

00000000

# Tic Tac Toe

WL Hack Club

# intro to python

00000001

## strings

Strings are the technical term for text.

They must be surrounded in single or double quotation marks in Python.

You can show them on screen by using `print()`.

```
>>> print('hello world')  
hello world
```

# intro to python

00000010

## variables

Variables can store data in them, such as numbers or strings.

You can change their values using `=` and do math with them.

```
>>> my_number = 5
>>> my_number = my_number + 1
>>> print(my_number)
6
>>> my_number = 420
>>> print(my_number)
420
```

# intro to python

00000011

## comparisons

You can check if two numbers are equal by using the equality operator `==`.

```
>>> a = 4
>>> b = 4
>>> print(a == b)
True
```

```
>>> password = '196572b'
>>> guess = '106572b'
>>> print(password == guess)
False
```

# intro to python

00000100

## more comparisons

You can use `!=` to check if two things are *not* equal.

You can also use the math comparisons `>=`, `>`, `<=`, and `<`.

```
>>> print(5 >= 3)
True
>>> print(4 != 17)
True
```

# intro to python

00000101

## control flow: if this, else that

Control flow lets us determine what we want to do depending on some condition.

Let's say we're at a restaurant that serves alcoholic drinks and we need to make sure everyone is drinking legally.

```
>>> age = 15
>>> if age >= 21:
...     print('What would you like to order?')
... else:
...     print('Sorry, you are not allowed to drink alcohol yet!')
...
Sorry, you are not allowed to drink alcohol yet!
```

# intro to python

00000110

## data structures: lists

Lists can store multiple items in one variable.

You can define them using the brackets [ and ].

You access them with `list[number]`. Be careful, `number` starts at 0!

```
>>> mylist = ['first', 'second', 'third', 'fourth']
>>> print(mylist[0])
first
>>> print(mylist[2])
third
```

# intro to python

## data structures: lists

You can also change items inside a list.

```
>>> mylist = [1, 3, 5]
>>> mylist[0] = 2
>>> mylist[1] = 5000
>>> print(mylist)
[2, 5000, 5]
```



# intro to python

00001000

## data structures: lists

You can add more numbers too.

```
>>> mylist = [5, 4, 3]
>>> mylist.append(2)
>>> mylist.append('ice cream')
>>> print(mylist)
[5, 4, 3, 2, 'ice cream']
```

# intro to python

00001001

## data structures: tuples

Tuples are almost the same as lists.

But once you make them, *they can never be changed*.

We use ( and ) to define them.

```
>>> mytuple = (1, 2, 3)
```

```
>>> mytuple[0] = 3
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: 'tuple' object does not support item assignment
```

Error!

# intro to python

00001010

## iteration

You can use the `for` loop to perform an action for every item in a list or tuple.

```
>>> numbers = (2, 3, 4, 7)
>>> for x in numbers:
...     print(x)
...
2
3
4
7
```

In a `for` loop, we have a loop variable that changes each time we run it.

Here, the loop variable is `x`.

# intro to python

00001011

## functions

Functions are pieces of code that we write once and can use over and over again.

```
>>> def my_function():  
...     print('a useful function')
```

```
>>> my_function()  
a useful function  
>>> my_function()  
a useful function
```

# intro to python

00001100

## advanced functions

Functions can sometimes take in parameters.

Trivia: What's the difference between an argument and a parameter?

This lets you pass some information *to* a function.

```
>>> def be_annoying(word):  
...     print(word)
```

```
>>> be_annoying('apple')  
apple  
>>> be_annoying('orange')  
orange
```

# intro to python

00001101

## advanced functions

Functions can also return a value.

It's like the opposite of parameters: we can get information *from* a function.

```
>>> def give_me_five():  
...     return 5
```

```
>>> number = give_me_five()  
>>> print(number)  
5
```

# intro to python

00001110

## methods

Advanced functions in python are called *methods*.

You'll have to use a dot to access them.

We'll see a few of these in Tic Tac Toe. Don't worry too much about them for now.

```
game_won = self.core.check_victory()
if game_won:
    self.playing = Playing.ENDING
    clear_canvas(self.canvas)
    self.core.handle_victory()
    self.core.game_won = True
```

# Questions?



# intro to python

00010000

## coding challenge

You get a list of numbers. You need to give back a list of numbers. The new list should have each old number, but it should be doubled.

Input: `[1, 6, 5, 1, 7, 4, 12]`

Required Output: `[2, 12, 10, 2, 14, 8, 24]`

No cheating!

# intro to python

00010001

## coding challenge solution

```
>>> first_list = [1, 6, 5, 1, 7, 4, 12]
>>> answer = []
>>> for number in first_list:
...     answer.append(number * 2)
```

```
>>> answer
[2, 12, 10, 2, 14, 8, 24]
```

# intro to python

00010010

## coding challenge solution

Or, if you're a pro:

```
>>> answer = [x*2 for x in first_list]
```

# intro to python

00010011

## review

Now you know the basics of Python!

Let's start working on our game.

# Tic Tac Toe

WL Hack Club

# getting started

create your project in [repl.it](https://repl.it)


Create a new python project in [repl.it](https://repl.it).

# getting started

## installing pygame

Type `import pygame` into repl.it and run it.

Pygame should install automatically for you.

▶ Run main.py ▾ × + main.py

```
1 import pygame
```

```
2
```

# getting started

00010111

## initialize pygame

This first piece of code will give us our first pygame display!

But don't run it yet!

```
1
2  import pygame
3
4
5  #####
6  # Initialize pygame
7  #####
8
9
10 pygame.init()
11 pygame.display.set_caption("Tic Tac Toe by greateric")
12 canvas = pygame.display.set_mode((1200, 800))
13
14 # "Times New Roman", "Courier New", "Ubuntu Mono", etc.
15 my_font = pygame.font.SysFont( name: 'Calibri', size: 36)
16
```



# getting started

00011000

## add some helper functions

Add this after your first piece of code:

```
17
18 #####
19 # Game state
20 #####
21
22
23 #####
24 # Game functions
25 #####
26
27
28 def draw_centered_text(canvas: pygame.Surface, text: pygame.Surface, x: float, y: float) -> None:
29     text_rect = text.get_rect()
30     canvas.blit(text, dest: (x - text_rect.width/2, y - text_rect.height/2))
31
```

# getting started

00011001

## drawing our board

No more boring black screen! Now let's add a function that draws our board.

```
32
33 def draw_board():
34     global canvas
35     canvas.fill(0x0000aa)
36     draw_centered_text(canvas, my_font.render(text: 'Tic Tac Toe', antialias: True, color: 0xffffffff), x: 600, y: 30)
37     pygame.draw.rect(canvas, color: 0x000000, rect: (495, 100, 10, 600))
38     pygame.draw.rect(canvas, color: 0x000000, rect: (695, 100, 10, 600))
39     pygame.draw.rect(canvas, color: 0x000000, rect: (300, 295, 600, 10))
40     pygame.draw.rect(canvas, color: 0x000000, rect: (300, 495, 600, 10))
41
```

# getting started

00011010

## having a real display

Now let's make our display actually work.

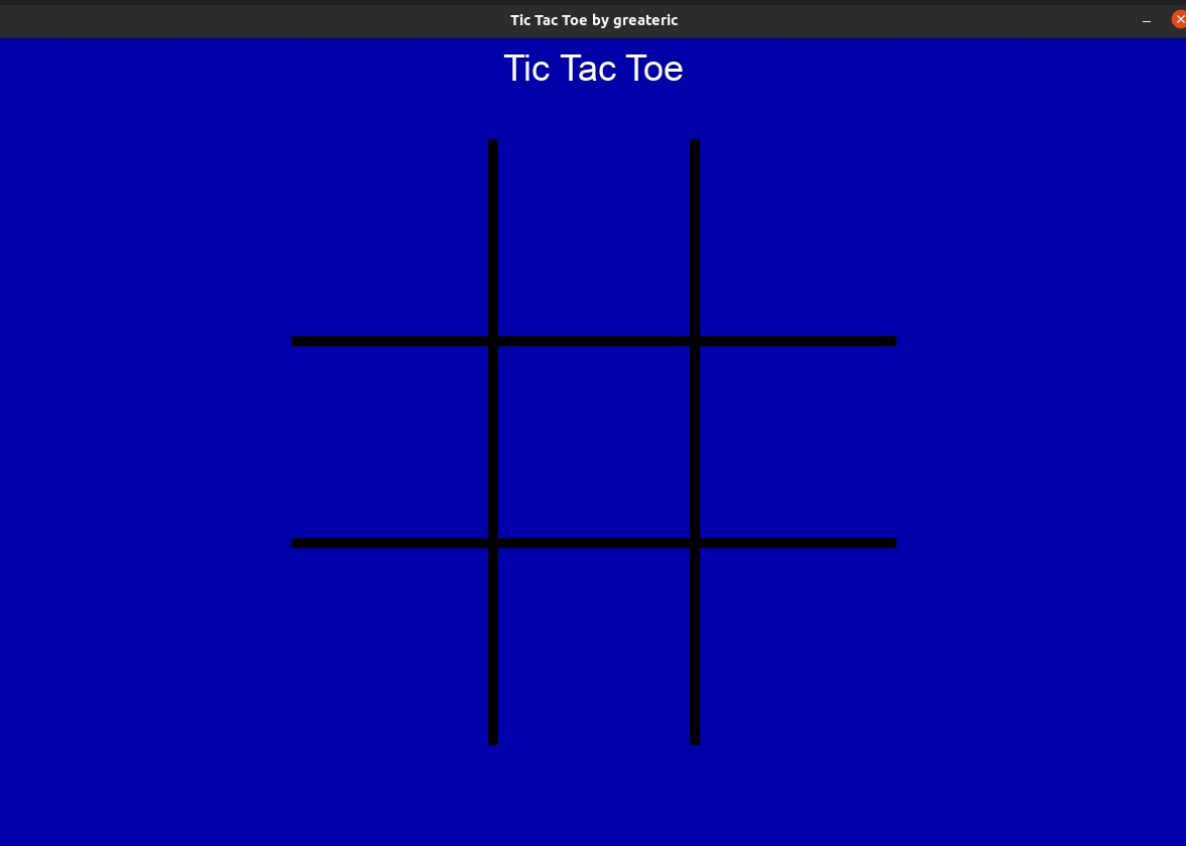
Add this to the end of your code.

```
42
43 #####
44     # Main game
45 #####
46
47
48 while True:
49     for event in pygame.event.get():
50         if event.type == pygame.QUIT:
51             pygame.quit()
52             exit()
53     draw_board()
54     pygame.display.update()
55
```

# getting started

00011011

## review



Great!

But it's just a  
screen for now.

Let's give it some  
functionality!

# Tic Tac Toe

WL Hack Club

# adding more

## another helper function

Add this code right after `draw_board` and right before the `Main game` block.

This code will help us figure out what to do if we click on the screen.

```
42
43 def mouse_button_press(pos):
44     x = pos[0]
45     y = pos[1]
46     # Check X
47     if 310 <= x <= 490:
48         board_x = 0
49     elif 510 <= x <= 690:
50         board_x = 1
51     elif 710 <= x <= 890:
52         board_x = 2
53     else:
54         # The user didn't click on the board.
55         return
56     # Check Y
57     if 110 <= y <= 290:
58         board_y = 0
59     elif 310 <= y <= 490:
60         board_y = 1
61     elif 510 <= y <= 690:
62         board_y = 2
63     else:
64         # The user didn't click on the board.
65         return
66     print('You clicked on X:', board_x, 'and Y:', board_y)
67
```

# adding more

## add helpers to the game

Add this code to the  
`Main game` section to  
use our helper function!

```
68
69 #####
70 # Main game
71 #####
72
73
74 while True:
75     for event in pygame.event.get():
76         if event.type == pygame.QUIT:
77             pygame.quit()
78             exit()
79         if event.type == pygame.MOUSEBUTTONDOWN:
80             mouse_button_press(event.pos)
81     draw_board()
82     pygame.display.update()
83
```

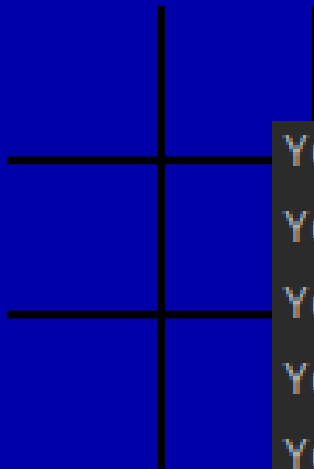
# adding more

00011111

## review

Tic Tac Toe by greateric

Tic Tac Toe



You clicked on X: 0 and Y: 0

You clicked on X: 1 and Y: 1

You clicked on X: 0 and Y: 2

You clicked on X: 1 and Y: 2

You clicked on X: 0 and Y: 1

You clicked on X: 1 and Y: 1

You clicked on X: 1 and Y: 1

You clicked on X: 1 and Y: 1

We can recognize when we click on the board!

Now let's turn this into real Tic Tac Toe!



00100000

# Tic Tac Toe

WL Hack Club

# X and O

00100001

## adding state

Add this code in the  
`Game state` section.

This will initialize our  
3x3 board and have X  
go first.

```
17
18 #####
19     # Game state
20 #####
21
22
23 board = [
24     ['none', 'none', 'none'],
25     ['none', 'none', 'none'],
26     ['none', 'none', 'none']
27 ]
28     turn = 'X'
29
```

# X and O

00100010

## drawing the board again

Now we want to add these to the board.

```
40
41 def draw_board():
42     global board, canvas
43     canvas.fill(0x0000aa)
44     draw_centered_text(canvas, my_font.render(text='Tic Tac Toe', antialias=True, color=0xffffffff), x=600, y=30)
45     pygame.draw.rect(canvas, color=0x000000, rect=(495, 100, 10, 600))
46     pygame.draw.rect(canvas, color=0x000000, rect=(695, 100, 10, 600))
47     pygame.draw.rect(canvas, color=0x000000, rect=(300, 295, 600, 10))
48     pygame.draw.rect(canvas, color=0x000000, rect=(300, 495, 600, 10))
49     for x in (0, 1, 2):
50         for y in (0, 1, 2):
51             x_coordinate = 400 + 200*x
52             y_coordinate = 200 + 200*y
53             if board[y][x] == 'X':
54                 draw_centered_text(canvas,
55                                     my_font.render(text='X', antialias=True, color=0xff5555ff),
56                                     x_coordinate, y_coordinate)
57             elif board[y][x] == 'O':
58                 draw_centered_text(canvas,
59                                     my_font.render(text='O', antialias=True, color=0xffff55ff),
60                                     x_coordinate, y_coordinate)
61
```

Add this to your  
`draw_board()`  
helper function.

# X and 0

## clicking on the board

Now update `mouse_button_press` so the board can be updated every time we click on it.

Also note that the `print` statement has been deleted.

00100011

```
62
63 def mouse_button_press(pos):
64     global board, turn
65     x = pos[0]
66     y = pos[1]
67     # Check X
68     if 310 <= x <= 490:
69         board_x = 0
70     elif 510 <= x <= 690:
71         board_x = 1
72     elif 710 <= x <= 890:
73         board_x = 2
74     else:
75         # The user didn't click on the board.
76         return
77     # Check Y
78     if 110 <= y <= 290:
79         board_y = 0
80     elif 310 <= y <= 490:
81         board_y = 1
82     elif 510 <= y <= 690:
83         board_y = 2
84     else:
85         # The user didn't click on the board.
86         return
87     if board[board_y][board_x] == 'none':
88         board[board_y][board_x] = turn
89         turn = 'X' if turn == '0' else '0'
90
```

# X and O

00100100

review

X	O	O
O	X	X
X	X	O

O	O	X
X	O	O
X	X	X

We're almost there!

Now we just need to figure out who wins.

# Tic Tac Toe

WL Hack Club

# winning the game

00100110

## our last helper function

Add this helper code in the `Game` functions section.

```
62
63 def check_win():
64     global board
65     for winner in ('X', 'O'):
66         if (
67             # rows
68             board[0][0] == board[0][1] == board[0][2] == winner
69             or board[1][0] == board[1][1] == board[1][2] == winner
70             or board[2][0] == board[2][1] == board[2][2] == winner
71             # columns
72             or board[0][0] == board[1][0] == board[2][0] == winner
73             or board[0][1] == board[1][1] == board[2][1] == winner
74             or board[0][2] == board[1][2] == board[2][2] == winner
75             # diagonals
76             or board[0][0] == board[1][1] == board[2][2] == winner
77             or board[0][2] == board[1][1] == board[2][0] == winner
78         ):
79             return winner
80     return 'none'
81
```

# winning the game

00100111

## displaying the winner

Add this code to the end of the `draw_board()` helper function.

This will display if someone wins.

```
52     y_coordinate = 200 + 200*y
53     if board[y][x] == 'X':
54         draw_centered_text(canvas,
55                             my_font.render(text='X', antialias=True, color=0xff5555ff),
56                             x_coordinate, y_coordinate)
57     elif board[y][x] == 'O':
58         draw_centered_text(canvas,
59                             my_font.render(text='O', antialias=True, color=0xffff55ff),
60                             x_coordinate, y_coordinate)
```

```
61     winner = check_win()
62     if winner == 'X':
63         draw_centered_text(canvas, my_font.render(text='X won!', antialias=True, color=0xff5555ff), x=600, y=80)
64     elif winner == 'O':
65         draw_centered_text(canvas, my_font.render(text='O won!', antialias=True, color=0xffff55ff), x=600, y=80)
```



```
40
41 def draw_board():
42     global board, canvas
43     canvas.fill(0x0000aa)
44     draw_centered_text(canvas, my_font.render( text: 'Tic Tac Toe', antialias: True, color: 0xffffffff), x: 600, y: 30)
45     pygame.draw.rect(canvas, color: 0x000000, rect: (495, 100, 10, 600))
46     pygame.draw.rect(canvas, color: 0x000000, rect: (695, 100, 10, 600))
47     pygame.draw.rect(canvas, color: 0x000000, rect: (300, 295, 600, 10))
48     pygame.draw.rect(canvas, color: 0x000000, rect: (300, 495, 600, 10))
49     for x in (0, 1, 2):
50         for y in (0, 1, 2):
51             x_coordinate = 400 + 200*x
52             y_coordinate = 200 + 200*y
53             if board[y][x] == 'X':
54                 draw_centered_text(canvas,
55                                     my_font.render( text: 'X', antialias: True, color: 0xff5555ff),
56                                     x_coordinate, y_coordinate)
57             elif board[y][x] == 'O':
58                 draw_centered_text(canvas,
59                                     my_font.render( text: 'O', antialias: True, color: 0xffff55ff),
60                                     x_coordinate, y_coordinate)
61     winner = check_win()
62     if winner == 'X':
63         draw_centered_text(canvas, my_font.render( text: 'X won!', antialias: True, color: 0xff5555ff), x: 600, y: 80)
64     elif winner == 'O':
65         draw_centered_text(canvas, my_font.render( text: 'O won!', antialias: True, color: 0xffff55ff), x: 600, y: 80)
```

# winning the game

00101001

# stopping the game

Add this code to `mouse_button_press()` to stop the game once someone wins.

```
87
88 def mouse_button_press(pos):
89     global board, turn
90     x = pos[0]
91     y = pos[1]
92     if check_win() != 'none':
93         # Don't allow the user to click if the game is over.
94         return
95     # ... rest of :mouse_button_press: ...
```

# winning the game

00101010

## review

Tic Tac Toe

X won!

X

O

X

O

X

O

X

Tic Tac Toe

O won!

X

O

O

O

X

X

O

X

# Congratulations!

You've created a functioning Tic Tac Toe game!

Now challenge your friends!

Check out our code

[github.com/WLHackClub/basicictactoe](https://github.com/WLHackClub/basicictactoe)

# Tic Tac Toe

WL Hack Club