Download (right-click, save target as ...) this page as a jupyterlab notebook from: Lab4

Laboratory 4: Input and Output

Medrano, Giovanni

R11521018

ENGR 1330 Laboratory 4 - In-Lab

In [36]:
```
# Preamble script block to identify host, user, and kernel

import sys
! hostname
! whoami
print(sys.executable)
print(sys.version)
print(sys.version_info)
```

```
DESKTOP-6HAS1BN
desktop-6has1bn\medra
C:\Users\medra\anaconda3\python.exe
3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
sys.version_info(major=3, minor=8, micro=5, releaselevel='final', serial=0)
```

Input

Useful programs take input and generate output

So here we practice capturing input from the keyboard (most likely)

First lets explore the `input()` function

In [1]:
```
# Case 1 - without a prompt
fries=input()
print('Data type  =: ',type(fries))
print('Data value =: ',fries)
```

```
Data type  =:  <class 'str'>
Data value =:  35
```

In [3]:
```
##### Case 2 - without a prompt - force data type
```

```
fries=input()
fries=str(fries) # force into a string
print('Data type  =: ',type(fries))
print('Data value =: ',fries)
```

```
Data type  =:  <class 'str'>
Data value =:  35
```

In [4]:
```
# Case 3 - without a prompt - force data to string, make string all upper case
fries=input()
fries=str(fries) # force into a string
fries=fries.upper()
print('Data type  =: ',type(fries))
print('Data value =: ',fries)
```

```
Data type  =:  <class 'str'>
Data value =:  APPLE
```

Exercise

Modify Case 3 to make the **fries** numeric (integer or float), then run the script to validate the modification

In [5]:
```
# Case 3 - without a prompt - force data to string, make string all upper case
fries=input()
fries=str(fries) # force into a string
fries=fries.upper()
fries=int(fries)
print('Data type  =: ',type(fries))
print('Data value =: ',fries)
```

```
Data type  =:  <class 'int'>
Data value =:  311
```

About the prompt

It is a good idea to tell the user what to input. You can do this by putting the hint in quotes inside the input parentheses. The hint will come in the next line and will wait for the user input. You can then type the input and when you hit the ENTER key, it will capture the input.

In this example, "tell me a beautiful number" is the hint. This gets printed in the next line when asking for the input. If you type 6, this will be assigned to the variable beautiful_number which we can print later.

In [6]:
```
# Example
beautiful_number = input("tell me a beautiful number ")
print(beautiful_number)
```

```
6
```

In [7]:
```
# Case 1 - with a prompt
fries=input('Would you like fries with that? Enter Y or N')
print('Data type  =: ',type(fries))
print('Data value =: ',fries)
```

```
Data type  =:  <class 'str'>
Data value =:  Y
```

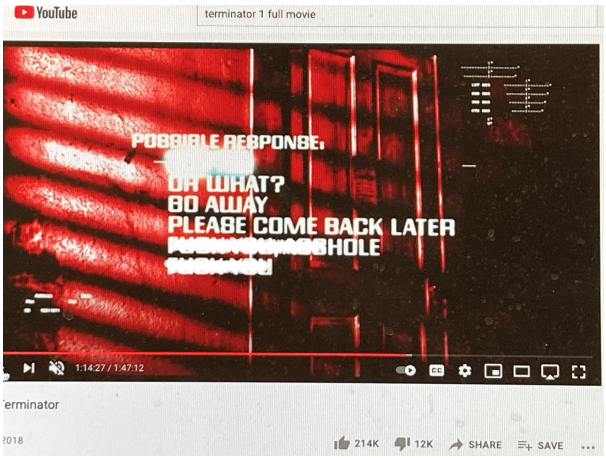
In [8]:
```
# Case 3 - with a prompt - force data to string, make string all upper case
fries=input('Would you like fries with that? Enter Y or N')
fries=str(fries) # force into a string
fries=fries.upper() # force to upper
print('Data type  =: ',type(fries))
print('Data value =: ',fries)
```

```
Data type  =:  <class 'str'>
Data value =:  N
```

The prompt can be any string variable

In [9]:
```
response = 'Would you like fries with that? Y or N'
fries=input(response)
fries=str(fries) # force into a string
fries=fries.upper() # force to upper
print('Data type  =: ',type(fries))
print('Data value =: ',fries)
```

```
Data type  =:  <class 'str'>
Data value =:  N
```



Or the prompt can be drawn from a list of prompts like in the Cyberdyne 101 shown above in its remote desktop mode (from Terminator (1985) approximately at timestamp 1:14:07)

Below the script might look something like (remember the terminator is responding to an input from the apartment super, and is responding). The point of the example is that the prompt in the `input()` function can be drawn from a list rather than retyping each use.

In [10]:
```
#Cyberdyne 101
prompt = ['YES','NO','OR WHAT?','GO AWAY','PLEASE COME BACK LATER','placeholder','place
mystring = input(prompt[4])
print(mystring)
```

ok

Exercise

Run the script below a few times. It should output a randomly selected integer from 0 to 6. Then copy the Cyberdyne prompt list above to the location shown, replace the last two placeholder strings with the responses from the movie (in the image above). Then run the script 6 times; how many times does the terminator respond with your replacements for the placeholders?

In [11]:
```
import random
low = 0 ; high = 6
rand = random.randrange(low,high) #generate random number in range low to high
myprompt= prompt[rand]
# copy cyberdyne prompt=[...] here
# print prompt element as determined by myprompt chosen at random
mystring = input(myprompt)
print(mystring)
```

```
rent is due
```

Example. Suggestive Sell!

Suppose we are building an automated food service program. If you recall from real life, that when you complete your order the cashier always asks "would you like fries with that?"

Well lets script the interaction, and get either a yes or no answer from the customer.

In [12]:
```
statements = ['Thank you for your order','Would you like fries with that too? (Y or N)'
'Excellent, I\'ll add them to your order, Have a nice day!',
'Are you sure? They are really yummy (Y or N)',
'Ok, no fries!  \nHave a nice day cheapskate',
'You did not enter a Y or N, I can\'t take this ambiguity, I quit']
print(statements[0])
fries = str(input(statements[1])).upper() # input with prompt, typecast into string, fo
# selection
if fries == 'Y':
    print(statements[2])
elif fries == 'N':
    print(statements[3])
    fries = str(input(statements[1])).upper() # input with prompt, typecast into string
    if fries == 'Y':
        print(statements[2])
    elif fries == 'N':
        print(statements[4])
else:
    print(statements[5])
```

```
Thank you for your order
```

```
You did not enter a Y or N, I can't take this ambiguity, I quit
```

In these examples we don't operate on the input much, however lets conclude the beautiful number example, and do something with it

In [13]:
```
# Example
beautiful_number = input("tell me a beautiful number ")
beautiful_number = float(beautiful_number)
print('The beautiful number %.3f squared is %.5f' %(beautiful_number,beautiful_number*b
```

```
The beautiful number 3.000 squared is 9.00000
```

Exercise/Discussion

How could you approximate the square root using just the tool (script) above?

Try it for:

- 100
- 4
- 2
- 204.490

In [14]:
```
# Discuss ideas, then implement a brute force approach
myNumber = input("input a number as a guess for square root")
myNumber = float(myNumber )
print('Your beautiful number %.3f squared is %.3f' %(myNumber, myNumber*myNumber))
```

```
Your beautiful number 14.300 squared is 204.490
```

About Output

Later on we will learn about printing to a file (here we use the word print as a surrogate for output, not necessarily actually printing something); the remainder of this lab explores some of the necessary ideas involved in output

We have already examined and played with "Hello World" as shown below - but I added some interaction.

In [15]:
```
name = input("What is your name?")
print("Hello, ",name)
```

```
Hello,  bender
```

We can also output values from a list such as

In [16]:
```
prompt = ['YES','NO','OR WHAT?','GO AWAY','PLEASE COME BACK LATER','placeholder','place
print(prompt[0])
print(prompt[1])
print(prompt[2])
print(prompt[3])
```

```
YES
NO
OR WHAT?
GO AWAY
```

Or the list can be numeric

In [17]:
```
numblist = [0,1,2,3,4,5]
print(numblist)
print(type(numblist[:]))
```

```
[0, 1, 2, 3, 4, 5]
<class 'list'>
```

Print blank lines by using the *newline* symbol, which is `\n` .

In [18]:
```
print ("\n\n\n\n\ttab\n\n\n\n\n")
```

```
	tab
```

Example

Let's build a script thats asks the users name, then says hello. Skips 8 lines and identify ourself as Cyberdyne Terminator Model 101.

The work flow is nothing more than

- Prompt for a name
- Echo the name
- Add 8 blank lines
- Identify as Cyberdyne

Using the bullet list as our structure generator we can start with

In [ ]:
```
# - Prompt for a name
# - Echo the name
# - Add 8 blank lines
# - Identify as Cyberdyne
```

Now copy-paste our prompt example

In [19]:
```
# - Prompt for a name
name = input("What is your name?")
# - Echo the name
print("Hello, ",name)
# - Add 8 blank lines
# - Identify as Cyberdyne
```

```
What is your name?gio
Hello,  gio
```

Next lets print 8 linefeeds

In [20]:
```
# - Prompt for a name
name = input("What is your name?")
# - Echo the name
print("Hello, ",name)
# - Add 8 blank lines
print ("\n\n\n\n\n\n\n\n\n")
# - Identify as Cyberdyne
```

```
What is your name?gio
Hello,  gio
```

Now we add our response.

In [21]:
```
# - Prompt for a name
name = input("What is your name?")
# - Echo the name
print("Hello, ",name)
# - Add 8 blank lines
print ("\n\n\n\n\n\n\n\n\n")
# - Identify as Cyberdyne
print('''My name is Bender, I am a Cyberdyne Terminator Model 101
disguised as a bending robot, my favorite beverage is Alby Bock,
Have you seen any living humans around here? death to all humans!''' )
```

```
What is your name?Giovanni
Hello,  Giovanni


My name is Bender, I am a Cyberdyne Terminator Model 101
disguised as a bending robot, my favorite beverage is Alby Bock,
Have you seen any living humans around here? death to all humans!
```

Notice the step-by-step build of the script; obviously you dont need to copy and paste so many cells, but here we want to be overly anal as we learn the JupyterLab environment.

Formatting the output

In the lecture we learned about formatting using the **%** operator and the **format()** method. They are roughly the same, I think the **%** approach is a little easier to read when maintaining scripts, but you may decide otherwise.

As an example we can combine the Terminator example and beautiful number example.

In [22]:
```
# - Prompt for a name
name = input("What is your name?")
# - Identify as Cyberdyne
print('''Hello %s My name is Bender, \n I am a Cyberdyne Terminator Model 101
disguised as a bending robot, \n my favorite beverage is Alby Bock,
Have you seen any living humans around here? death to all humans!''' %(name) )
beautiful_number = input("So %s tell me a beautiful number " %(name))
beautiful_number = float(beautiful_number)
print('The beautiful number %.3f squared is %.3f' %(beautiful_number,beautiful_number*b
```

```
Hello Giovanni My name is Bender,
 I am a Cyberdyne Terminator Model 101
disguised as a bending robot,
 my favorite beverage is Alby Bock,
Have you seen any living humans around here? death to all humans!

The beautiful number 15.000 squared is 225.000
```

About ending the line

By default, print function in Python ends with a newline. This function comes with a parameter called 'end.' The default value of this parameter is `\n`, i.e., the new line character. You can end a print statement with any character or string using this parameter, but you have to specify in the print statement.

In [23]:
```
print ("Welcome to", end = ' ') # ends with a null character, so NO LINEFEED
print ("Bender's world", end = '!')
```

```
Welcome to Bender's world!
```

Exercise

Explore the %.3f format code above, what happens if you change the numeric part of the format code?

the number of decimal places changes.

In [ ]:
```

```