

Problem Set Two

Learning Goals

- 'for' loops • variable assignment • keyboard input
- flow of control between functions in Python • Basic HTML

Timeline

<u>Part</u>	<u>Due Date</u>	<u>Note</u>
Part A	Saturday, July 1, 3 PM	
Part B	Saturday, July 1, 3 PM	.py files
Part C	Tuesday, July 5, Noon	Includes "graduate credit" and optional problems for "extra credit"

If you submit any part of the assignment up to 24 hours late, you will receive a 10% deduction on that part. Any submission after 24 hours will not be accepted.

Deliverables via Gradescope

<u>Part A</u>	<u>Part B</u>	<u>Part C</u>
Problems 1 through 7 should be answered directly on <i>Gradescope</i>	One .zip file containing: <ul style="list-style-type: none"> • temperature.py • diamond.py • oops.py • kernighan.py • easter.py (for grad-credit students) 	One .zip file containing all files for the solutions attempted <ul style="list-style-type: none"> • donor.py • batman.py • right_triangle.py • drawing.py • triangle.py • primes.py • song.py

Suggested Reading

Chapters 1 & 2 in Reges & Stepp

Collaboration Policy Reminder

Warning: PLEASE DON'T COPY OTHER PEOPLE'S WORK! It is fine to discuss algorithmic approaches with fellow students and talk about how to solve a problem. It is not OK to look at someone else's solution before writing or modifying yours.


Depending on the severity, solutions that are inappropriately similar to other students' homework may have their scores divided in two, or may receive a zero score, or may cause the student to be referred to the Summer School administrative board for disciplinary action, which could result in a requirement to withdraw from the Harvard Summer School.

YOU MUST ELECTRONICALLY SUBMIT YOUR SOLUTIONS USING Gradescope, available via our Canvas course website. Merely saving your files in your online account is not enough. Place your answers into separate files using the names indicated on each problem. **You will lose at least one point** for each problem where you submit an incorrectly named file, including capitalization.

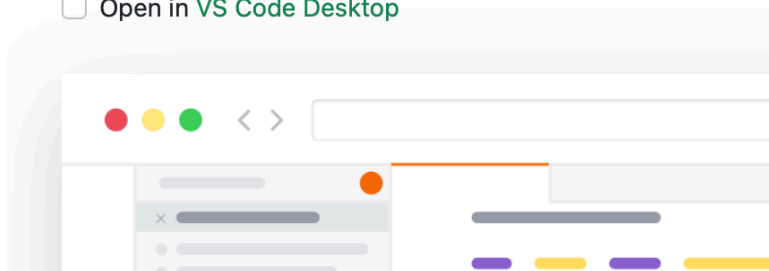
Before you do anything else, make sure you have created an account for yourself on the website <http://cs50.dev> and on <http://github.com>. It is essential to use the *Google Chrome* browser. The cs50.dev website will present a page that looks like this:

Visual Studio Code for C

CS50's adaptation of **Codespaces** for student

 Log in via GitHub or browse **documentation**

☐ Open in VS Code Desktop



Notice at the top of the main page, there are 2 links. One says: Browse the [browse the documentation](#) while the other one says to *Log in via GitHub*.

If you click the *Log in via GitHub* link, you will be taken to a page that allows you to create a free account on Github, which you should do!

You will receive an email authorization to complete the creation of your *Github* account. After this has happened, you will be able to log in to [cs50.dev](#) very easily.

Part A — due prior to 3 PM on Saturday, July 1

Pencil-and-Paper Exercises (30 points total)

You need not run *Python* programs on a computer in solving the following problems. Answer all questions in this part directly on *Gradescope*!

[1] 3 points

What is the precise output from the following code?

```
bar = 13
foo = 23 - 45 // 10
foo += 6
bar = foo % bar
print (foo * 2)
print (foo + bar)
print (foo, bar)
```

[2] 3 points

Assume that you have a variable named **count** that will take on the values 1, 2, 3, 4, and so on. You are going to formulate a *Python* expression in terms of **count** that will yield a particular sequence. For example, to get the sequence 2, 4, 6, 8, 10, 12, ..., you would use the expression $(2 * \text{count})$.

What *Python* expression would yield the sequence 4, 19, 34, 49, 64, 79, ... ?

[3] 3 points

Precisely what does this program print when executed?

```
def main():
    print ("She said, \"", end="")
    print ("Never put off till tomorrow... ", end="")
    print ("\nWhat you can do")
    print ("The day", end="")
    print ()
    print ("After tomorrow!\n")

main()
```

[4] 3 points

Precisely what are the values of **first** and **second** after the following code executes? In one or two unambiguous English sentences, how would you describe the overall effect of the statements in this exercise?

```
first = 18
second = -9
first += second
second = first - second
first = first - second
```

[5] 7 points total

Part (a): 4 points

Precisely what does the following program output when we begin by executing `main()` ?

```
def first():
    print ("Inside first function!")
```

```
def second():
    first()
    print ("Inside second function!")

def third():
    print ("Inside third function!")
    first()
    second()

def main():
    first()
    third()
    second()
    third()
```

Part (b): 3 points

What would have been the output of the preceding program, starting with **main()** if the function **third** had instead been written like this?

```
def third():
    first()
    print("Inside third function!")
    second()
```

[6] 5 points

Part (a): 2 points

Complete the following *Python* code. You may modify only the indicated comment.

```
for j in range (5) :
    # your code goes right here and nowhere else!
```

so that this loop will print the following numbers, one per line:

-2
1
4
7
10

Part (b): 3 points

Produce this same sequence in a different way, replacing the question marks ONLY in the following code to output the same thing as in part (a):

```
for j in range ( ??? ) :  
    print (j)
```

[7] 6 points (2 points for each)

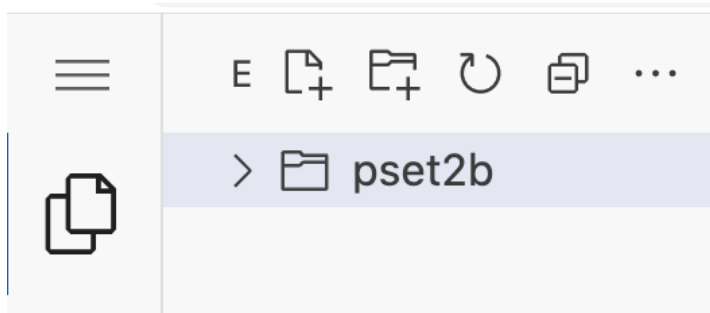
Precisely how many values do the each of the following loops print? (In other words, how many times does each loop iterate?)

- (a) **for j in range (3000) :**
 print(j)
- (b) **for j in range (-5, 5) :**
 print(j)
- (c) **for j in range (50, -10, -8) :**
 print(j)

Part B — due prior to 3 PM on Saturday, July 1

Sign on to [cs50.dev](#), and then create a new folder named **pset2b** by typing the command **mkdir pset2b** in the lower window:

The command you just typed created a “subdirectory” (or folder) in your home folder — this is where you will create files for Part B of this problem set. The lefthand side of your browser window should now look like this:



Now solve the following problems by creating files in the **pset2b** folder. You should issue the command

cd pset2b

in the lower window to be sure everything you work on at this point will take place inside the

pset2b folder.

IMPORTANT -- if [cs50.dev](#) goes offline or has technical problems that hinder your ability to do this assignment, you should instead use [ide.cs50.io](#) ... it is very similar to the [cs50.dev](#) environment and will also utilize your *Github* account.

Programming Problems (38 to 50 points total)

Solve the following problems using **Python** within your [cs50.dev](#) account. Be sure you use the precise file names shown when you write your solutions! User input is underlined in a **bold** typeface.

You should begin by first created a folder named **pset2b** in the same manner that you created **pset2a**. Then issue the command **cd pset2b** so that all your work will occur in this new folder.

Work carefully! YOUR PYTHON PROGRAMS MUST RUN. Any *Python* programs submitted that do not execute because they have syntax errors will automatically have their scores divided in two.

[8] **13 points**

Use file *temperature.py*

To convert a temperature from Fahrenheit degrees to Kelvin (absolute), the following relationship holds:

$$K = \frac{5}{9} (^\circ F - 32) + 273.16$$

Write a *Python* program that prints the Kelvin equivalent of any Fahrenheit temperature, based upon the value of a numeric variable named **fahrenheit** that is input from the keyboard using the `input()` function. Here is what your program might look like in action:

Input a Fahrenheit temperature to be converted: 212.0
212.0 degrees Fahrenheit equals 373.16 degrees Kelvin

[9] **7 points**

Use file *diamond.py*

Write a simple *Python* program using **print** statements (no loops, no conditional statements) to produce the following output:

```
$ python diamond.py
D
 I I
A   A
M   M
 O O
 N N
  D
 $ █
```

Note: there is NO space character in front of the first **M** character.

[10] 8 points**Use file *oops.py***

Carefully type the following program into a file named **oops.py**

```
def main():  
    studs = input("How many students are enrolled? ")  
    Print ("We need, ' studs%15, 'teaching assistants')  
  
main()
```

If you attempt to run this program using *python*, you will discover several errors that prevent it from executing. There are 4 distinct mistakes that need to be corrected. Once the program has been completely debugged, it should look like this in action:

```
~/ $ python oops.py  
How many students are enrolled? 152  
We need 10 teaching assistants
```

Fix this program so that it will run without error.

Note: we are trying to calculate one TF for every 15 students, rounding down.

[11] 10 points**Use file *evens.py***

Write a *Python* program that outputs the sum of all even integers from 2 up through **n** (where **n** is an integer input by the user). For example,

```
~/ python evens.py  
I will compute the sum of even integers from 2 through? 50  
The sum is 650
```

Your solution must use a **for** loop.

[12] 12 points: For Graduate Credit students Use file *easter.py*

This problem may also be solved for "extra credit" by undergraduate credit students. It's required for graduate credit students.

Write a program to compute the date of Easter Sunday. Easter Sunday is the first Sunday after the first full moon of Spring. Use the following algorithm, invented by the mathematician Carl Friedrich Gauss in 1800:

1. Let **y** be the year (such as 1800 or 2001 or whatever)
2. Divide **y** by **19** and call the remainder **a**. Ignore the quotient.
3. Divide **y** by **100** to get a quotient **b** and a remainder **c**.
4. Divide **b** by **4** to get a quotient **d** and a remainder **e**.
5. Divide **8 * b + 13** by the value **25**, to get a quotient **g**. Ignore the remainder.
6. Divide **19 * a + b - d - g + 15** by the value **30**, to get a remainder **h**. Ignore the quotient.
7. Divide **c** by **4** to get a quotient **j**, and a remainder **k**.
8. Divide **(a + 11*h)** by the value **319**, to get a quotient **m**. Ignore the remainder.
9. Divide **2 * e + 2 * j - k - h + m + 32** by the value **7** to get a remainder **r**. Ignore the quotient.
10. Divide **h - m + r + 90** by the value **25**, to get a quotient **n**. Ignore the remainder.
11. Divide **h - m + r + n + 19** by the value **32**, to get a remainder **p**. Ignore the quotient.

Then Easter falls on day **p** of month **n**.

For example, if **y** is the year **2001**, we would compute

```
a = 6
b = 20
c = 1
d = 5, e = 0
```

```

g = 6
h = 18
j = 0, k = 1
m = 0
r = 6
n = 4
p = 15

```

Therefore, in 2001, Easter Sunday fell on **4/15**. Write a complete program that allows the user to input a year from the keyboard, and have the program output the month number, followed by a slash, followed by the day number. For example,

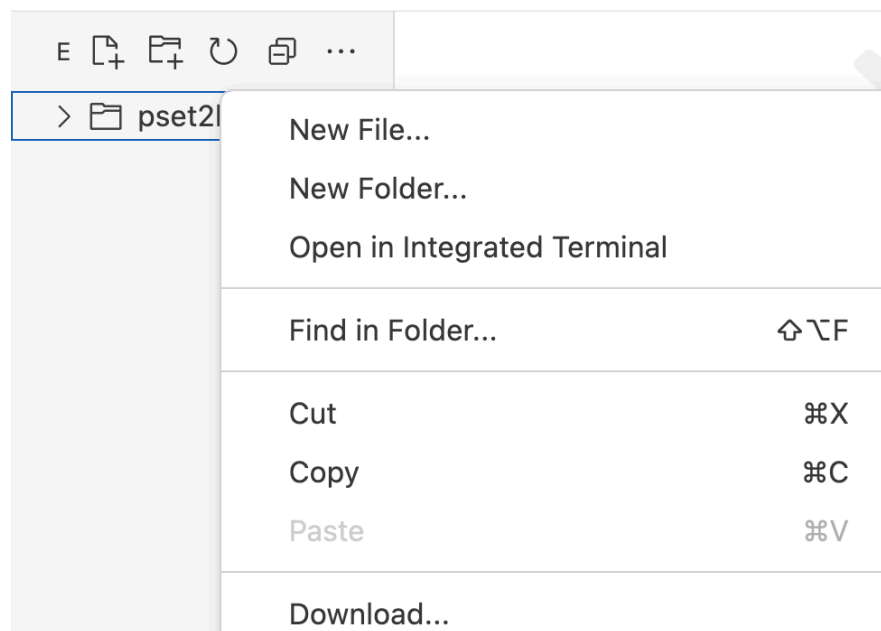
```

$ python easter.py
Enter a year: 2022
4/17
$ 

```

Submitting Part B

First download your files to your personal computer by right-clicking (on a Mac, *control-clicking*) the **pset2b** folder and selecting **Download...** from the menu as shown here. Be careful to not download the entire *workspace* by clicking on an enclosing folder.



Your browser will download a **.zip** file. Next, you will **upload** this file on the Canvas course website via *Gradescope* as your solution to **PSet2, Part B**. If *Gradescope* says the **.zip** file is too big, it means you downloaded the whole workspace by mistake.

Note that once you have downloaded the **.zip** file, you can double-click on this file to extract and verify it contains the files you want to submit. On a *Windows* machine, you might need to install the 7-zip utility [http:// www.7-zip.org/](http://www.7-zip.org/)

Part C — due prior to Noon on Tuesday, July 4

The same way you created a **pset2b** folder, you will now create a new folder named **pset2c** by right-clicking or control-clicking on a Mac and selecting *new folder*, or using the **mkdir** command. Then proceed to problem #13 below! All of the following require using *Python* with your *code.cs50.io* account. All your solutions to Part C should reside in the **pset2c** folder.

[13] 14 points

Use file *donor.py*

A certain charity designates donors who give \$10,000.00 or more as *Benefactors*; those who give \$1,000.00 or more but less than \$10,000.00 as *Patrons*; those who give \$200.00 or more but less than \$1,000.00 as *Supporters*; those who give \$15.00 but less than \$200.00 as *Friends*; and those who give less than \$15 as *Cheapskates*. \$0 would be considered a *Cheapskate* amount.

Write a function **donor()** containing a nested **if-else** statement (or statements) that, given a keyboard input with the amount of a contribution, **returns** the correct designation for that contributor.

Here's an example to illustrate what your program should look like in action:

```
~/ python donor.py
```

```
Enter the amount of a contribution: $5.20
```

```
Cheapskate!
```

Test your program also for the input values: **10000.00, 1000.00, 999.99, 12200.00, 15.00, 499.99, 2200.00, 199.99, 75.33**

An appropriate error message should be output (and your program should halt) when an amount less than 0 is input.

[14] 8 points

Use file *batman.py*

Write a *Python* program containing a **for** loop to precisely print the message

Na na na na na na na na na na na na na na na na ... BATMAN!

[15] 12 points

Use file *right_triangle.py*

Write a complete *Python* program to output the following pattern that resembles a right triangle with the base on top:

```

* * * * *
  * * * *
    * * *
      * *
        *

```

Your program should begin by asking the user how tall to make the output (5 in the above figure). If the user instead input **7** instead of **5**, the output would look like this:

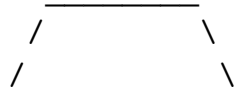
```

* * * * *
 * * * * *
  * * * *
   * * * *
    * * *
     * *
      *

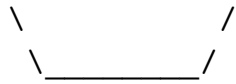
```

[16] 15 points**Use file** *drawing.py*

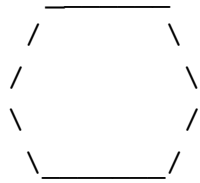
Write a function named **top()** that draws this figure (there are precisely 7 **underscore** characters in the top line):



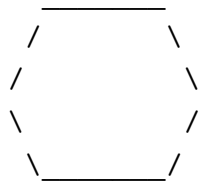
Now write a function named **bottom()** that draws this figure:



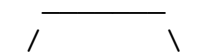
Finally, define a **main** function that uses the **top()** and **bottom()** functions (along with any additional functions you might like) to output the following. Note that the line containing quotation marks is using hyphen characters, not underscores.



- " - ' - " - ' - " -



- " - ' - " - ' - " -



```

/      \
- " - ' - " - ' - " -
\      /
 \_____/

```

Grad-Credit and Extra Credit Problems (up to 20 points possible)

Graduate-credit students must answer at least one of the following three problems (and may answer one other for additional extra credit). Undergraduate-credit students may answer no more than two of the following for additional extra credit.

[17] **10 points**

Use file *triangle.py*

Write a *Python* program that produces the output

```

100
200    202
300    304    308
400    406    412    418
...
...
...
900    916    932    ...    1028

```

Your program must not, of course, consist entirely of **print()** statements! Use **for** loops.

[18] **10 points**

Use file *perfect.py*

Write a *Python* program that prompts the user for an integer and then prints out all the perfect numbers up to that integer. For example, if the user enters 500, the program should print

```

6
28

```

496

An integer is *perfect* if it is equal to the sum of its divisors, except for itself. For example, 28 is perfect because it is equal to the sum of $1+2+14+4+7$

[19] 14 points**Use file *song.py***

Write a *Python* program that produces as output the lyrics of the song, “There Was an Old Lady.” Use functions for each verse and the refrain. Here are the song’s the complete lyrics:

**There was an old lady who swallowed a fly
I don’t know why she swallowed that fly
Perhaps she’ll die**

**There was an old lady who swallowed a spider
That wriggled and jiggled and wiggled inside her
She swallowed the spider to catch the fly
I don’t know why she swallowed that fly
Perhaps she’ll die**

**There was an old lady who swallowed a bird
How absurd, to swallow a bird
She swallowed the bird to catch the spider
That wriggled and jiggled and wiggled inside her
She swallowed the spider to catch the fly
I don’t know why she swallowed that fly
Perhaps she’ll die.**

**There was an old lady who swallowed a cat
Imagine that, she swallowed a cat
She swallowed the cat to catch the bird
She swallowed the bird to catch the spider
That wriggled and jiggled and wiggled inside her
She swallowed the spider to catch the fly
I don’t know why she swallowed that fly
Perhaps she’ll die**

**There was an old lady who swallowed a dog
What a hog! To swallow a dog**

She swallowed the dog to catch the cat
She swallowed the cat to catch the bird
She swallowed the bird to catch the spider
That wriggled and jiggled and wiggled inside her
She swallowed the spider to catch the fly
I don't know why she swallowed that fly
Perhaps she'll die

There was an old lady who swallowed a goat
Just opened her throat and swallowed a goat
She swallowed the goat to catch the dog
She swallowed the dog to catch the cat
She swallowed the cat to catch the bird
She swallowed the bird to catch the spider
That wriggled and jiggled and wiggled inside her
She swallowed the spider to catch the fly
I don't know why she swallowed that fly
Perhaps she'll die

There was an old lady who swallowed a cow
I don't know how she swallowed a cow
She swallowed the cow to catch the goat
She swallowed the goat to catch the dog
She swallowed the dog to catch the cat
She swallowed the cat to catch the bird
She swallowed the bird to catch the spider
That wriggled and jiggled and wiggled inside her
She swallowed the spider to catch the fly
I don't know why she swallowed that fly
Perhaps she'll die

There was an old lady who swallowed a horse
She died, of course.

Submitting Part C

Submit part C the same way you submitted part B. First download your homework by right-clicking (or control-clicking) on your **pset2c** folder and selecting **download**.

Next, upload this file on the Canvas course website using *Gradescope* for **Pset2, Part C**

All Done!