + blobUrl)
    try:
        # If you are using Python 3+, replace urllib2 with urllib.request in
        response = urllib2.urlopen(blobUrl)
    except urllib2.HTTPError, error:
        printHttpError(error)
        return

    with open(output_file, "w+") as f:
        f.write(response.read())
    print(resultsLabel + " have been written to the file " + output_file)
    return

```

11610 Python BAS Titanic

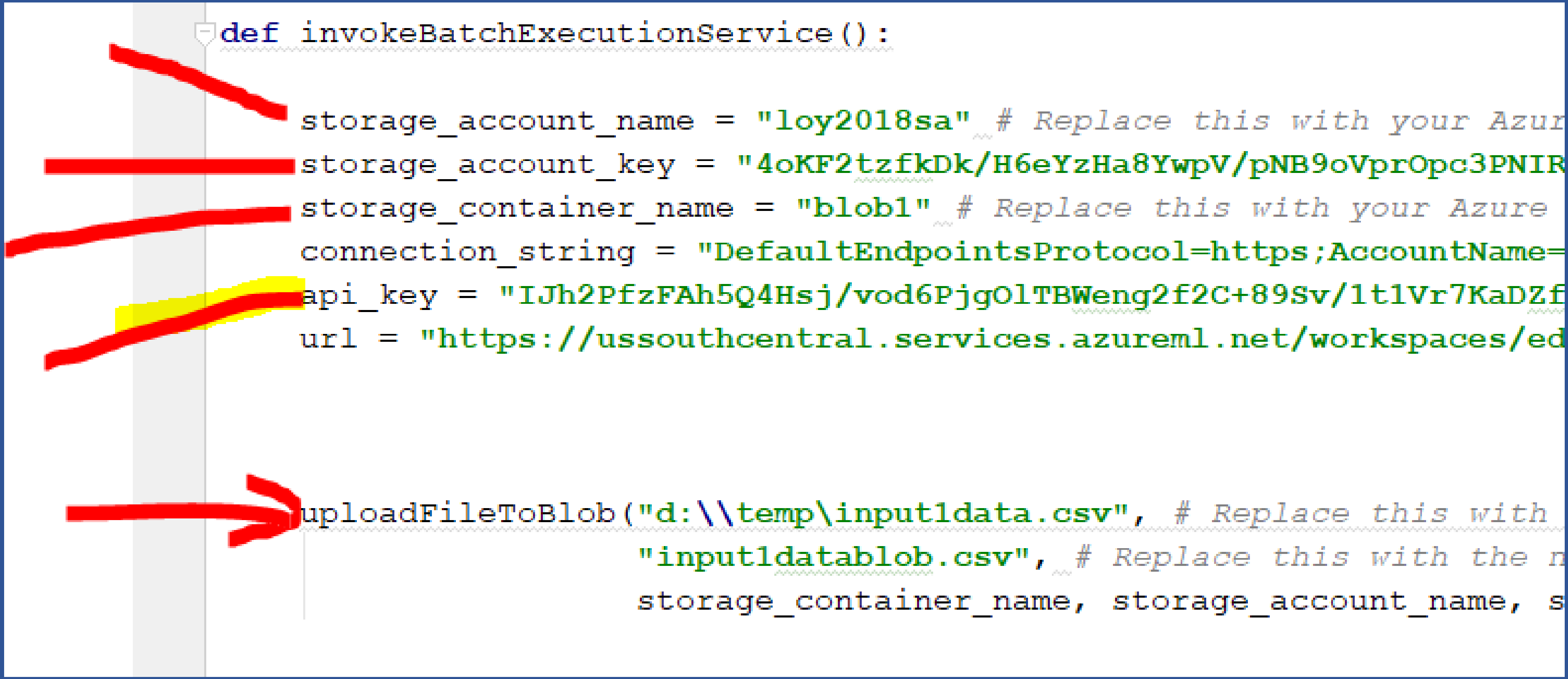
Set blob_service



```
def uploadFileToBlob(input_file, input_blob_name, storage_container_name, storage_account_name, storage_account_key):
    #blob_service = BlobService(account_name=storage_account_name, account_key=storage_account_key)
    blob_service = BlockBlobService(account_name=storage_account_name, account_key=storage_account_key)

    print("Uploading the input to blob storage...")
    data_to_upload = open(input_file, "r").read()
#     blob_service.put_blob(storage_container_name, input_blob_name, data_to_upload, x_ms_blob_type="BlockBlob")
    blob_service.create_blob_from_path(
    storage_container_name,
    input_blob_name,
    input_file,
    content_settings= ContentSettings(content_type='text')
            )
```

11610 Python BAS Titanic

Set account key



```
def invokeBatchExecutionService():

    storage_account_name = "loy2018sa" # Replace this with your Azur
    storage_account_key = "4oKF2tzfkDk/H6eYzHa8YwpV/pNB9oVprOpc3PNIR
    storage_container_name = "blob1" # Replace this with your Azure
    connection_string = "DefaultEndpointsProtocol=https;AccountName=
    api_key = "IJh2PfzFAh5Q4Hsj/vod6PjgOlTBWeng2f2C+89Sv/1t1Vr7KaDZf
    url = "https://ussouthcentral.services.azureml.net/workspaces/ed



    uploadFileToBlob("d:\\temp\input1data.csv", # Replace this with
                     "input1datablob.csv", # Replace this with the n
                     storage_container_name, storage_account_name, s
```

11610 Python BAS Titanic

Input file csv

- Need column title.
- No new line at the last letter (don’t press enter).
- Save this file to d:\\temp\input1data.csv

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
```

11610 Python BAS Titanic

Output result file written in to hard disk

```
Survived,PassengerClass,Gender,Age,SiblingSpouse,ParentChild,FarePrice,PortEmbarkation,Scored Labels,Scored Probabilities

0,3,male,22,1,0,7.25,S,0,1.76168905454688E-05

1,1,female,38,1,0,71.2833,C,1,0.999995827674866

1,3,female,26,0,0,7.925,S,1,0.979553580284119
```

11610 Python BAS Titanic

```
# How this works:
#
# 1. Assume the input is present in a local file (if the web service accepts input)
# 2. Upload the file to an Azure blob - you"d need an Azure storage account
# 3. Call BES to process the data in the blob.
# 4. The results get written to another Azure blob.

# 5. Download the output blob to a local file
#
# Note: You may need to download/install the Azure SDK for Python.
# See: http://azure.microsoft.com/en-us/documentation/articles/python-how-to-install/

import urllib2
# If you are using Python 3+, import urllib instead of urllib2

import json
import time
from azure.storage.blob import BlockBlobService
from azure.storage.blob import ContentSettings

def printHttpError(httpError):
    print("The request failed with status code: " + str(httpError.code))

    # Print the headers - they include the requert ID and the timestamp, which are useful for
debugging the failure
    print(httpError.info())

    print(json.loads(httpError.read()))
    return


def saveBlobToFile(blobUrl, resultsLabel):
    output_file = "d:\\temp\myresults.csv" # Replace this with the location you would like to
```

11610 Python BAS Titanic

```
use for your output file
    print("Reading the result from " + blobUrl)
    try:
        # If you are using Python 3+, replace urllib2 with urllib.request in the following code
        response = urllib2.urlopen(blobUrl)
    except urllib2.HTTPError, error:
        printHttpError(error)
        return

    with open(output_file, "w+") as f:
        f.write(response.read())
    print(resultsLabel + " have been written to the file " + output_file)
    return


def processResults(result):


    first = True
    results = result["Results"]
    for outputName in results:
        result_blob_location = results[outputName]
        sas_token = result_blob_location["SasBlobToken"]
        base_url = result_blob_location["BaseLocation"]
        relative_url = result_blob_location["RelativeLocation"]

        print("The results for " + outputName + " are available at the following Azure Storage
location:")
        print("BaseLocation: " + base_url)
        print("RelativeLocation: " + relative_url)
        print("SasBlobToken: " + sas_token)
```

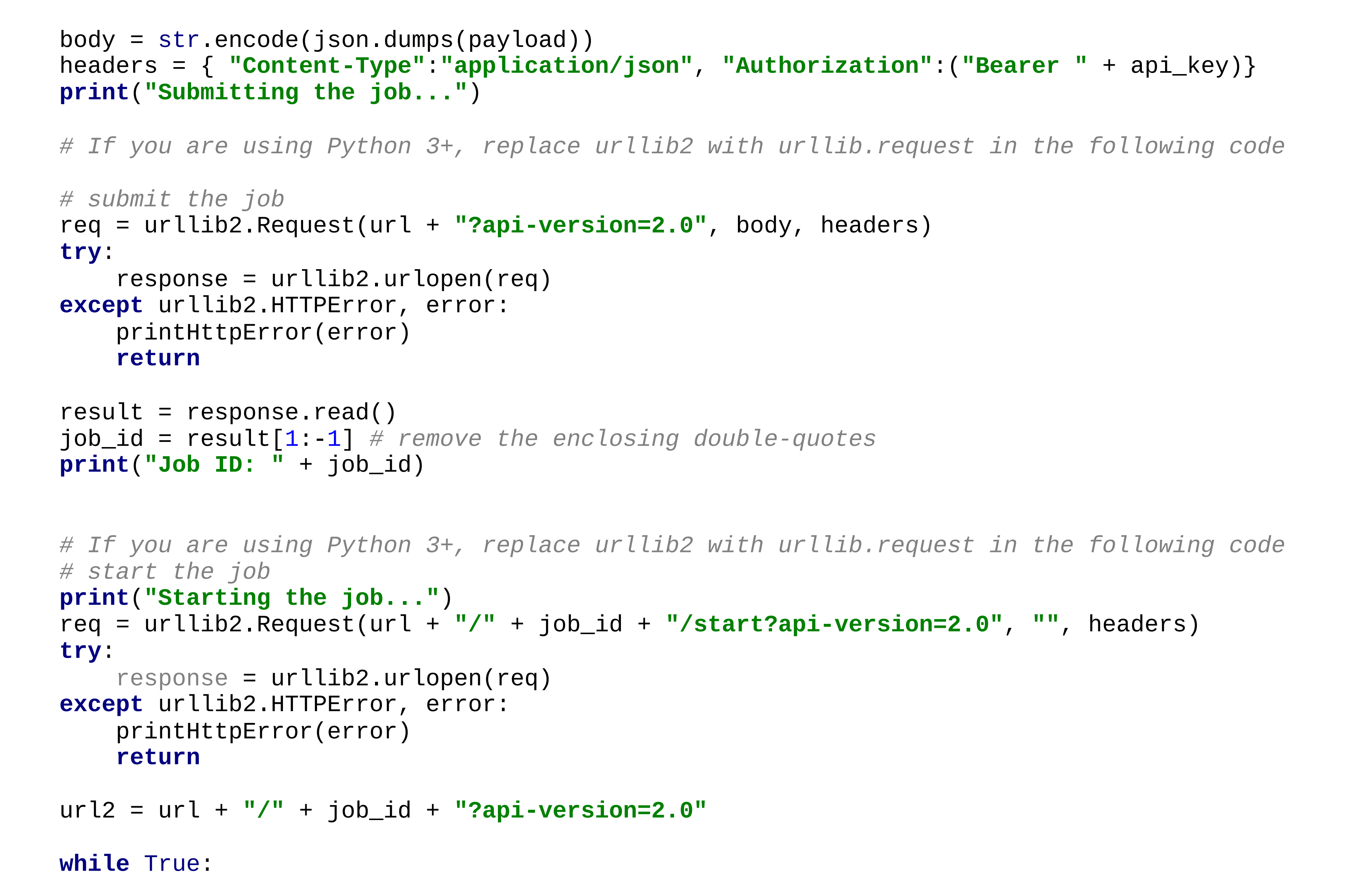
11610 Python BAS Titanic

```
        if (first):
            first = False
            url3 = base_url + relative_url + sas_token
            saveBlobToFile(url3, "The results for " + outputName)
    return



def uploadFileToBlob(input_file, input_blob_name, storage_container_name, storage_account_name, 
storage_account_key):
    #blob_service = BlobService(account_name=storage_account_name, 
account_key=storage_account_key)
    blob_service = BlockBlobService(account_name=storage_account_name, 
account_key=storage_account_key)

    print("Uploading the input to blob storage...")
    data_to_upload = open(input_file, "r").read()
#    blob_service.put_blob(storage_container_name, input_blob_name, data_to_upload, 
x_ms_blob_type="BlockBlob")
    blob_service.create_blob_from_path(
    storage_container_name,
    input_blob_name,
    input_file,
    content_settings= ContentSettings(content_type='text')
            )

def invokeBatchExecutionService():

    storage_account_name = "loy2018sa" # Replace this with your Azure Storage Account name
    storage_account_key = 
"4oKF2tzfkDk/H6eYzHa8YwpV/pNB9oVprOpc3PNIRrL/EduRP6/o2css1tX4p47ateS8AfT2DUetjgLv4Tr3hg==" # 
Replace this with your Azure Storage Key
    storage_container_name = "blob1" # Replace this with your Azure Storage Container name
```

11610 Python BAS Titanic

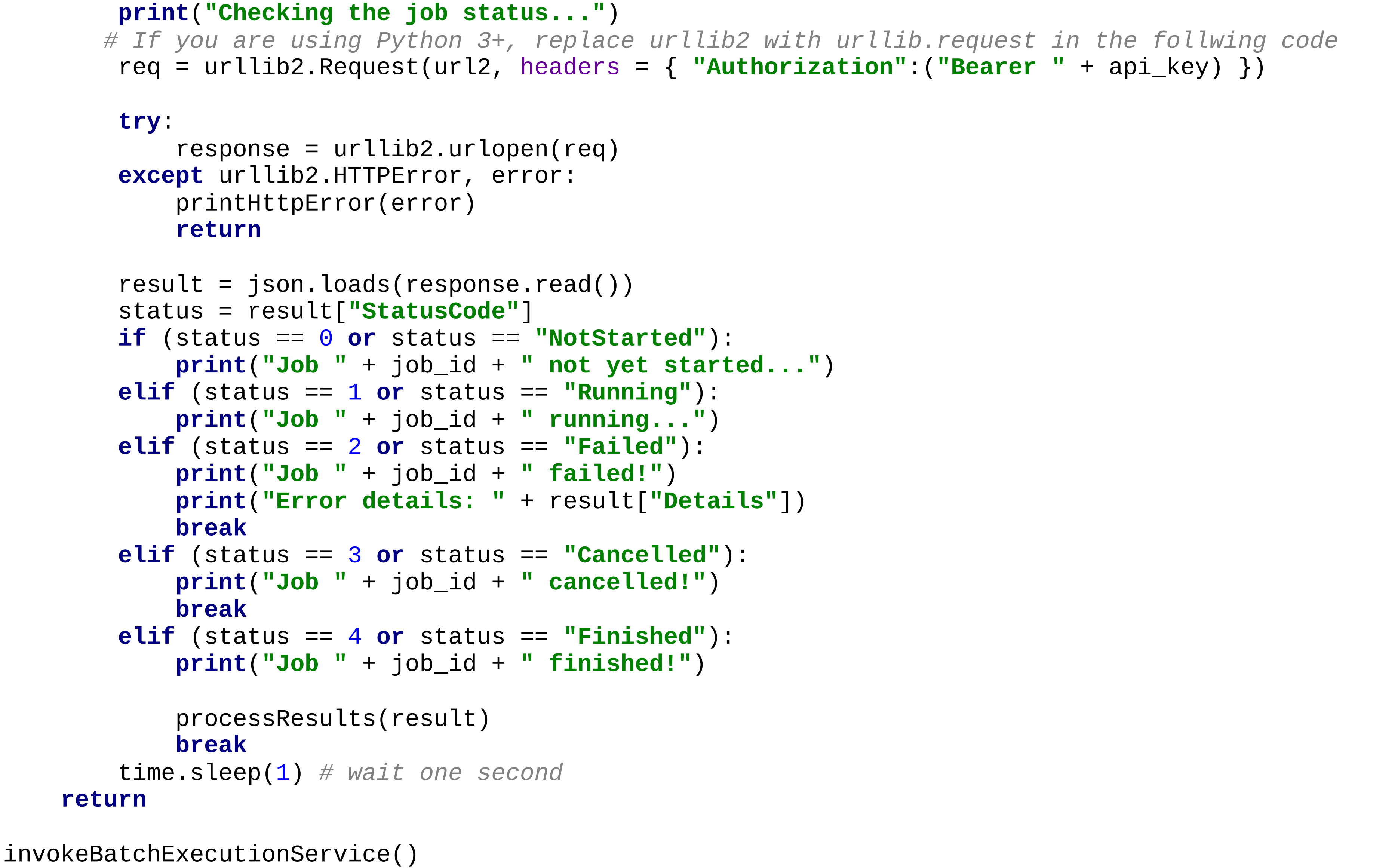
```
    connection_string = "DefaultEndpointsProtocol=https;AccountName=" + storage_account_name +
";AccountKey=" + storage_account_key
    api_key =
"IJh2PfzFAh5Q4Hsj/vod6PjgOlTBWeng2f2C+89Sv/1t1Vr7KaDZfequmXPzhAZNs9KjkaklAcSuRvTLy47/yw==" #
Replace this with the API key for the web service
    url =
"https://ussouthcentral.services.azureml.net/workspaces/ede12cb3aaf24c7e826493f4e309f1e1/service
s/ad3b577804c443d08f0f30b6c8028411/jobs"



    uploadFileToBlob("d:\\temp\input1data.csv", # Replace this with the location of your input
file
                     "input1datablob.csv", # Replace this with the name you would like to use
for your Azure blob; this needs to have the same extension as the input file
                     storage_container_name, storage_account_name, storage_account_key)

    payload =  {

        "Inputs": {

            "input1": { "ConnectionString": connection_string, "RelativeLocation": "/" +
storage_container_name + "/input1datablob.csv" },
        },

        "Outputs": {

            "output1": { "ConnectionString": connection_string, "RelativeLocation": "/" +
storage_container_name + "/output1results.csv" },
        },
        "GlobalParameters": {
}
    }
```

11610 Python BAS Titanic



```
body = str.encode(json.dumps(payload))
headers = { "Content-Type":"application/json", "Authorization":("Bearer " + api_key)}
print("Submitting the job...")

# If you are using Python 3+, replace urllib2 with urllib.request in the following code

# submit the job
req = urllib2.Request(url + "?api-version=2.0", body, headers)
try:
    response = urllib2.urlopen(req)
except urllib2.HTTPError, error:
    printHttpError(error)
    return

result = response.read()
job_id = result[1:-1] # remove the enclosing double-quotes
print("Job ID: " + job_id)


# If you are using Python 3+, replace urllib2 with urllib.request in the following code
# start the job
print("Starting the job...")
req = urllib2.Request(url + "/" + job_id + "/start?api-version=2.0", "", headers)
try:
    response = urllib2.urlopen(req)
except urllib2.HTTPError, error:
    printHttpError(error)
    return

url2 = url + "/" + job_id + "?api-version=2.0"

while True:
```

11610 Python BAS Titanic



```
        print("Checking the job status...")
        # If you are using Python 3+, replace urllib2 with urllib.request in the follwing code
        req = urllib2.Request(url2, headers = { "Authorization":("Bearer " + api_key) })

        try:
            response = urllib2.urlopen(req)
        except urllib2.HTTPError, error:
            printHttpError(error)
            return

        result = json.loads(response.read())
        status = result["StatusCode"]
        if (status == 0 or status == "NotStarted"):
            print("Job " + job_id + " not yet started...")
        elif (status == 1 or status == "Running"):
            print("Job " + job_id + " running...")
        elif (status == 2 or status == "Failed"):
            print("Job " + job_id + " failed!")
            print("Error details: " + result["Details"])
            break
        elif (status == 3 or status == "Cancelled"):
            print("Job " + job_id + " cancelled!")
            break
        elif (status == 4 or status == "Finished"):
            print("Job " + job_id + " finished!")

            processResults(result)
            break
        time.sleep(1) # wait one second
    return

invokeBatchExecutionService()
```

11610 Python BAS Titanic

More information

https://docs.microsoft.com/en-us/azure/storage/blobs/storage-python-how-to-use-blob-storage