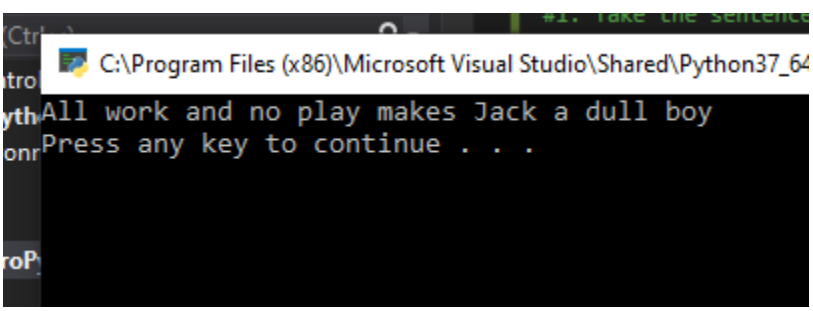
CIS-298 Intro to Python
With Professor Robert Mann
HW #1
Student: Demetrius Johnson
17 January 2023
Due: 17 January 2023 at 4pm

Send your code and output snippet showing results for the 8 questions starting on page 21 of the textbook.

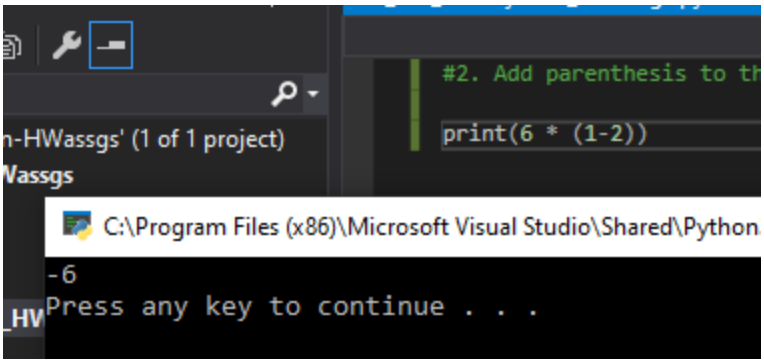
1. Take the sentence: All work and no play makes Jack a dull boy. Store each word in a separate variable, then print out the sentence on one line using print.

```
#1. Take the sentence: All work and no play makes Jack a dull boy. Store each word in a
separate variable,
#then print out the sentence on one line using print.

sentence = "All work and no play makes Jack a dull boy"
split_sentence = sentence.split()

word1 = split_sentence[0]
word2 = split_sentence[1]
word3 = split_sentence[2]
word4 = split_sentence[3]
word5 = split_sentence[4]
word6 = split_sentence[5]
word7 = split_sentence[6]
word8 = split_sentence[7]
word9 = split_sentence[8]
word10 = split_sentence[9]

print(word1 + ' ' + word2 + ' ' + word3 + ' ' + word4 + ' ' + word5
      + ' ' + word6 + ' ' + word7 + ' ' + word8 + ' ' + word9 + ' ' + word10)
```



2. Add parenthesis to the expression 6 * 1 - 2 to change its value from 4 to -6.

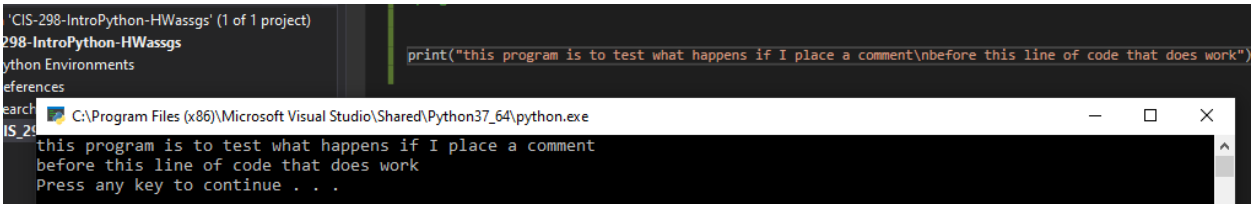
```
#2. Add parenthesis to the expression 6 * 1 - 2 to change its value from 4 to -6.

print(6 * (1-2))
```



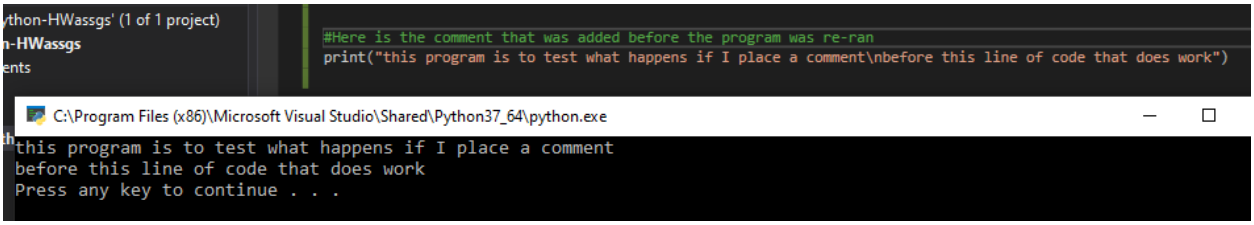
3. Place a comment before a line of code that previously worked, and record what happens when you rerun the program.

```
#3. Place a comment before a line of code that previously worked, and record what happens
when you rerun the
#program.

#Here is the comment that was added before the program was re-ran
print("this program is to test what happens if I place a comment\nbefore this line of
code that does work")
```

Before the comment is added:



After the comment is added:

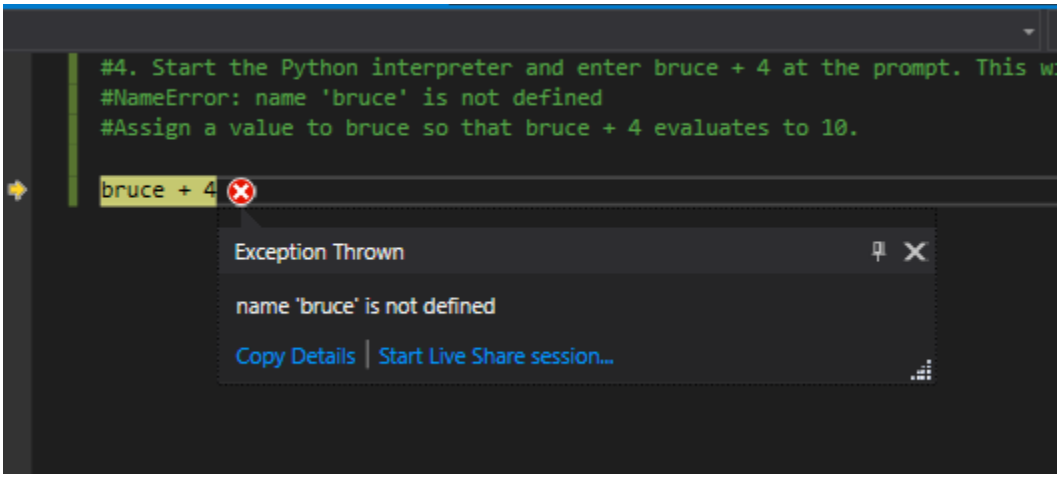


I notice there is no change in the output of the program whatsoever.

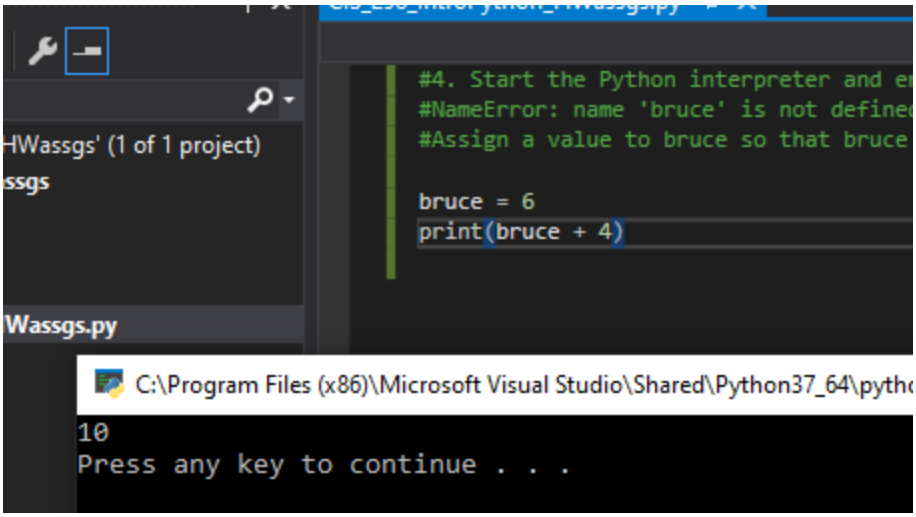
4. Start the Python interpreter and enter bruce + 4 at the prompt. This will give you an error:

NameError: name 'bruce' is not defined.

Assign a value to bruce so that bruce + 4 evaluates to 10.




```
#4. Start the Python interpreter and enter bruce + 4 at the prompt. This will give you an
error:
#NameError: name 'bruce' is not defined
#Assign a value to bruce so that bruce + 4 evaluates to 10.

bruce = 6
print(bruce + 4)
```

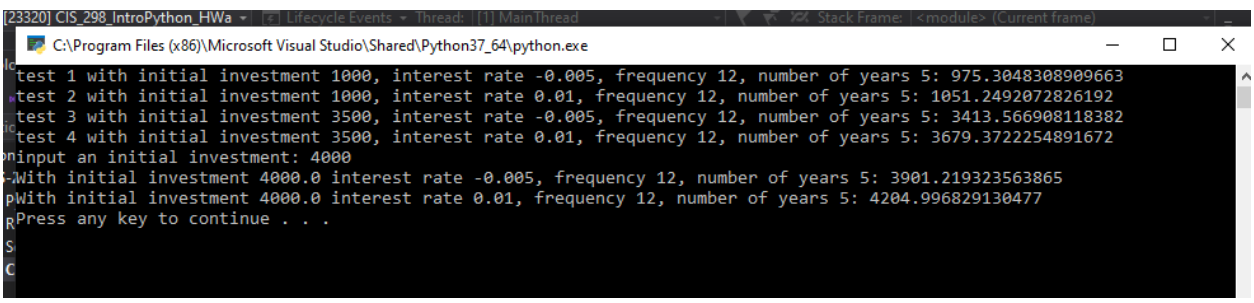


5. The formula for computing the final amount if one is earning compound interest is given on Wikipedia as A = P(1+ r/n)^(nt)

Here, P is the principal amount (the amount that the interest is provided on), n the frequency that the interest is paid out (per year), and r the interest rate. The number of years that the interest is calculated for is t. Write a program that replaces these letters with something a bit more human-readable, and calculate the interest for some varying amounts of money at realistic interest rates such as 1%, and also -0.05%. When you have that working, ask the user for the value of some of these variables and do the calculation.

```
#5. The formula for computing the final amount if one is earning compound interest is
given on Wikipedia as A = P(1+ r/n)^(nt)
#Here, P is the principal amount (the amount that the interest is provided on), n the
frequency that the interest
#is paid out (per year), and r the interest rate. The number of years that the interest
is calculated for is t. Write
#a program that replaces these letters with something a bit more human-readable, and
calculate the interest for
#some varying amounts of money at realistic interest rates such as 1%, and -0.05%. When
you have that working,
#ask the user for the value of some of these variables and do the calculation.



initial_investment_1 = 1000.00
initial_investment_2 = 3500.00
interest_rate_1 = -0.005
interest_rate_2 = 0.01
frequency_per_year = 12.00
num_years_t = 5.00

final_amount = initial_investment_1 * (1 + interest_rate_1 /
frequency_per_year)**(frequency_per_year * num_years_t)
print("test 1 with initial investment 1000, interest rate -0.005, frequency 12, number of
years 5:", final_amount)
```

```
final_amount = initial_investment_1 * (1 + interest_rate_2 /
frequency_per_year)**(frequency_per_year * num_years_t)
print("test 2 with initial investment 1000, interest rate 0.01, frequency 12, number of
years 5:", final_amount)

final_amount = initial_investment_2 * (1 + interest_rate_1 /
frequency_per_year)**(frequency_per_year * num_years_t)
print("test 3 with initial investment 3500, interest rate -0.005, frequency 12, number of
years 5:", final_amount)

final_amount = initial_investment_2 * (1 + interest_rate_2 /
frequency_per_year)**(frequency_per_year * num_years_t)
print("test 4 with initial investment 3500, interest rate 0.01, frequency 12, number of
years 5:", final_amount)

#now user will enter their initial investment and the total amount the gain/loss +
initial ivestment will be output
user_init_investment = float(input("input an initial investment: "))

final_amount = user_init_investment * (1 + interest_rate_1 /
frequency_per_year)**(frequency_per_year * num_years_t)
print("With initial investment", user_init_investment, "interest rate -0.005, frequency
12, number of years 5:", final_amount)

final_amount = user_init_investment * (1 + interest_rate_2 /
frequency_per_year)**(frequency_per_year * num_years_t)
print("With initial investment", user_init_investment, "interest rate 0.01, frequency 12,
number of years 5:", final_amount)
```



6. Evaluate the following numerical expressions in your head, then use the Python interpreter to check your results:

(a) >>> 5 % 2

(b) >>> 9 % 5

(c) >>> 15 % 12

(d) >>> 12 % 15

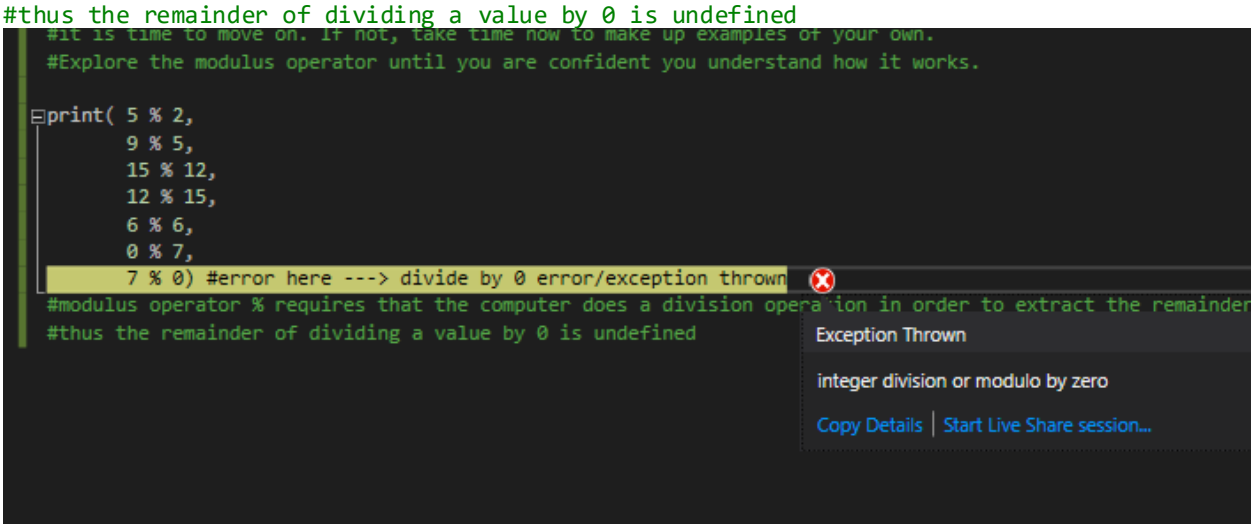
(e) >>> 6 % 6

(f) >>> 0 % 7

(g) >>> 7 % 0

What happened with the last example? Why? If you were able to correctly anticipate the computer's response in all but the last one, it is time to move on. If not, take time now to make up examples of your own. Explore the modulus operator until you are confident you understand how it works.

```
#6. Evaluate the following numerical expressions in your head, then use the Python
interpreter to check your results:
#(a) >>> 5 % 2
#(b) >>> 9 % 5
#(c) >>> 15 % 12
#(d) >>> 12 % 15
#(e) >>> 6 % 6
#(f) >>> 0 % 7
#(g) >>> 7 % 0
#What happened with the last example? Why? If you were able to correctly anticipate the
computer's response in all but the last one,
#it is time to move on. If not, take time now to make up examples of your own.
#Explore the modulus operator until you are confident you understand how it works.

print( 5 % 2,
       9 % 5,
       15 % 12,
       12 % 15,
       6 % 6,
       0 % 7,
       7 % 0) #error here ---> divide by 0 error/exception thrown
#modulus operator % requires that the computer does a division operation in order to
extract the remainder,
#thus the remainder of dividing a value by 0 is undefined
```



Now notice if I change 7%0 to 7%1, the program will run and output the remainder of each division operation:

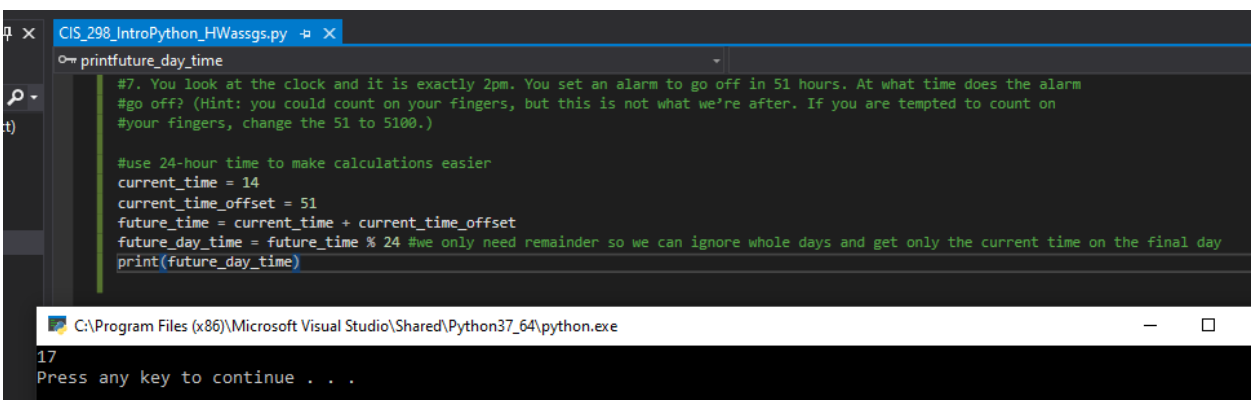
```
print( 5 % 2,
       9 % 5,
       15 % 12,
       12 % 15,
       6 % 6,
       0 % 7,
       7 % 1) #error here ---> divide by 0 error/e
#modulus operator % requires that the computer doe
#thus the remainder of dividing a value by 0 is ur
```

C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python

```
1 4 3 12 0 0 0
Press any key to continue . . .
```

7. You look at the clock and it is exactly 2pm. You set an alarm to go off in 51 hours. At what time does the alarm go off? (Hint: you could count on your fingers, but this is not what we're after. If you are tempted to count on your fingers, change the 51 to 5100.)

```
#7. You look at the clock and it is exactly 2pm. You set an alarm to go off in 51 hours.
At what time does the alarm
#go off? (Hint: you could count on your fingers, but this is not what we're after. If you
are tempted to count on
#your fingers, change the 51 to 5100.)

#use 24-hour time to make calculations easier
current_time = 14
current_time_offset = 51
future_time = current_time + current_time_offset
future_day_time = future_time % 24 #we only need remainder so we can ignore whole days
and get only the current time on the final day
print(future_day_time)
```



Thus, based on my program output, it will be 1700 = 5pm when the alarm goes off (on the 3rd day after the current 2pm time).

8. Write a Python program to solve the general version of the above problem. Ask the user for the time now (in hours), and ask for the number of hours to wait. Your program should output what the time will be on the clock when the alarm goes off.

```
#8. Write a Python program to solve the general version of the above problem.
#Ask the user for the time now (in hours), and ask for the number of hours to wait.
#Your program should output what the time will be on the clock when the alarm goes off.

#use 24-hour time to make calculations easier

current_time = int(input("Enter current time of day in 24-hour format - use nearest hour: "))
current_time_offset = int(input("Enter the time to wait - in hours: "))
future_time = current_time + current_time_offset
future_day_time = future_time % 24 #we only need remainder so we can ignore whole days and get only the current time on the final day
print("The clock will read this value when the alarm goes off: ", future_day_time)
```