Final Project In the making - ctec 121

First I made some basic pseudocode…

Final Project - ctec121 Hunter Nelson

Program is to act as a MadLibs story teller

A Defining diagram

Input	Processing	Output
Names, places, things, activities, ext. Selection of story type	Prompt user for variables to be used later (in a interactive manner opposed to repetitive questioning) Prompt for story type Get variables Get story type Print pre formatted story that corresponds with story type, using variables respectively to input verbs, additives, ext.	A story using given variables that fits requested story line

B Solution algorithm

MadLibs

prompt for name, pets, hobbies, locations, friends, actions, ect
prompt for StoryLine

get name, pets, hobbies, locations, friends, actions, ect
get StoryLine

If Var(any variable) = "" OR var = null THEN
generate random one from array

print StoryLine (i.e. name "went on a hike with his pet" petName "in the local" place...)

END

****note: needed fields will simply have to be figured out as story lines are chosen...***
User should be promted to try other story lines of end the program after each story

With that in place I created a short list, so I could work through each step without having to simultaneously be figuring out what would come next…

Modular list of things I need to create in hierarchal list:

1. prompt user for inputs
2. prompt for story line
3. list of fallback variables incase inputs were left blank
4. function that grabs RANDOM words from list (of fallback inputs)
5. Printing of story
6. prompt to carry on or end

*I would like things to be VERY modular

With all of this in place I simply started coding out the start of the program that would get the inputs to be used later from the user…

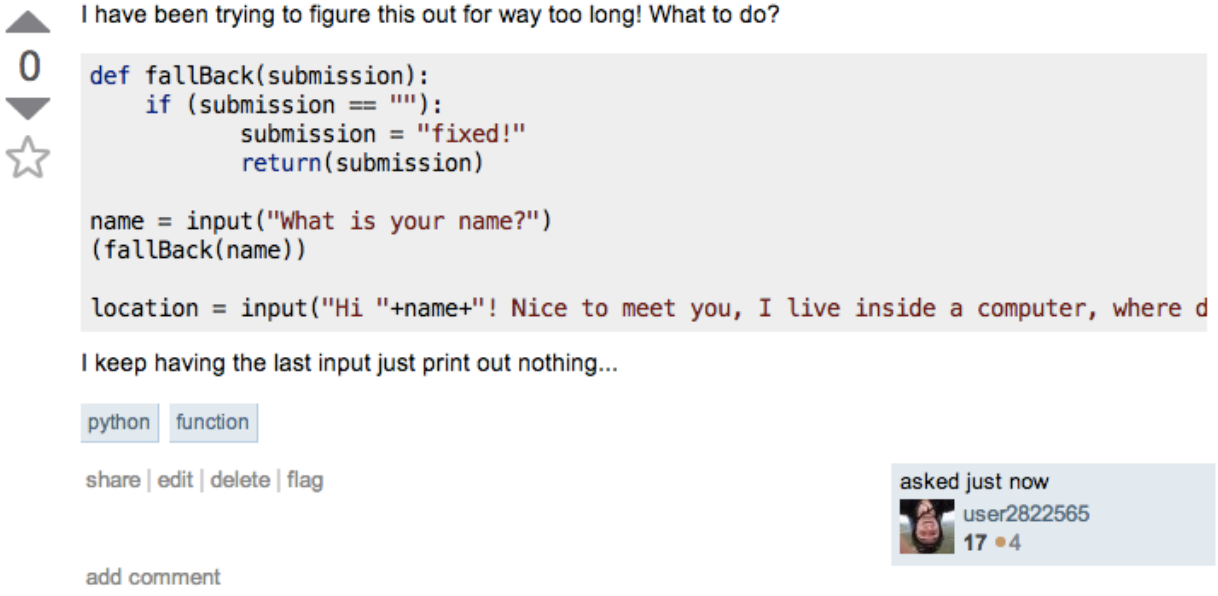
```
#Final Project ctec121
#Madlibs program
#By: Hunter Nelson


#start user prompt (variables to be used in story)

#list of needed variables: Name, location, hobbie, friendsName, RelitivesName, animal, celeberity, feeling,
print("I am a MadLibs program, if at you do not enter a responce then a random word will be selected for you")
print(" ")

name = input("What is your name?")
location = input("Hi "+name+"! Nice to meet you, I live inside a computer, where do you live?")
feeling = input("Interesting, I have never been too "+location+" what are the people like there?") # "the man was" feeling
#start finding animalName
animalTest = input("I have been told that I am "+feeling+" aswell! I bet my pet mouse would agree with that, do you have any pets? (yes or no)").lower()
if (animalTest == "yes"):
    animalNum = eval(input("Cool how many pets do you have?"))
    if animalNum == 1:
        animalName = input("Whats the name of your pet?")
    elif animalNum == 2:
        animalName == input("Whats the name of your favorite pet?")
    elif animalNum < 0 :
        print("I dont know how you could end up with "+animalNum+" pets... and that worrys me")
    else:
        animalName("Ok thats a lot of pets! Whats the name of your favorite one?")
        print("I like the name, "+animalName+"! Thanks for sharing that! Hmm, what else would I like to know...")
elif (animalTest ==  "no"):
    print("Yeah pets can be a handful... ")
else:
    print("Your responce confuses me... umm...")
#end finding animalName

bestFriend = input("Well out of your friends...Who is your best friend?")
activity = input("You and "+bestFriend+" must have lots of fun together! What is your favorite activity to do with "+bestFriend+", like (i.e. running, go
relitive = input("Interesting... being a computer I have never got the chance to try"+activity+"... I dont have any family either... but you must, what's
#end user prompt
#now I have: name, location, feeling, animalName, BestFriend,relitive

print("You actually just picked a 'favorite relitive'! Even computers know your not supposed to do that! Lets start with the mad libs before things start
```

The code was ready for every circumstance (including negative pet amounts!) except for blank inputs. Instead of putting out an error if blank, I simply wanted to have the program place a fitting word in the variable. This would create a nice option for the user, if they want to save time/effort!

To do this I would first need a function to run with every input, checking for filled forms…

```
#fallback word function
fallBack(submission):
    if submission = "":
            submission = "fixed!"  #will be submission[random(0,arrayLength)]
```

I had a lot of trouble getting the function to work…



I have been trying to figure this out for way too long! What to do?

```
def fallBack(submission):
    if (submission == ""):
            submission = "fixed!"
            return(submission)

name = input("What is your name?")
(fallBack(name))

location = input("Hi "+name+"! Nice to meet you, I live inside a computer, where d
```

I keep having the last input just print out nothing...

python function

share | edit | delete | flag

asked just now
user2822565
17 •4

add comment

A visit to stack overflow.com cleared everything up…

You need to store the result of `fallBack()` . Also, change `fallBack()` to return the original value if it is non-null:

```
def fallBack(submission):
    if not submission:
        return "fixed!"
    else:
        return submission
```

Then, use it like this:

```
name = fallBack(input("What is your name?"))
```

share | edit | flag

answered 11 mins ago
Joel Cornett
9,497 •1 •12 •25

Thats beautiful! Thanks so much! – user2822565 just now edit

Now that I had an input being put in the right place… I needed the correct input! I decided that this would mean having a name, place, or action pulled from a list randomly…

I started by simply making one list of names and doing a little testing to be sure I could use them as I intended…

```
#list of inputs per input type (to be used when no entry was given)

names = ("Bob","Bruce","Ryan","ASAP Rocky", "Timmy", "Mom", "Dad", "Rando")
print(names[1])
print(len(names))
```

```
I
```

What I had to do now was the output of fallback to a random name from “names” using the random function with the range being 0 - the length of names, which I had already found useable.

```
#list of inputs per input type (to be used when no entry was given)

names = ("Bob","Bruce","Ryan","ASAP Rocky", "Timmy", "Mom", "Dad", "Rando")
print(names[1])
print(len(names))


#random imput generator (for blank inputs)

from random import randint
randint(2,9) #Inclusive

print(names[randint(0,len(names))])
```

Nailed it, first time!

Now I needed to expand this so it worked for more than just names, which starts with lists for actions, feelings, locations…

```
#list of inputs per input type (to be used when no entry was given)

names = ("Bob","Bruce","Ryan","ASAP Rocky", "Timmy", "Mom", "Dad", "Rando")
feelings = ("sad","happy","confused","weird", "insane", "lost", "nice", "too nice")
locations = ("Canada","Hell","Africa","Cape Horn", "Clark College", "Washington", "Oregon", "another dimension")
actions = ("drumming","sking","skating","sleeping", "flying", "eating", "killing", "enhancing")

#END list of inputs
```

With those in place I had to set up my code to be able to choose which list to pick from based of its own needs i.e. names get names and feelings get feelings

This would require an addition to my fallBack function that checked for the input being worked on and grabbing a random word from the PROPER list…

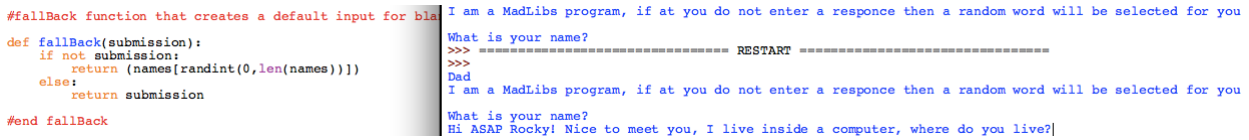
The first bump I hit in trying to do this was that I needed to know the variables name instead of its value as it came in, as of now print(name) would print the value and through my fallBack function I would be getting the value by the name of “submission”.

The solution started with this I found… that I was looking at way (WAY!) too much work.

So I instead I figured that I could just pass a second peramiter into my fallBack function that matched the type of input I was looking for.

expl. location = fallBack((locations)(input(“hgfcjhgcjhgcjh”))

Before doing so I made sure I could return a random name…



```
#fallBack function that creates a default input for bla

def fallBack(submission):
    if not submission:
        return (names[randint(0,len(names))])
    else:
        return submission

#end fallBack
```

```
I am a MadLibs program, if at you do not enter a responce then a random word will be selected for you

What is your name?
>>> ================================ RESTART ================================
>>>
Dad
I am a MadLibs program, if at you do not enter a responce then a random word will be selected for you

What is your name?
Hi ASAP Rocky! Nice to meet you, I live inside a computer, where do you live?
```

It worked! So I got a second parameter in there!

```
#fallBack function that creates a default input for blank submissions during the user prompt

from random import randint

def fallBack(subType, submission):
    if not submission:
        return (subType[randint(0,len(subType))])
    else:
        return submission

#end fallBack
```

It was so cool to see it working!

```
What is your name?
Hi Mom! Nice to meet you, I live inside a computer, where do you live?
Interesting, I have never been too Clark College what are the people like there?
```

Finally! I can make a few stories!

```
Once upon a time there was a elephant named Bruce who was owned by a man named Mom. In the city of Washington was the Ryan zoo, it was here that Bruce would spend each day flying.
Feeling weird Bruce decided to run off and chase his/her dreams of flying. In sharing plans with ASAP Rocky it became clear to Bruce that it would be really hard to do this. So Br
uce decided to stay at the zoo.
>>>
```

Some final touches…

I created a system to securely prompt the user for a story choice

```
stories = [story1, story2];

storiesNum = len(stories)

#print(storiesNum)

def storyChoicePrompt():
    choice = eval(input('We have '+str(storiesNum)+' stories to choose from... pick one (enter a number 1-'
    return choice


def storyChoiceCheck():
    if (choice > storiesNum) or (choice < 0) or (choice==""):
        print("Errored input! Try again...")
        storyChoicePrompt()
        storyChoiceCheck()
    else:
        print ("\n",stories[choice-1])
        file = open("newfile.txt", "w")

        file.write(stories[choice-1])
```

You will notice that I put the output into idle and an external file…