


Unit 7: Game Programming in Python

Lesson 1: Intro to PyCharm & PyGame

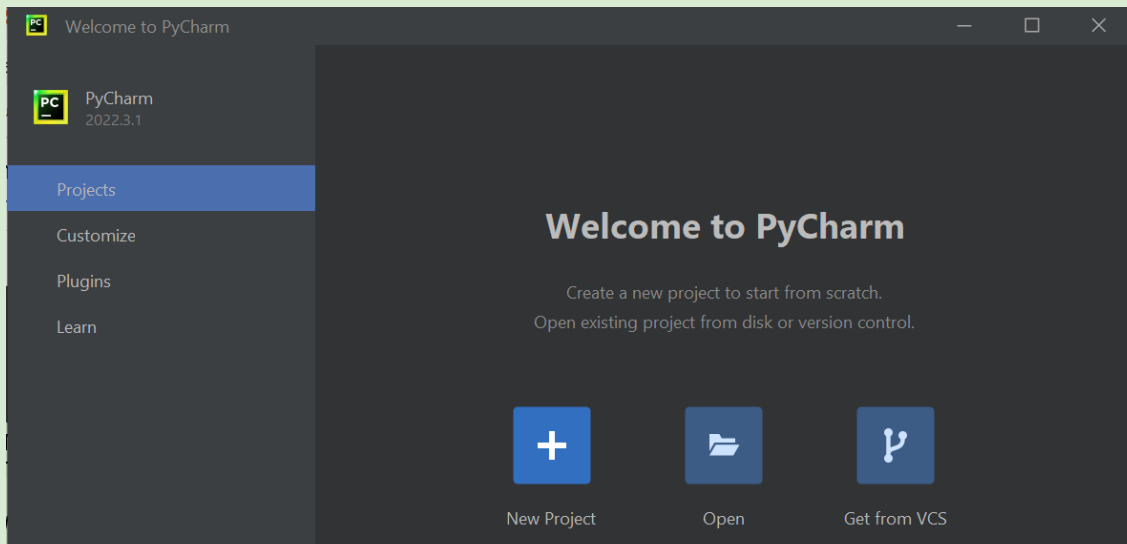
Name: _____

Creating a Project in PyCharm

1. Open up **PyCharm** on your desktop:  You may need to click "Agree" and go through a few other dialogues. After that, you want to **create a new PyCharm python project**. *But depending on your computer, you will see one of three different windows when you start PyCharm.*

Here is what to do next in each case:

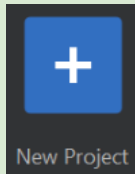
If you see a window like this:

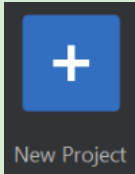


This window means:

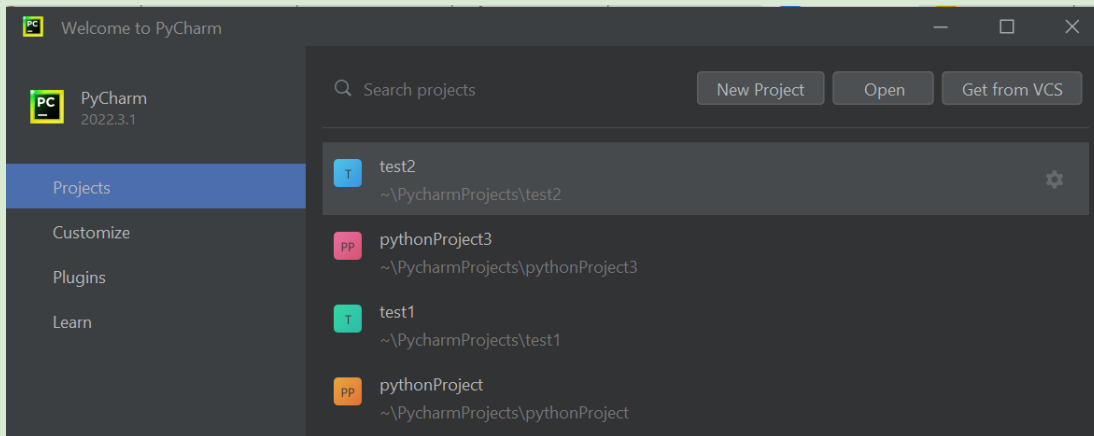
You are the first person to create a PyCharm project on the computer, or there are no projects currently saved on the computer. This is what you will see if your machine was rebooted and all user files deleted.

What to do to create a new project:



Click  to create a new project.

If you see a window like this:



This window means:

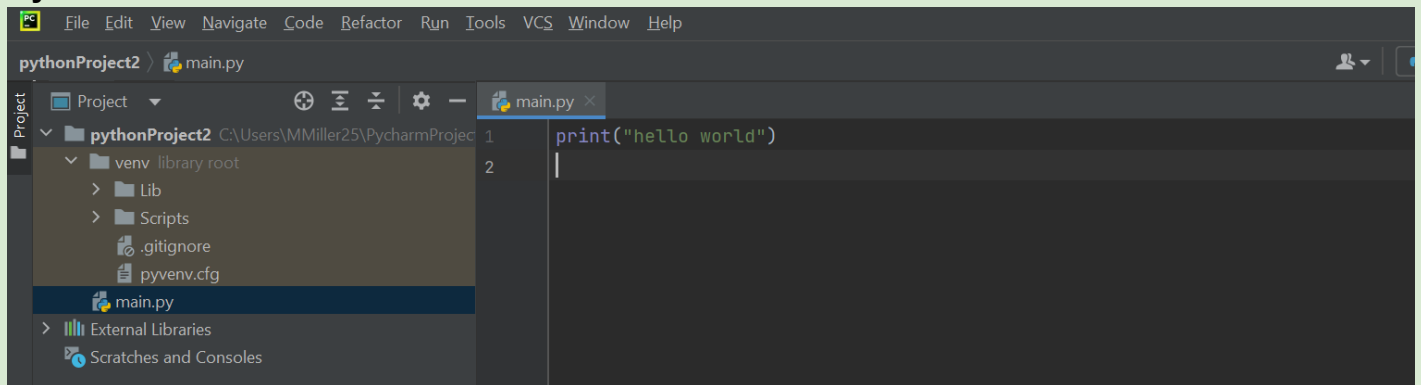
There are already PyCharm projects saved on the computer (i.e. the computer hasn't been restarted and there are projects still on the computer from previous classes)

What to do to create a new project:

New Project

Click to create a new project.

If you see a screen like this:

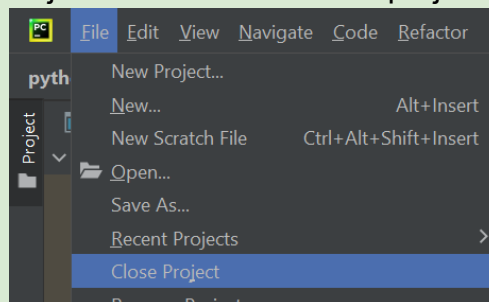


This means:

Someone used PyCharm before you on the computer and forgot to close their last project and PyCharm automatically loaded their project.

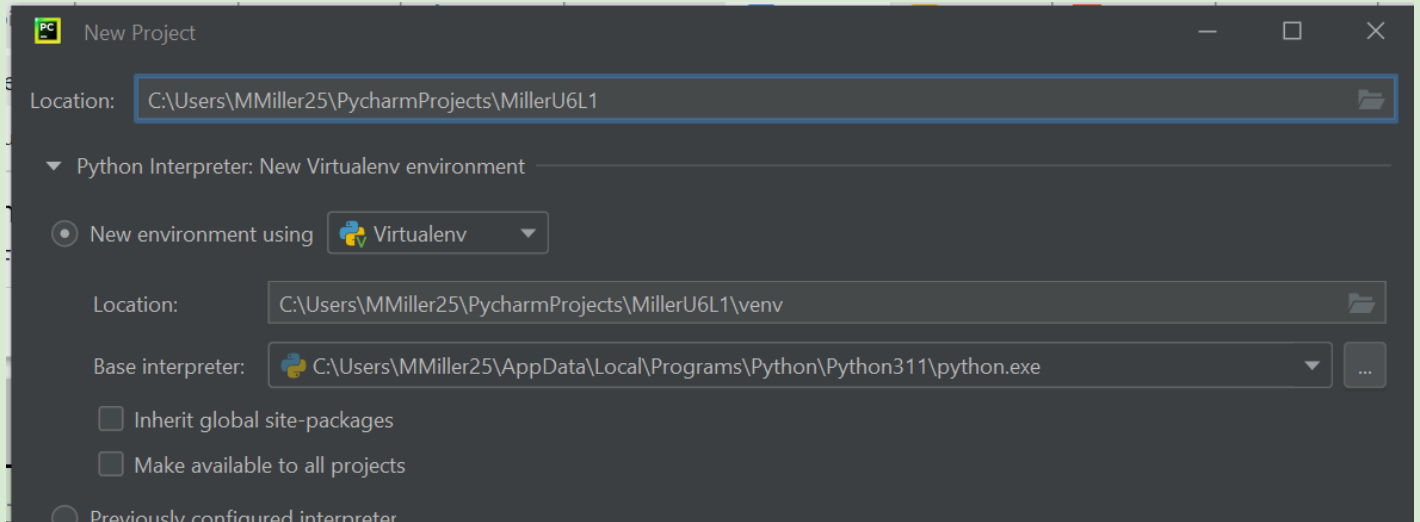
What to do to create a new project:

A. First, go to File → Close Project to close the current project:

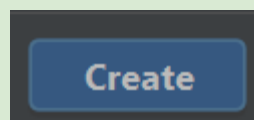
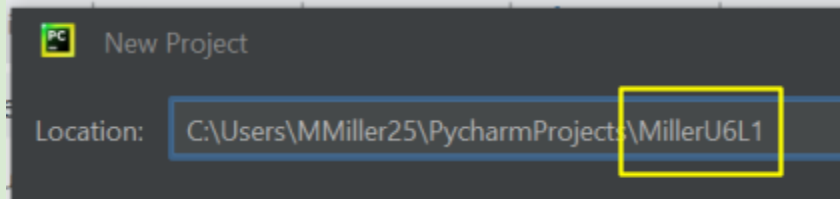


B. Then, it will take you back to one of the two windows above; follow the appropriate instructions to create a new project.

2. You should see a window like this:

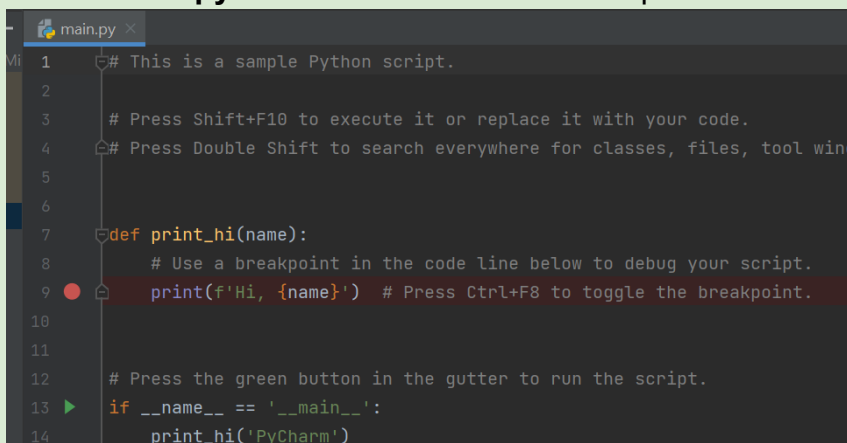


In the "**Location**", edit the part following the last "\" to be your **LAST NAME** followed by the **lab**:

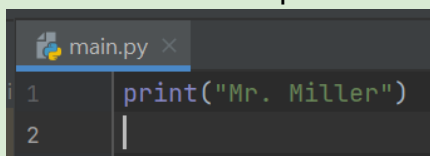



Leave all other settings their default values, and click:

3. Your **main.py** file will have a bunch of sample code in it:

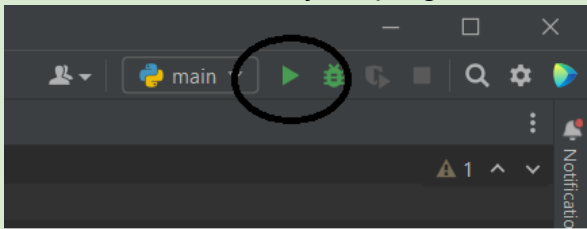


Delete it all then replace it with a simple print statement to print your name.



Click on the  button on the left to remove a default "breakpoint" (we won't be using those now).

4. Run your program to make sure everything works! At the top right, you should see a **Green Arrow**. Click it to run your program.



5. The output of your program will be at the bottom of the window.

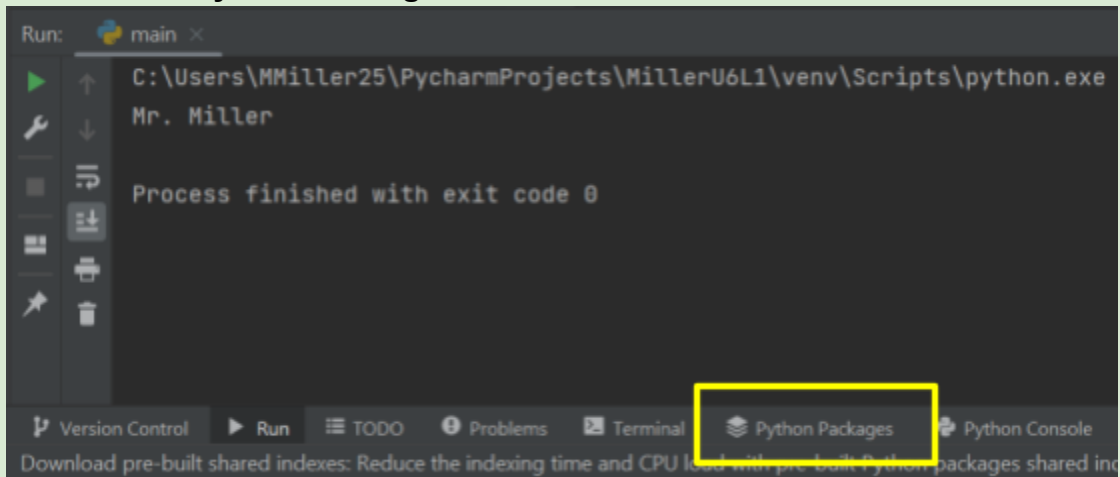
A screenshot of the PyCharm console window. The title bar says 'main'. The console output shows the file path 'C:\Users\MMiller25\PycharmProjects', the name 'Mr. Miller', and the message 'Process finished with exit code 0'.

Take a screenshot of your program printing your name:

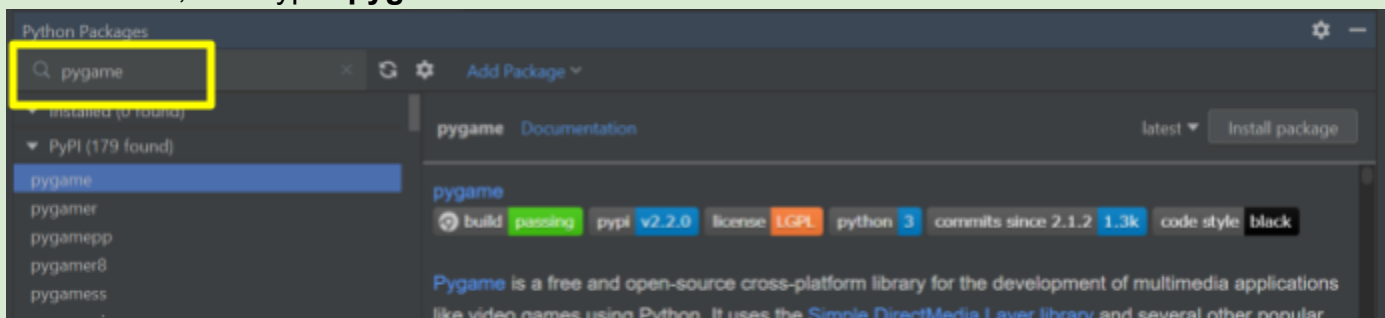
Next Up: Getting started with PyGame!

Installing PyGame

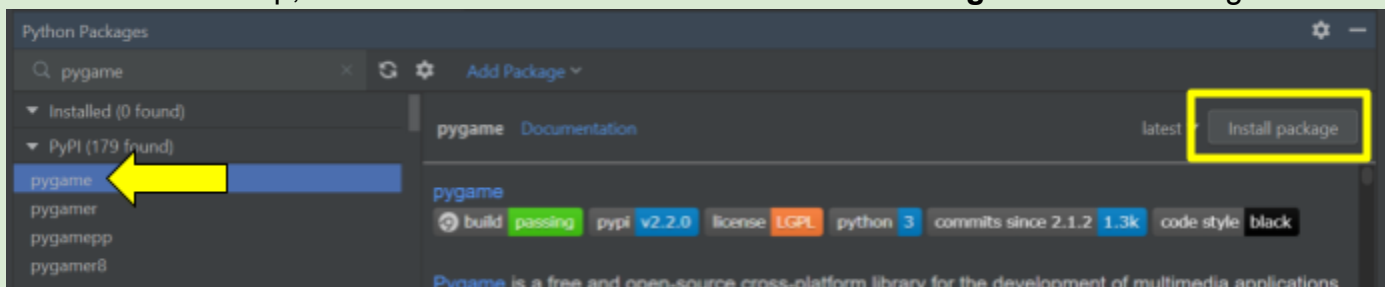
6. For each project, you'll have to install the **pygame** library/module/package. Go to the bottom of the screen and click on **Python Packages**.



7. Click that, then type "pygame" in the search field:



8. Once it shows up, select it on the left then click the **Install Package** button on the right:

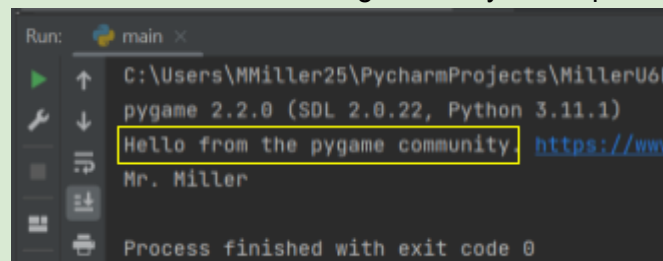
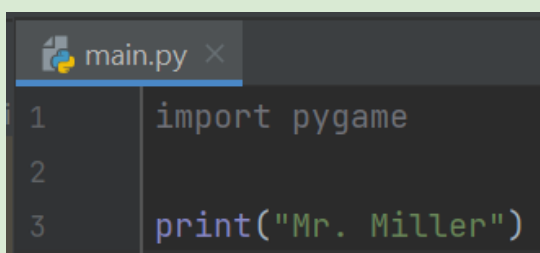


Package pygame installed

Give it 10 seconds or so, and you should see a message like this:

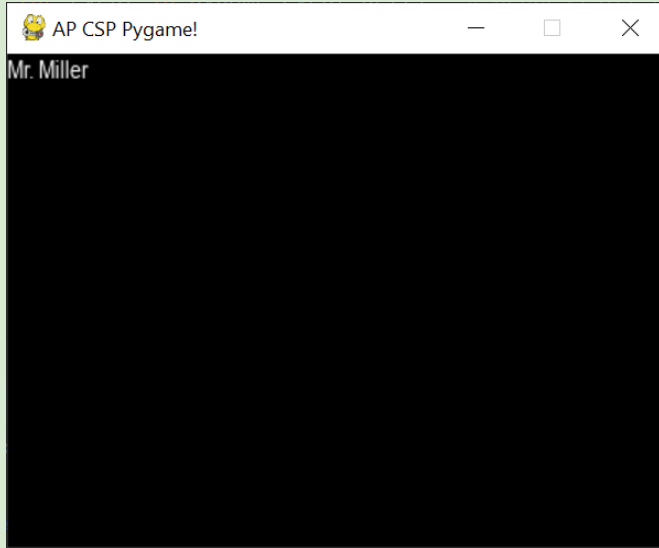
9. **Test that it worked** by adding "import pygame"

at the top of your main.py file and **running again** → You should see this message above your output:



Creating a Window and Drawing Text

Our goal today is to create a basic Window using pygame, draw some text on it and experiment with **Fonts**, **Background Colors** and **Text Positioning**. Here is an example of what this example will walk you through:



1. In your PyCharm project, **remove** the initial print statement that you added previously to print your name. Then, **add the following lines of code (copy/paste)**; this is "cookie cutter" code that you will use to start most of your pygame projects:

```
import pygame

# set up pygame modules
pygame.init()
pygame.font.init()
font = pygame.font.SysFont('Arial', 15)
pygame.display.set_caption("AP CSP Pygame!")

# set up variables for the display
size = (400, 300)
screen = pygame.display.set_mode(size)
```

Every pygame project you will write should start like this. The only thing that may change is the **Font** and **Font Size** you want to use, along with the size of the screen.

2. Rectangles, Rectangles, Rectangles: *Everything* that gets drawn to a window in pygame is represented by a **Rectangle** object (even text). **Note** that the Rectangle class used to create these objects is defined somewhere in the PyGame library; it is **NOT** the same Rectangle class we wrote ourselves in earlier labs.

So, to create a text and have it display at the top right of the screen, we first need to "render" a **Rectangle** object; **copy/paste the additional line of code below** and change it to your name:

```
import pygame

# set up pygame modules
pygame.init()
```

```

pygame.font.init()
font = pygame.font.SysFont('Arial', 15)
pygame.display.set_caption("AP CSP Pygame!")

# set up variables for the display
size = (400, 300)
screen = pygame.display.set_mode(size)
display_name = font.render("Mr. Miller", True, (255, 255, 255))

```

`display_name` represents a **Rectangle** object that is the result of calling the `render` method.

The `render` method takes in three arguments:

1. The **String** you want to have drawn on the screen (in this case, "Mr. Miller")
2. A boolean value that allows you to **smooth** out the font so it looks nicer (we should really always set this to `True`).
3. A **tuple** that represents the RGB value for the text (in this case, we are making it white).

3. Program Loops! Every pygame program will have a **main program loop** and inside that loop, a *nested event-listening loop*. We will talk more about the event loop in a later lesson, but for now, **copy/paste the additional code below:**

```

import pygame

# set up pygame modules
pygame.init()
pygame.font.init()
font = pygame.font.SysFont('Arial', 15)
pygame.display.set_caption("AP CSP Pygame!")

# set up variables for the display
size = (400, 300)
screen = pygame.display.set_mode(size)
display_name = font.render("Mr. Miller", True, (255, 255, 255))

# main "program loop"
# keeps the window open via the "update" method being called
# over and over again
run = True
while run:
    screen.fill((0, 0, 0))
    screen.blit(display_name, (0, 0))

    # "event listening" loop
    for event in pygame.event.get(): # process all user events
        if event.type == pygame.QUIT: # check if user clicked close
            run = False
    pygame.display.update()

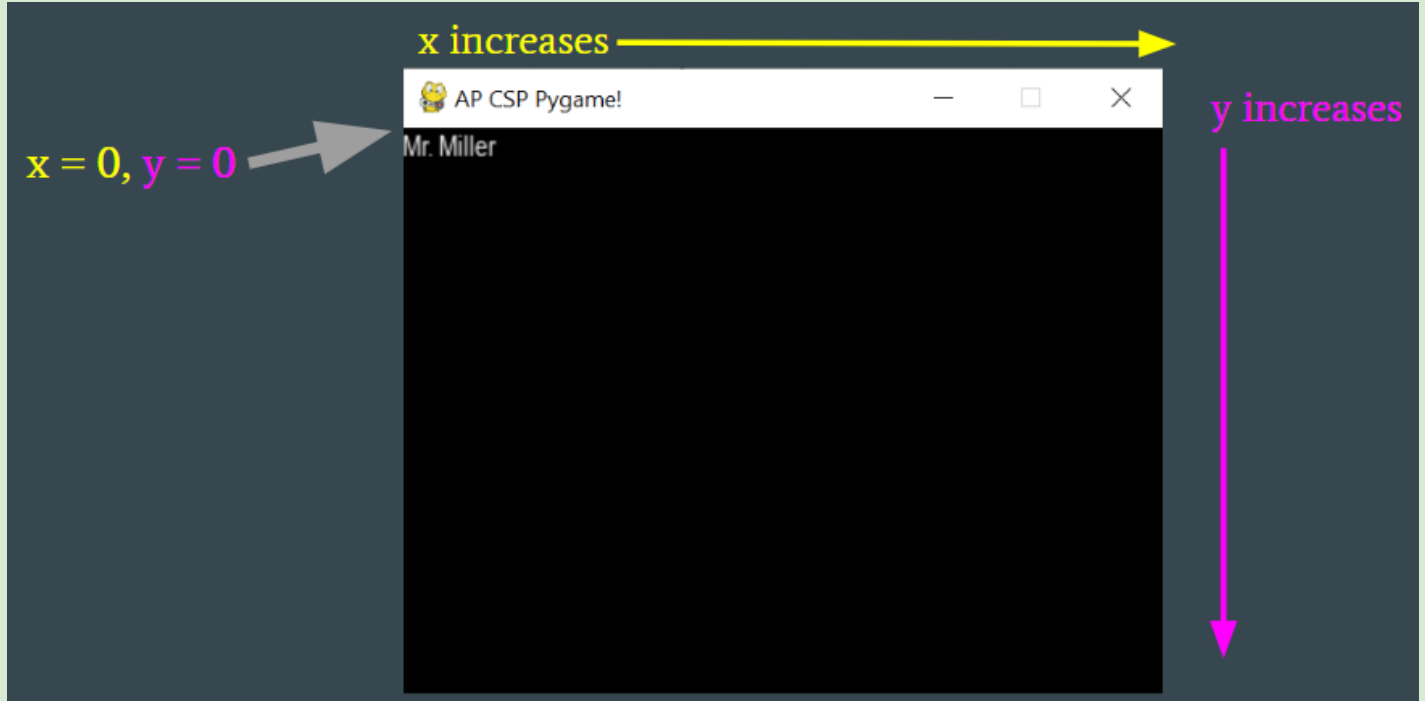
```

4. Understanding `screen.fill` and `screen.blit`

`screen` is created earlier in the program and refers to the main window of our program.

The `fill` method allows us to set a background color for the window using an RGB tuple (here it's black).

The `blit` function allows us to place **Rectangle objects** that we've previously created onto our main surface. Here, we are placing the `display_name` object in the position (`x = 0`, `y = 0`). Unlike in math, where we increase `x` right and `y` *up*, in pygame, `x` increases right and `y` increases *DOWN*, which means **(0, 0) is the top left**, *not* the bottom left:



`pygame.display.update()` does a **full refresh** of the screen based on what you placed on it with `fill` and `blit`.

Insert a screenshot of a pygame window showing your name printed in the top left:

Lab continues on the next page

Adding more to the window!

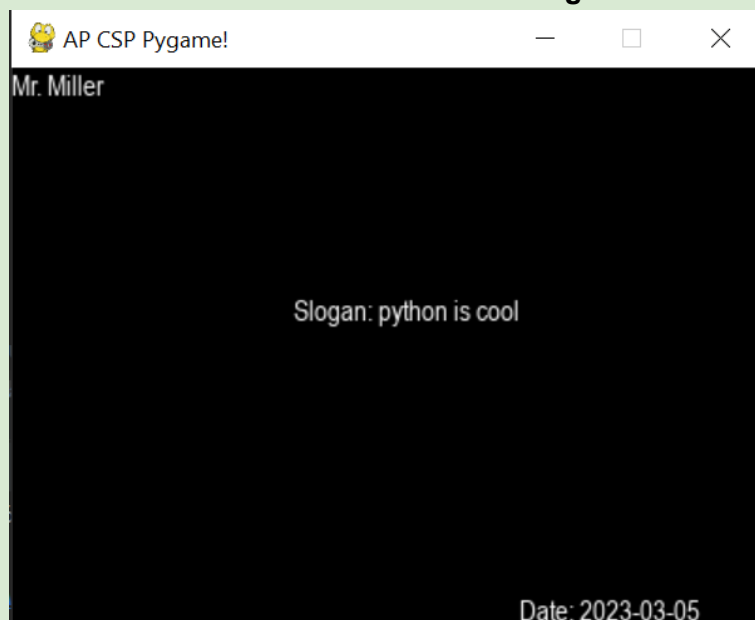
Add a "**slogan**" and today's **date** to the window and position them appropriately.

- Your name should already be at the top left.
- Add a slogan and position it in the **center**
- Add today's date and position it in the **bottom right**.

You can get the current date in Python like this:

```
from datetime import date  
  
current_date = "Date: " + str(date.today())
```

Your final window should look something like this:



(note: yours should show *today's* date, *not* 03-05)

Insert a screenshot of your window showing your name, slogan, and today's date:

Copy and paste your main.py code here that produces your window above:

[Sample solution](#)

FREE STYLE!

Try experimenting with different background colors, font colors, fonts, and font sizes!

1. Change the font and font size to something of your liking in this line of code:

```
my_font = pygame.font.SysFont('Arial', 15)
```

PRO TIP: You can *print to the console* all possible fonts by running this line of code:

```
print(pygame.font.get_fonts())
```

2. Make all the texts and the background different colors (i.e. different RGB tuples).

3. Add another message of your choice somewhere else on the screen.

Challenge!

Can you figure out how to draw all the different texts using *different* fonts?

Insert a screenshot of your window showing your freestyle changes:

Copy and paste your main.py code here that produces your window above:

DONE!

Submit in Google Classroom

Sample code ([back](#))

```
main.py x
1  import pygame
2  from datetime import date
3
4  # set up pygame modules
5  pygame.init()
6  pygame.font.init()
7  my_font = pygame.font.SysFont('Arial', 15)
8  pygame.display.set_caption("AP CSP Pygame!")
9
10 # set up variables for the display
11 size = (400, 300)
12 screen = pygame.display.set_mode(size)
13
14 name = "Mr. Miller"
15 slogan = "Slogan: python is cool"
16 date = "Date: " + str(date.today()) # get today's date!
17
18 # create text objects in white font
19 display_name = my_font.render(name, True, (255, 255, 255))
20 display_slogan = my_font.render(slogan, True, (255, 255, 255))
21 display_date = my_font.render(date, True, (255, 255, 255))
22
23 run = True
24 while run:
25     for event in pygame.event.get():
26         if event.type == pygame.QUIT:
27             run = False
28
29     screen.fill((0, 0, 0)) # black background
30     screen.blit(display_name, (0, 0)) # top left
31     screen.blit(display_slogan, (150, 120)) # somewhere in middle
32     screen.blit(display_date, (270, 280)) # bottom right
33     pygame.display.update()
34
```