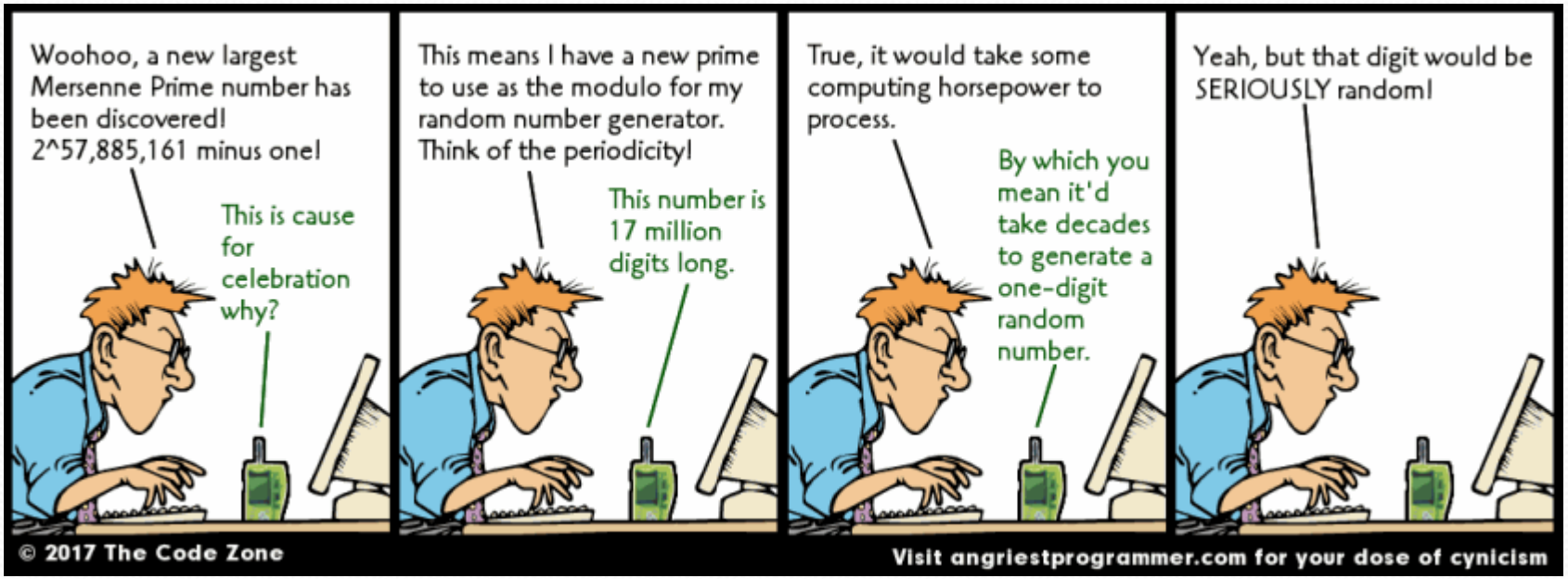
CS 115, Autumn 2021

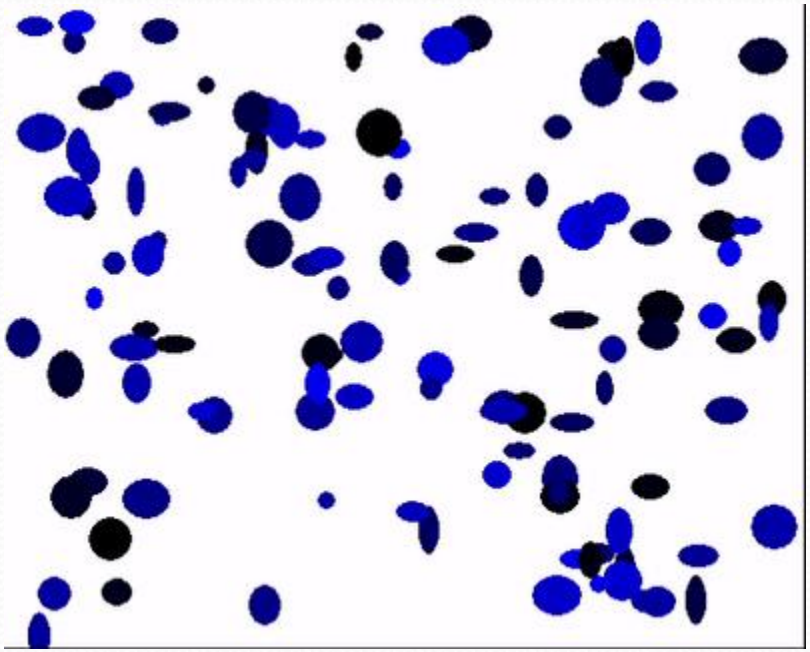
Lecture 11: graphics; random



Thanks to Marty Stepp and Stuart Reges for parts of these slides

Making it rain

- What if we want to animate rain?
  - We could just draw a lot of blue circles in specific x, y locations
    - That wouldn't look very realistic – rain is much more random!



Pseudo-Randomness

- Computers generate numbers in a predictable way using a mathematical formula
- Function may include current time, mouse position
  - In practice, hard to predict or replicate
- True randomness uses natural processes
  - Atmospheric noise (http://www.random.org/)
  - Lava lamps (patent #5732138)
  - Radioactive decay

Random

- `random` generates pseudo-random numbers.
  - `random` can be accessed by including the following statement:

```
from random import *
```

Method name	Description
`random()`	returns a random float in the range [0, *1*) in other words, 0 inclusive to *1* exclusive
`randint(`***min, max***`)`	returns a random integer in the range [min, max] in other words, min to max inclusive

  - Example:

```
from random import *
random_number = randint(1, 10)    # 1-10
```

Generating random numbers

- To get a number in arbitrary range [*min*, *max*] inclusive:

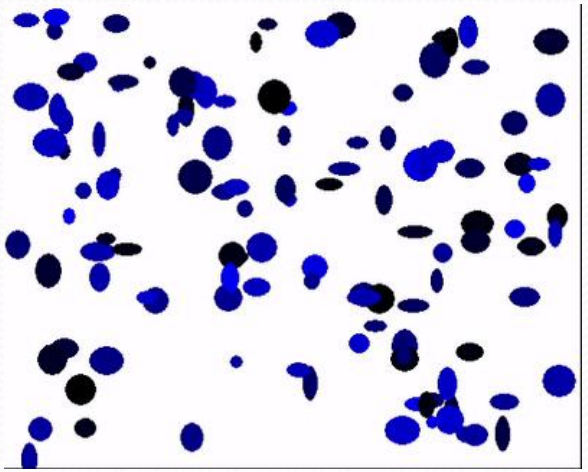
```
randint(min, max)
```

  - Example: A random integer between 4 and 10 inclusive:

```
n = randint(4, 10)
```

Exercise: rain

- Write a program that draws a new oval on a `drawing_panel` every 10ms. All ovals should be:
  - At random x locations
  - At random y locations
  - Between 10 and 30 pixels wide
  - Between 10 and 30 pixels tall
  - Be random colors with 0 red, 0 green and a random amount of blue



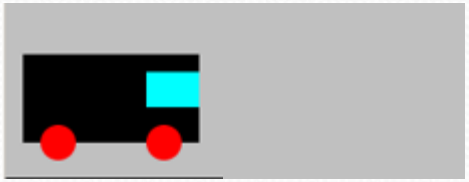
Constants

- When we have a constant value that our program depends on and we might want to change it later it is helpful to store this in a variable.
  - Make this a special variable that you guarantee will stay the same throughout an entire run of your program
  - Place it in an easy to find spot so the user can find it to change it between runs – convention: at the top of your file
  - Naming convention: ALL_CAPS
  - Example

```
PANEL_WIDTH = 500
PANEL_HEIGHT = PANEL_WIDTH // 2
```

Moving in a pattern

- How can we move a shape in a specific pattern?
  - Example: make the car appear to drive across the panel
  - We will need to keep track of the x location of the car and make it increase a little each time we draw the car
    - How can we keep track of this?

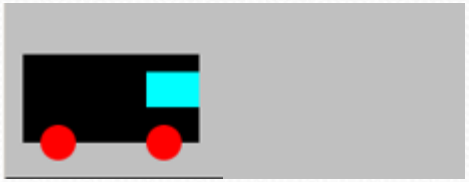


```
def repeat():
    x = 10
    y = 30
    panel.fill_rect(x, y, 100, 50, "black")
    panel.fill_oval(x + 10, y + 40, 20, 20, "red")
    panel.fill_oval(x + 70, y + 40, 20, 20, "red")
    panel.fill_rect(x + 70, y + 10, 30, 20, "cyan")

panel = drawing_panel(260, 100, "gray")
panel.animate(50, repeat)
```

Can we use variables?

- Option 1: declare some variables inside our animation function
  - problem: how do we make the x, y position change?
  - possible solution: set the variables to themselves plus something
    - problem: what are x and y the first time?
      - this doesn't work



```
def repeat():
    x = x + 10
    y = y + 10
    panel.fill_rect(x, y, 100, 50, "black")
    panel.fill_oval(x + 10, y + 40, 20, 20, "red")
    panel.fill_oval(x + 70, y + 40, 20, 20, "red")
    panel.fill_rect(x + 70, y + 10, 30, 20, "cyan")

panel = drawing_panel(260, 100, "gray")
panel.animate(50, repeat)
```

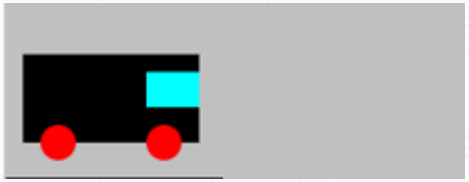
Can we use variables?

- Option 2: declare some variables **outside** our animation function
  - increase them **inside** the animation function
  - problem: our code won't run! Why not?
    - by default we can only change variables created inside a function when we are in a function

```
x = 10
y = 30

def repeat():
    x = x + 10
    y = y + 10
    panel.fill_rect(x, y, 100, 50, "black")
    panel.fill_oval(x + 10, y + 40, 20, 20, "red")
    panel.fill_oval(x + 70, y + 40, 20, 20, "red")
    panel.fill_rect(x + 70, y + 10, 30, 20, "cyan")

panel = drawing_panel(260, 100, "gray")
panel.animate(50, repeat)
```



global

- The **`global`** keyword allows us to change a variable from **outside** of a function when we are **inside** a function
  - create the variable outside of the function
  - before you assign the variable inside the function include a line with: `global` **variable_name**
  - now use this variable just as you would any other variable

```
x = 10
y = 30

def repeat():
    global x
    global y
    x = x + 10
    y = y + 10
    panel.fill_rect(x, y, 100, 50, "black")
    panel.fill_oval(x + 10, y + 40, 20, 20, "red")
    panel.fill_oval(x + 70, y + 40, 20, 20, "red")
    panel.fill_rect(x + 70, y + 10, 30, 20, "cyan")

panel = drawing_panel(260, 100, "gray")
panel.animate(50, repeat)
```