

Scripting Mod 01 Tutorial – Python Basics and Flow Control

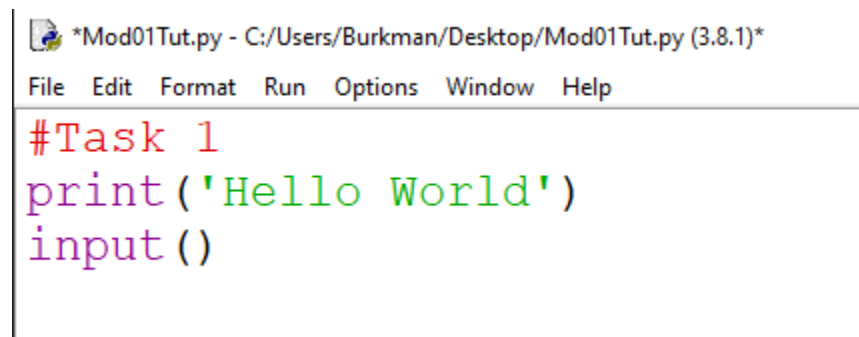
Our goal here is to write a script that let us play with these new. So our first step is to open IDLE on your Win10 machine. Tutorial examples are likely to be different than you might see on your machine. You can find IDLE on your virtual machine desktop.

With IDLE open, File – New File. Save this on the desktop as Mod01Tutorial. Now that it is saved we just have to hit F5 to run our program after saving changes. Leave both the file window and the Python shell window open. We will type our commands in the file window (not the shell window).

Start your script (all Python scripts that we write this semester in Windows) with your first and last name commented. On the second line add a comment indicating #Tutorial 1
Make a print statement to print out Hello World
Make an input statement on a line all by itself

Run this with F5, and you'll get a prompt to save then your script will run. It should print "Hello World!" in your shell then wait for you to hit the enter key before closing. We always want to build our script this way, with the use of an empty input to wait on the user (and always at the end of the script). If you close out the IDLE window and the shell window, then navigate to your new script on the desktop and double click it, it should run in a black command line window.

Right click your script and "edit in IDLE". Let's go to the line above print and type #Task 1. That's a comment and we need them in our scripts so that others can know what's going on. Our code is super obvious now but as we write more complicated scripts we will want those comments to help us remember what we were doing. My script so far looks like this:



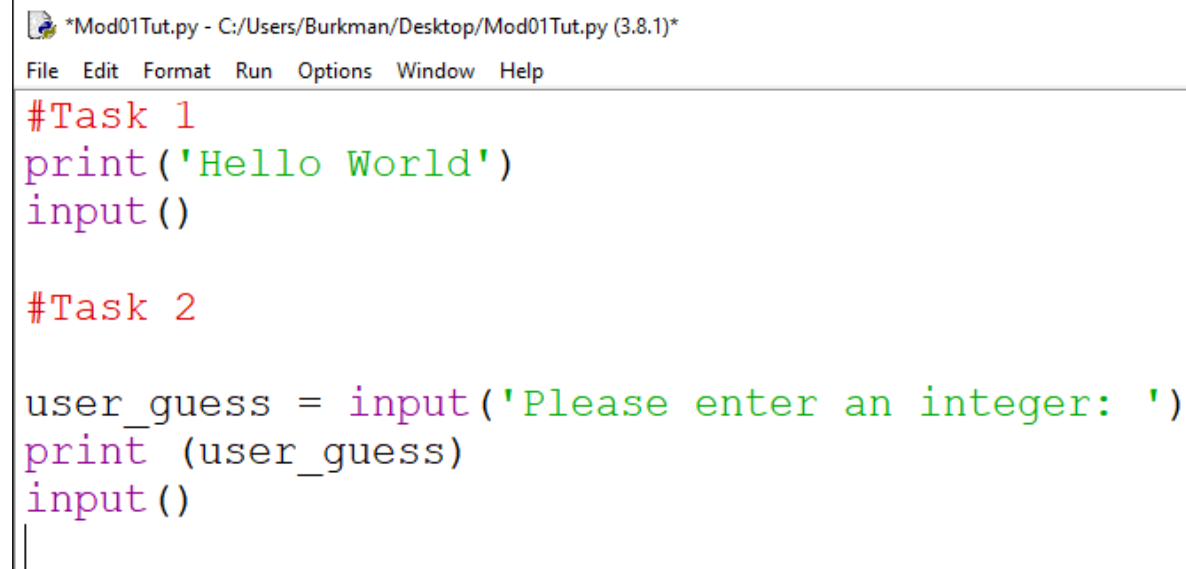
```
*Mod01Tut.py - C:/Users/Burkman/Desktop/Mod01Tut.py (3.8.1)*  
File Edit Format Run Options Window Help  
#Task 1  
print('Hello World')  
input()
```

Task 2 (comment this in your script):

Set a variable user_guess and give a nice prompt like 'Please enter an integer: ' Notice that I have a colon and space after the word integer to make this look nice. On the next line print the variable user_guess. Then add an empty input.
Your code should use this kind of format:
dog = input('My nice prompt: ')

Run your script

My script so far looks like this:



```
*Mod01Tut.py - C:/Users/Burkman/Desktop/Mod01Tut.py (3.8.1)*
File Edit Format Run Options Window Help

#Task 1
print('Hello World')
input()

#Task 2

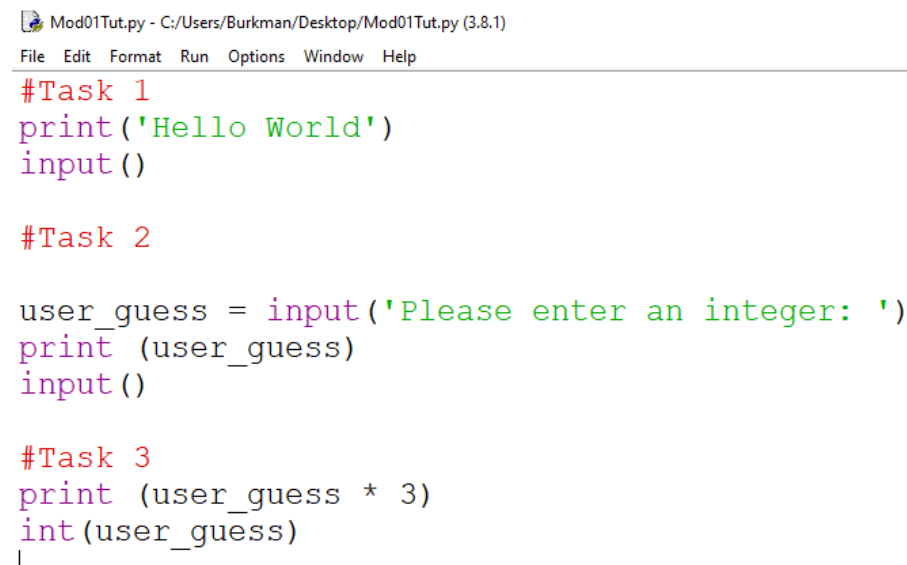
user_guess = input('Please enter an integer: ')
print (user_guess)
input()
|
```

Task 3 (comment this in the script)

Print out `user_guess` multiplied by 3 and run your script. It looks like that because all user input is a string. Just prior to your Task 3 print statement put this conversion line in:

```
int(user_guess)
```

And do that same print statement that you did before the conversion (`user_guess times 3`).



```
Mod01Tut.py - C:/Users/Burkman/Desktop/Mod01Tut.py (3.8.1)
File Edit Format Run Options Window Help

#Task 1
print('Hello World')
input()

#Task 2

user_guess = input('Please enter an integer: ')
print (user_guess)
input()

#Task 3
print (user_guess * 3)
int(user_guess)
|
```

What the heck? We still get the user input concatenated together 3 times. That's because conversion doesn't change the variable permanently. We will need to assign the conversion to a new variable (or to

itself and that will permanently change it). Replace your Task 3 lines with these four lines and run your script:

```
user_guess = int(user_guess)
converted_user_guess = int(user_guess)
print(user_guess * 3)
print(converted_user_guess * 3)
```

Cool. Now let's add these lines to convert user_guess back to a string and run the script again:

```
user_guess = str(user_guess)
print(user_guess * 3)
```

There are other explicit conversions but these are the most important starting out. You can't do math on strings and you cannot concatenate numbers. Try this:

```
print('I entered ' + user_guess)
print('I entered ' + converted_user_guess)
```

*Note: don't copy/paste. The single quotes in Word aren't the right kind and you'll get errors. Besides, you learn by typing code, not pasting code.

Run your script. You'll see that the second line threw an error because you are trying to concatenate an integer onto a string. Remember this, because a lot of errors happen when people don't pay attention to data types.

Here's my Task 3, which is part of my whole script:

```
#Task 3
user_guess = int(user_guess)
converted_user_guess = int(user_guess)
print(user_guess * 3)
print(converted_user_guess * 3)
user_guess = str(user_guess)
print(user_guess * 3)
print('I entered ' + user_guess)
print('I entered ' + converted_user_guess)
```

Task 4:

I didn't need to remind you to add a comment, right?

We will combine some concepts to identify some numbers as even or odd. First let just crank out some numbers with a FOR loop and RANGE. Look back at the slides. We want numbers from 1 to 20. Print them out to make sure you are getting what you want.

*TIP: I highlighted all the previous tasks on my script in IDLE and hit ALT and 3 to comment them out. Alt and 4 will uncomment. This is under “Format” in the menu options. This way I can run my script without repeating my work over and over. Just be sure to uncomment when done.

```
for i in range (1,21,1):  
    print(i)
```

Remember that we can count backwards by making the start number bigger than the second number, then making the third number a negative number. Also remember that the range doesn't include the ending number.

Ok, we need a logical way to see if each number is even or odd. Change your existing print to print `i % 2` and run your program. Or whatever variable you used where I used `i`. The percentage sign is modulus, and a number mod 2 divides our variable by 2 and tells us what is left. Handy. Even numbers return zero, odd numbers return 1.

Delete your print statement. Let's add some logic to capture that by adding an IF statement right after the FOR statement. FOR gets us items one at a time. When we use RANGE it returns one item from the range at a time. So we will use an IF statement to see if mod 2 of the variable equals zero. If so, we will print out the word Even. Then we will use an else statement to print out the word Odd. If the IF statement doesn't work out to be TRUE then the else statement happens. Try this on your own before looking at my code below.

< See Task 4, Figure 1 at the end of the document >

We do this a lot. Iterate (go through one at a time) a collection of things, and do something (or many things) to each individual thing. We can add to this with an elif statement to catch the number 7 and print the word Snowflake, but print Odd for all the other odd numbers and Even for all the even number. Try this before looking at my code.

< See Task 4, Figure 2 at the end of the document >

It's top-down logic. I want to look for the most unusual thing first then at the end the “default” or most common thing. This is a very important concept with IF statements. Think of it as a series of increasingly smaller screens. The first screen captures the largest rocks but lets the rest pass through. The next screen capture slightly smaller rocks and lets the rest pass through, etc. If you did it the opposite way, with the first screen being a very fine mesh, it would just capture all the rocks and let nothing pass through for further evaluation.

Task 5:

Copy your Task 4 code. Instead of using the number 21, get input from the user. Give a nice prompt. Print 'Lucky' if the number 7 shows up and 'Unlucky' if the number 13 shows up. Print Even or Odd for all the other numbers (not for 7 or 13).

Did you remember to convert the input for the range? Did you remember to add +1 to the range limit so we capture the last digit that the user wanted? Did you add an elif? Note that if we wanted the looping of the FOR statement to stop for like the number 13, we would replace that print statement with the work BREAK. Here's my solution:

< See Task 5, Figure 1 at the end of the document >

Task 6:

If we want to do any task (any chunk of code) multiple times we just throw it in a WHILE loop. Let's make a WHILE True loop. Inside the loop get user input for a last name (include a nice prompt) and get out of the loop when the name matches your last name. Try this on your own before looking at my code.

< See Task 6, Figure 1 at the end of the document >

Task 7:

We can also use a while statement with a counter and branching. Set a counter variable equal to zero. Then make a while statement that is true as long as that variable is less than ten. While that is true, print out the counter variable. Then increment the counter variable by one. You'll want to look back at the slides. Try this on your own without looking at my code below:

< See Task 7, Figure 1 at the end of the document >

Task 8:

One last thing. Import the random library (import random). Normally we would put all of our imports on the first line of our script but we'll make an exception here. Make a variable and set it equal to random.randint (-10,10). Randint is a function of random, and we are specifying the low number and the high number. It returns integers. Print the variable with the random number. Now put that code in a FOR loop that runs five times. We won't actually use the numbers in the FOR loop. We'll just use it to loop a certain number of times. Try this on your own before looking at my code below:

< See Task 8, Figure 1 at the end of the document >

This covers the big parts of flow control. Go back and

- ✓ uncomment all of your code that should run
- ✓ have your script nicely print out each Task number.
 - Include a new line when you do that like print ('\nsome words') and it will make a new line at the beginning.

- ✓ Comment out the bad line on Task 3 with a # and make a line to comment that it throws an error.
- ✓ Change Task 5 to tell the user to enter a number greater than 13. You don't have to check to see if they actually did that.
- ✓ Change the input message on Task 6 to include your last name for easier grading.
 - Mine looks like `user_name = input('Please enter a last name (Burkman): ')`
- ✓ Make Task 8 look nicer by adding `end = " "` after the variable with a comma. Like `print(my_variable, end=" ")`.
- ✓ After Task 8 add two lines of `print()`, then a line that prints Press Enter to end the script, then an empty input. I'll explain why we needed the two lines of `print()` when we review.

My script output begins on the next page. Your output must match, except for the obvious input differences and their outcomes. Remember to backup your script to the sftp server at 192.168.1.3

Script Output

Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:/Users/Burkman/Desktop/Mod01Tut.py =====

Task 1

Hello World

Task 2

Please enter an integer: 4

4

Task 3

12

12

444

I entered 4

Task 4

Odd

Even

Odd

Even

Odd

Even

Snowflake

Even

Odd

Even

Odd

Even

Odd

Even

Odd

Even

Odd

Even

Odd

Even

Task 5

Enter a number greater than 13: 14

Odd

Even

Odd

Even
Odd
Even
Lucky
Even
Odd
Even
Odd
Even
Unlucky
Even

Task 6

Please enter a last name (Burkman): Smith

Please enter a last name (Burkman): Burkman

Task 7

0
1
2
3
4
5
6
7
8
9

Task 8

4 6 1 -2 2

Press Enter to end the script

>>>

Tutorial Hidden Figures

Task 4, Figure 1:

```
#Task 4
for i in range (1,21,1):
    if (i % 2) == 0:
        print('Even')
    else:
        print ('Odd')
```

Task 4, Figure 2:

```
#Task 4
for i in range (1,21,1):
    if i == 7:
        print('Snowflake')
    elif (i % 2) == 0:
        print ('Even')
    else:
        print ('Odd')
```

Task 5, Figure 1

```
#Task 5
for i in range(1,int(input('Enter a number greater than 13: '))+1,1):
    if (i) == 7:
        print('Lucky')
    elif (i) == 13:
        print('Unlucky')
    elif (i % 2) == 0:
        print('Even')
    else:
        print('Odd')
```

Task 6, Figure 1

```
while True:
    user_name = input('Please enter a name: ')
    if user_name == 'Burkman':
        break
```

My IF statement didn't need an else because I only wanted something to happen if my one condition was met. But I could have done this as well:

```
while True:
    user_name = input('Please enter a name: ')
    if user_name == 'Burkman':
        break
    else:
        continue
```

Task 7, Figure 1

```
counter = 0

while counter < 10:

    print (counter)
    counter += 1
```

Task 8, Figure 1

```
import random

for i in range (0,5):
    random_value = random.randint(-10,10)
    print (random_value)
```
