One morning…. John receives a call from his bob his boss.

Bob: Hello, john when can you create 200 work plans for all our employees

John: 200 work plans? That’s going to take ages, or several days.

Bob: Can you do it before three days?

John: Uhmm... I’m not sure

Bob: well if you cannot update me, as soon as possible, then I’ll tell my manager.

*Bob Hangs up the phone…*

John (mumble): oh no! 200 documents, does this boss thinks I’m a robot or something?

**NEED TO MAKE 100’S OF WORKPLAN DOCUMENTS AT ONCE?**

This article will show you how to create multiple docx documents at once at the click of a finger using python

Before we begin, you have to install the required package with pip in your favorite IDE.

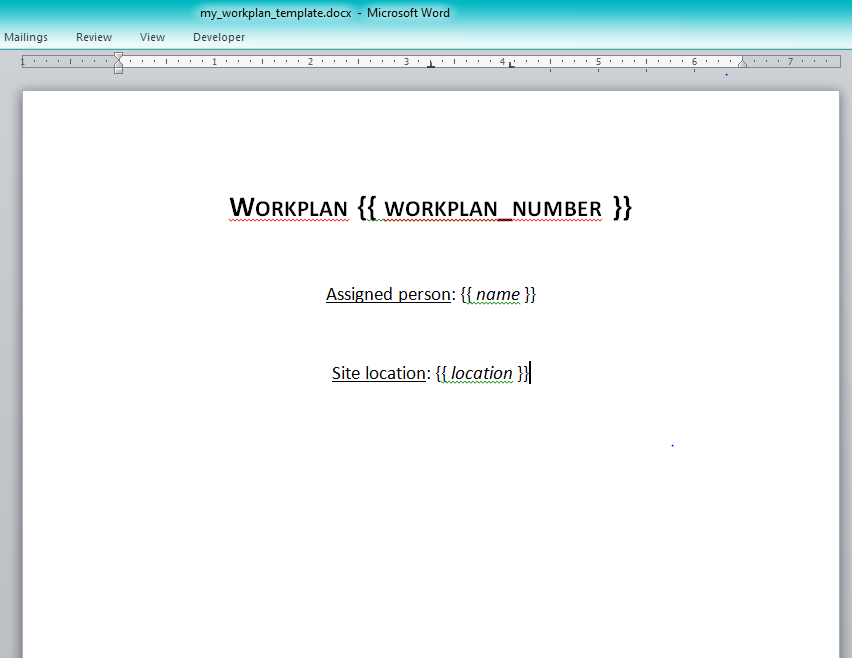
**STEP1: INSTALL THE DOCXTPL PACKAGE**

The package you will install is docxtpl (python docx)

So go to your terminal or command line and type in ***Pip install docxtpl***

**STEP 2: CREATE A WORKPLAN TEMPLATE IN WORD.**   
Create the Work plan template exactly how you intend the final output to look like. The variables you’ll be changing with the python code will be inside a double curly bracket in our template.

So our template will look like this

. 

As you can see from the image, the workplan\_Number, name and location are all in curly brackets. These are the variables that we’ll be changing with our python code.

now save it as “my\_workplan\_template” in the same file as your python script. or better still save your python script in the same location as your “my\_workplan\_template”.

STEP 3: MODIFY THE DEFAULT CODE ACCORDING TO THE TEMPLATE.

#DEFAULT CODE

from docxtpl import DocxTemplate

doc = DocxTemplate("my\_word\_template.docx")

context={'company\_name':"World company"}

doc.render(context)

doc.save("generated\_doc.docx")

so since we save our template with “my\_workplan\_template”. we will change the "my\_word\_template.docx" in the default code to "my\_workplan\_template"(every single value in the name must match including the spaces and underscores).

now the variable in the double curly braces in our template is "workplan\_number" so will change "company\_name" in the default template to **"workplan\_number"** so it will fit our template.

now we will change the "World\_company" in our default code to whatever we intend it to be in our template. in this case. lets put "100". next we will add the “name” variable is in our template into our code. note: what you type here must match what you have in your template. Next give the name variable what should appear in the template. in this case, we are going with "John".

next, we will add the "location" variable in the code and give it the value we want to appear in the template. in this case, lets go with "london"

so at the end, our code will appear as follows.

from docxtpl import DocxTemplate

doc = DocxTemplate("my\_workplan\_template.docx")

context = {'workplan\_number': "100", 'name':"John", 'location':"london"}

doc.render(context)

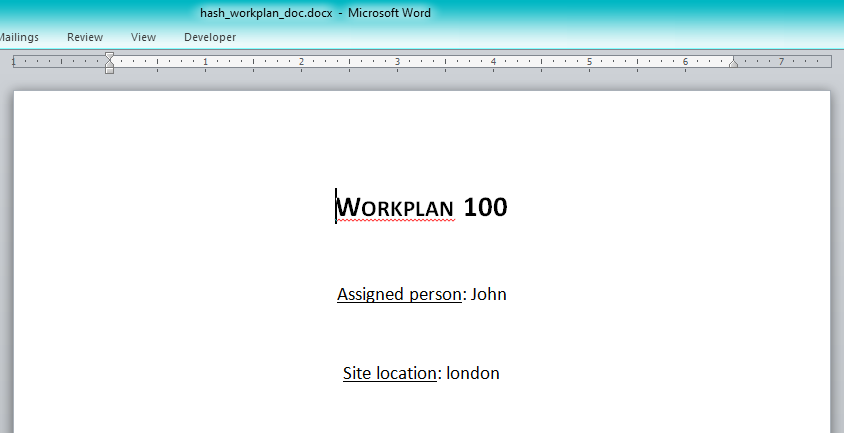
doc.save("hash\_workplan\_doc.docx")

now the python script will render out the document, and will substitute. the 100, John and london with the variables we’ve defined in our output file.

now rename the "generated\_doc" variable in the sample code to whatever you

want to call the file. in this case, we want to call the file, "hash\_workplan\_doc.docx"

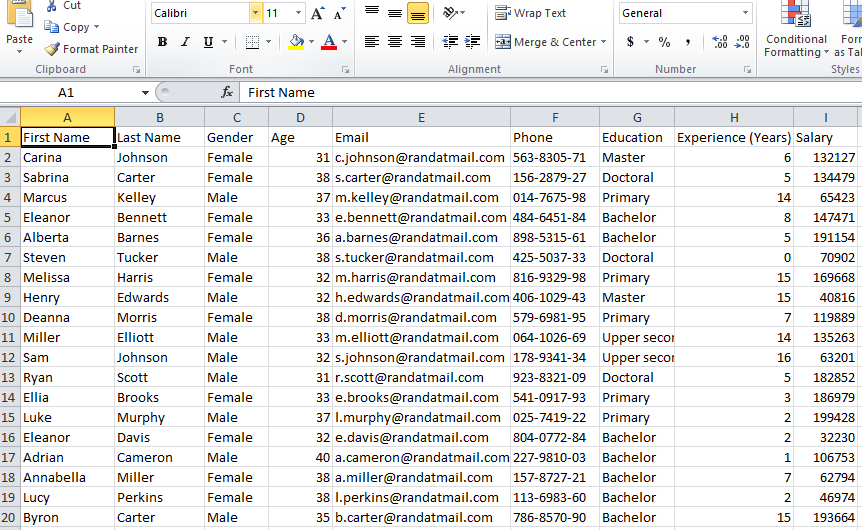
now run the code and if everything works as fine, you should have this as the output. (Insert output image.)



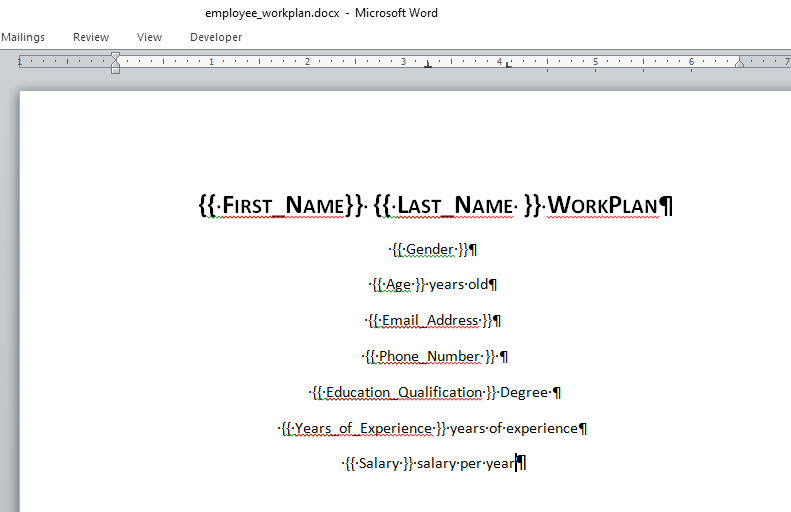
CREATING 200 WORKPLANS AT ONCE

Now we will take this a notch further and try to do this on 200 word files at once using data from a csv. the data we'll be substituting into our word files are all contained in the csv file.

so lets change the values in the docx template file to match the columns in our csv file.



In our word file, we will have this



so we will save the new word file as employee\_data.docx in the same file as the csv file and the python script

now we create a new python script and save it in the folder as the csv file and the word template file.

In your python IDE, you will first import the docxTemplate module. now after the import statement. Import the csv file containing the workplan data

**From docxtpl import docxTemplate**

**with open("persondata.csv", "r") as csvfile: #name the csvfile whatever you want.**

**file = csvfile.readlines()**

**#this will import the csv file and convert it into a list.**

**print(type(file))**

**<class 'list'>**

Now we’ll write a for-loop to split the list, extract each column and use it to create a new word file based on our template.

Remember that in our csv file, the first column is the column name, which we won’t be making use of. so we will tell python to ignore it.

So, therefore, we will add this to our forloop [1:]. This will tell python to start reading from index one instead of index 0

**for column in file [1:]:**

Now we want to separate each column in the csv file and assign it to a variable so that we can pass that into the context variable in the code.

Since, a csv file is just a comma separated file, we can tell python to separate each element and store it in the variable we will pass it.

**first\_name = column.split(",")[0] #first column**

**last\_name = column.split(",")[1] #second column**

**gender = column.split(",")[2] #thrid column**

**age = column.split(",")[3] #fourth column**

**email = column.split(",")[4] #fifth column**

**phone = column.split(",")[5] #sixth column**

**education = column.split(",")[6] #seventh column**

**experience = column.split(",")[7] #eight column**

**salary = column.split(",")[8] #ninth column**

**convert the salary to currency.**

In the csv file, we’ve imported, the salary column is made up of pure strings(numbers). However, we want to a dollar currency figure. To do that, you’ll use the locale module. The locale module is a default module in python, so you won’t need to install anything. Add this at the top of your code

**import locale**

**import re #this module will substitute substrings which match a certain pattern with another string.**

**locale.setlocale(locale.LC\_ALL, 'en\_US')**

**pattern = r'\d+(\. \d{1,2})?' #a regex pattern used to match any number**

now after this after the forloop

**salary\_locale = re.sub(pattern, lambda x:locale.currency(float(x.group()), grouping =True), salary)**

doc = DocxTemplate("my\_word\_template.docx"). so we will change the DocxTemplate file which is "my\_word\_template" to the name of our template file. in this case, we called it "employee\_data".

so well have doc = DocxTemplate("employee\_data.docx")

**creating the context**

#now for the context, we will change the values in our word documents to the values in our csv file.

in the below code, the values on A matches the values in double curly braces in our word template and the values on B matches the variable name we've assigned above to the column in the csv file.

#A = (placeholder name)

#B = (name of the variable extracted from the csv file)

A B

context={'First\_Name': first\_name,

'Last\_Name': last\_name,

'Gender': gender,

'Age': age,

'Email\_Address': email,

'Phone\_Number': phone,

'Education\_Qualification': education,

'Years\_of\_Experience': experience,

'Salary': salary\_locale}

doc = DocxTemplate("employee\_workplan.docx")

doc.render(context)

**Note:** the context is still inside the forloop created ealier.

**Saving the file**

Now, you’ll save each word file created from the template using an f-string. in this case we want to save each word file with the matching first name and last name of each employee. so we will write an f-string to do just that.

doc.save("f {workplan\_number} {first\_name} {last\_name} workplan.docx") #the first name and last name must be in curly brackets because we want to replace their values with the values in each column of our csv file.

**So at the end here is the code….**

from docxtpl import DocxTemplate

import locale

import re #this module will substitute substrings which match a certain pattern with another string.

locale.setlocale(locale.LC\_ALL, 'en\_US')

pattern = r'\d+(\. \d{1,2})?' #a regex pattern used to match any number

with open("employee\_data.csv", "r") as csvfile: #name the csvfile whatever you want.

file = csvfile.readlines()

for column in file [1:]:

workplan\_number = column.split(",")[0]#first column

first\_name = column.split(",")[1] #second column

last\_name = column.split(",")[2] #thrid column

gender = column.split(",")[3] #fourth column

age = column.split(",")[4] #fifth column

email = column.split(",")[5] #sixth column

phone = column.split(",")[6] #seventh column

education = column.split(",")[7] #eight column

experience = column.split(",")[8] #ninth column

salary = column.split(",")[9] #tenth column

salary\_locale = re.sub(pattern, lambda x:locale.currency(float(x.group()), grouping =True), salary)

context={'First\_Name': first\_name,

'Last\_Name': last\_name,

'Gender': gender,

'Age': age,

'Email\_Address': email,

'Phone\_Number': phone,

'Education\_Qualification': education,

'Years\_of\_Experience': experience,

'Salary': salary\_locale}

doc = DocxTemplate("employee\_workplan.docx")

doc.render(context)

doc.save(f'{workplan\_number} {first\_name} {last\_name} workplan.docx')

print(f'{workplan\_number} {first\_name} {last\_name} workplan.docx')

here is our 200 workplan done automatically with python….

