Smash - Python – Microsoft Arcade

The following instructions will take you through the steps of creating a game replicating the famous 1976 game Breakout where you try to destroy rows of blocks. A screenshot from the Atari 2600 version is shown below.

Use the D-Pad to move your tank left and right and the A button to fire the tanks cannon.

Controls

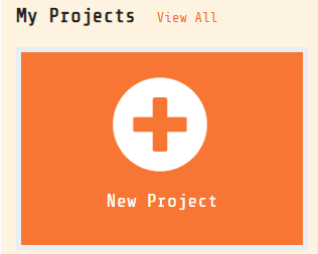
To control the players paddle, use the direction keys on the console. To launch a ball, use the A key on the console. Alternatively, it may be easier to use the keyboard mappings for the keys as follows:

	Up (↑ or W)						
Left (← or A)		**Right** (→ or D)		**B** (E, X or Enter)		**A** (Q, Z or Space)	
	Down (↓ or S)						

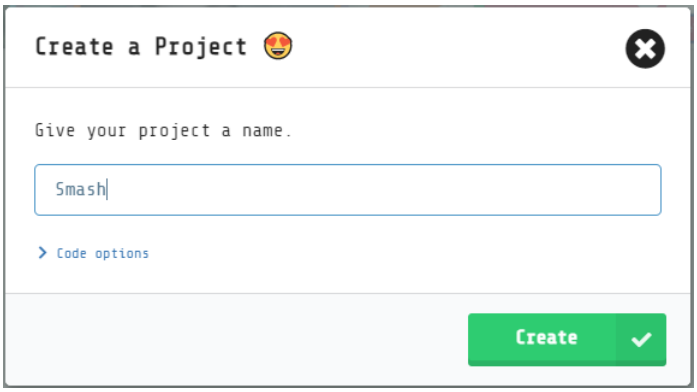
Smash - Python – Microsoft Arcade

Step 1 – Setup MakeCode Arcade for Python development

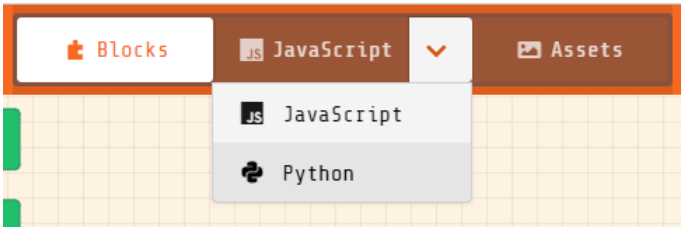
Navigate to https://arcade.makecode.com and create a new project by pressing the “New Project” icon.



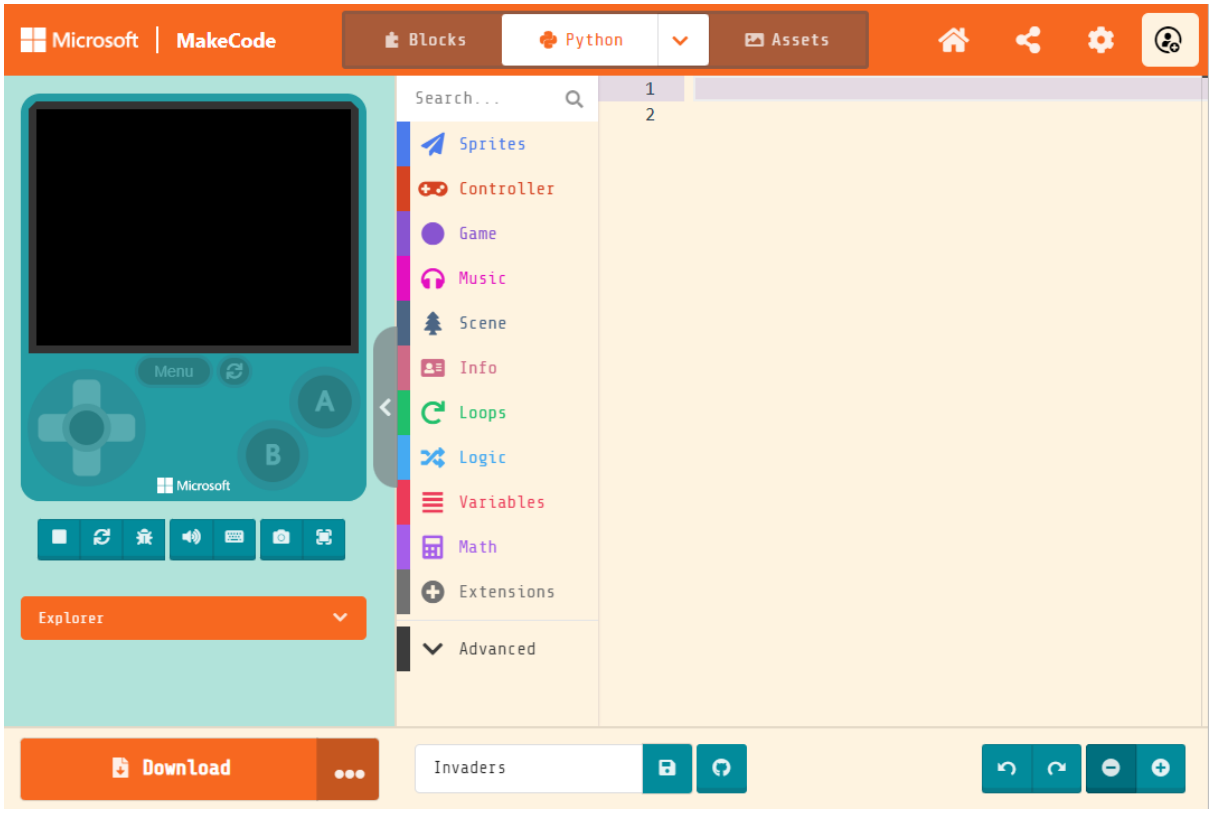
Name your game Smash and click Create.



From the top bar menu, click the down arrow and select Python.



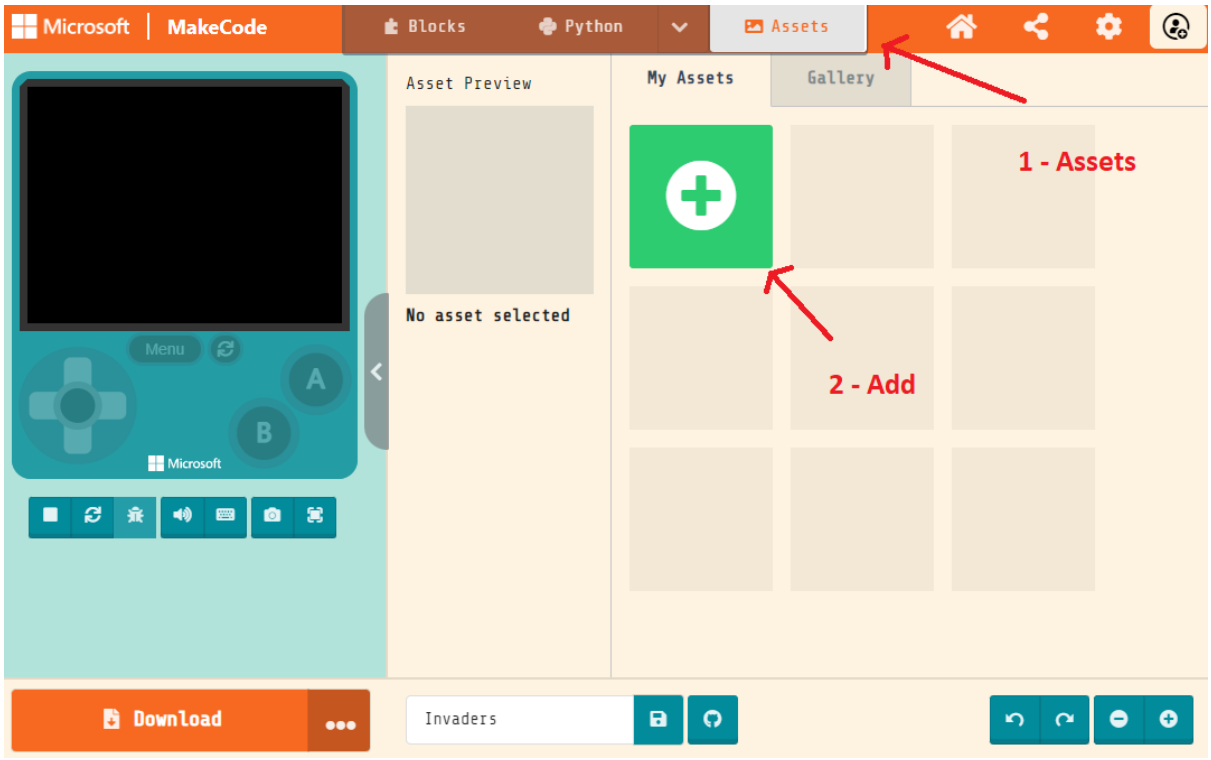
You should be presented with an editor that looks like this.



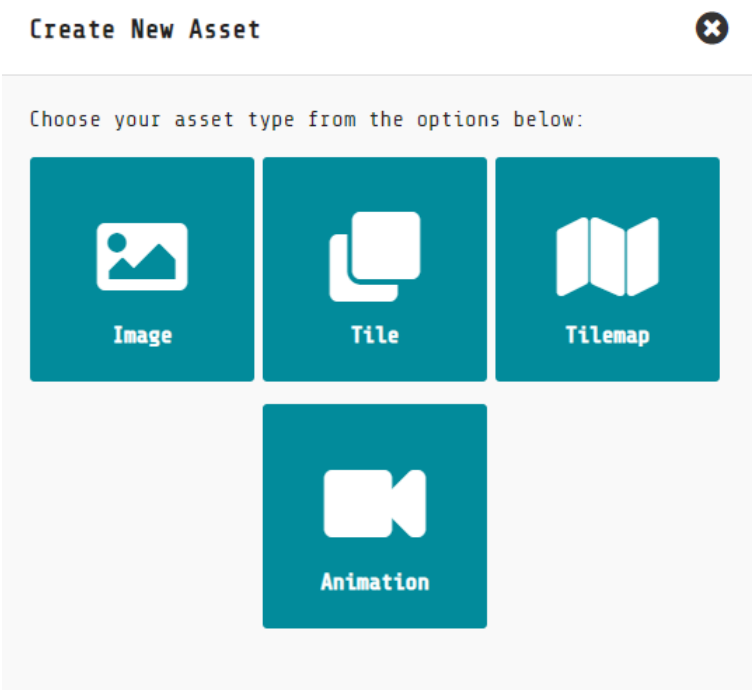
Smash - Python – Microsoft Arcade

Step 2 – Create the paddle sprite

From the top bar menu, select “Assets” and then click the “Add” button:

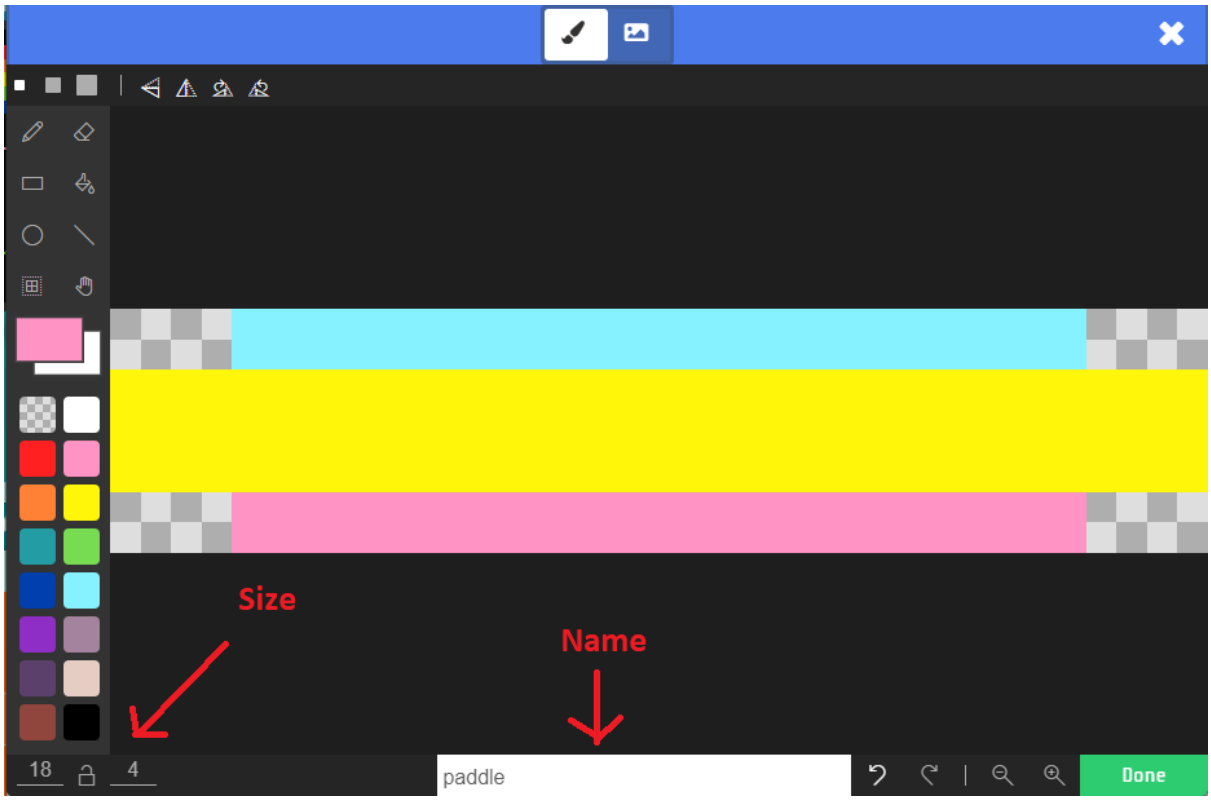


From the “Create New Asset” dialogue, select “Image”.



Smash - Python – Microsoft Arcade

Change the sprites size to 18 x 4, name to paddle, draw the paddle and click done. The image below is an example but you can use your own imagination to draw your paddle.



Switch back to Python using the top bar menu and enter the following code which will create a paddle sprite and allow it to be moved left and right with the controller.

```
paddle = sprites.create(assets.image("""paddle"""), SpriteKind.player)
paddle.set_position(80, 108)
paddle.set_stay_in_screen(True)
controller.move_sprite(paddle, 150, 0)
```

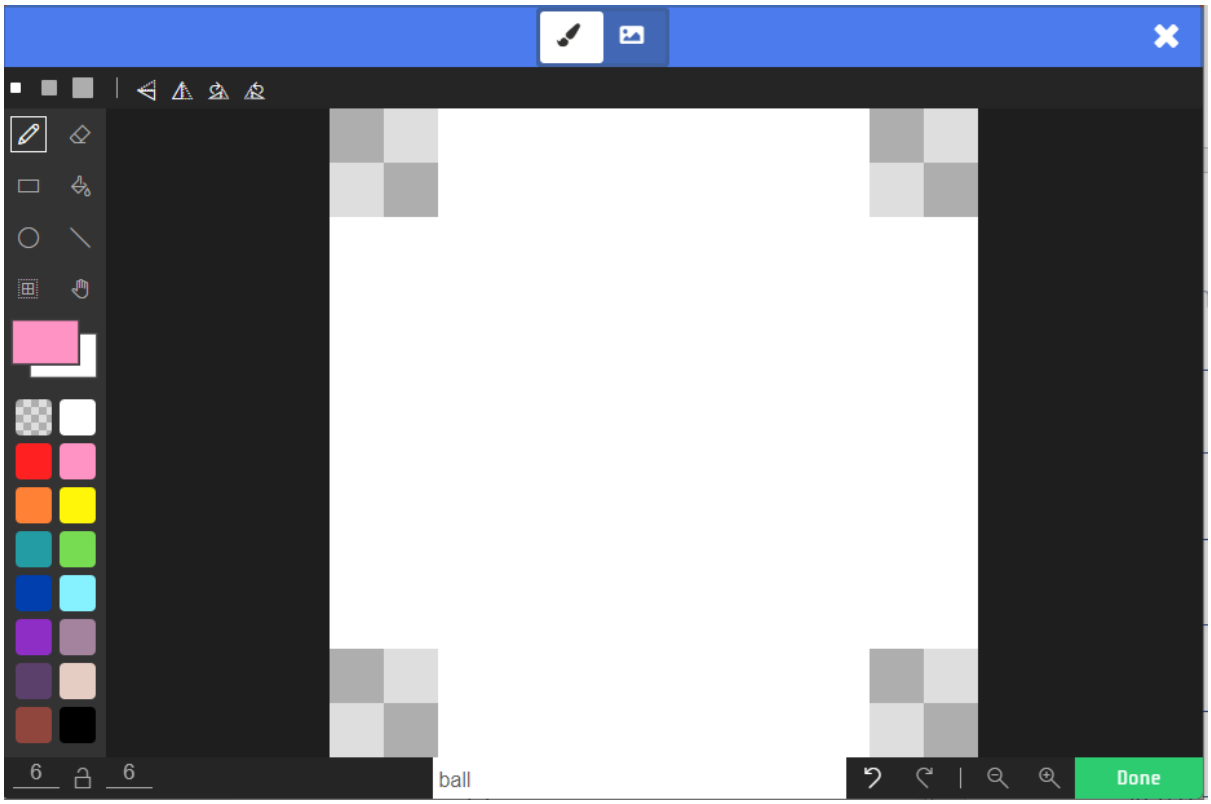
Press the play button on the simulator to try out your game.

Notice that the paddle is not completely at the bottom of the screen. This is intentional as that space is where the score counter and remaining lives counter will go. We will also use the gap to help determine when the paddle misses the ball.

Smash - Python – Microsoft Arcade

Step 3 – Adding the ball

For the ball, create a new 6 x 6 asset called ball.



Add the following code that creates the ball and keeps it in the screen.

```
ball = sprites.create(assets.image("""ball"""), SpriteKind.projectile)
ball.set_stay_in_screen(True)
ball.set_bounce_on_wall(True)
```

If you run the game now, the ball remains still in the middle of the screen. Before the ball gets launched, it should stay with the paddle along the bottom of the screen. Add the following code.

```
launching = True

def game_update():
    global launching
    if launching:
        ball.set_position(paddle.x, paddle.y - 5)

game.on_update(game_update)
```

The `global` keyword in the above code ensures that the global variable launching is used and not a local variable of the function.

If you run the game now and move the paddle, the ball will follow it. Add the following code to launch the ball and it should now bounce around the screen:

```
def launch():
    global launching
    ball.set_velocity(paddle.vx + randint(-20, 20), -100)
    launching = False

controller.A.on_event(ControllerButtonEvent.PRESSED, launch)
```

The horizontal speed of the ball is set to the speed of the paddle plus a little random variation. Presently, the ball just passes through the paddle. Add the following code which will make the ball reflect off the paddle and apply a little bit of extra momentum from the paddle.

```
def ball_hits_paddle(paddle, ball):
    ball.set_velocity(ball.vx + (paddle.vx * 0.2), -100)

sprites.on_overlap(SpriteKind.player, SpriteKind.projectile, ball_hits_paddle)
```

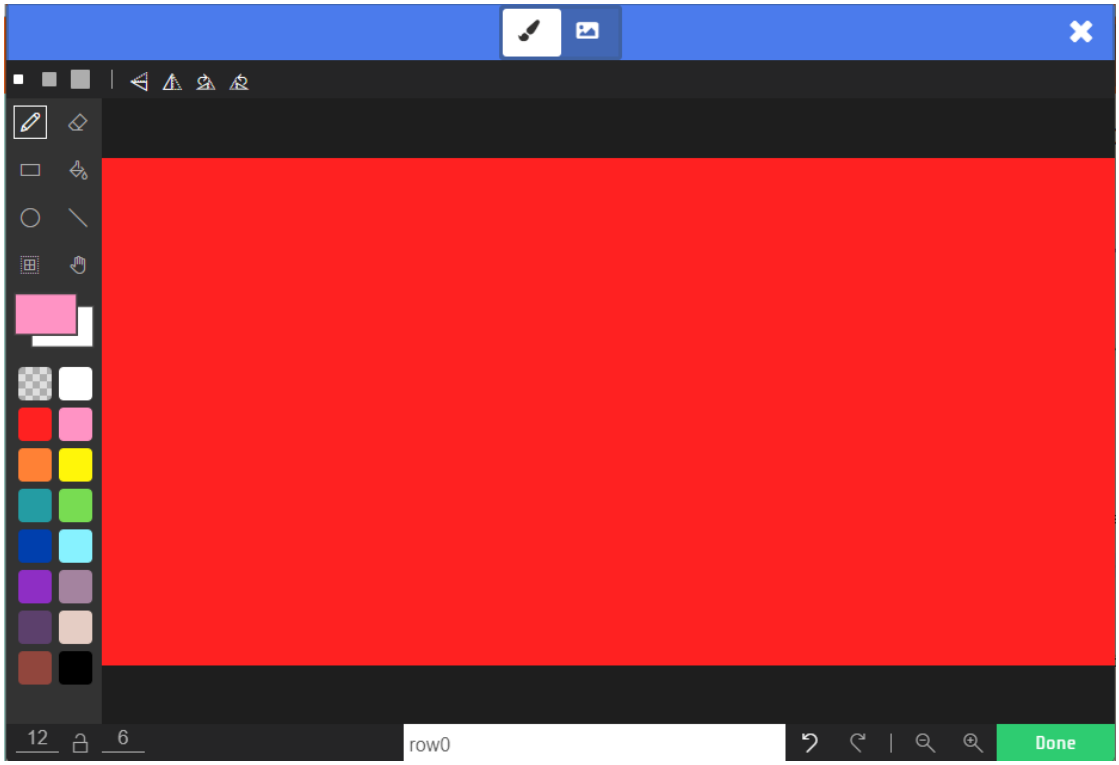
When playing the game now, notice how the paddle can be used to deflect the ball. However, when the paddle misses the ball, it bounces off the bottom of the screen. When the paddle misses the ball, it should be thought of as dropping the ball. Modify your `game_update()` function to add in the `elif` clause to test for the ball passing past the paddle. We will check both it's vertical position and its vertical velocity. The full code for the `game_update()` function is given below:

```
def game_update():
    global launching
    if launching:
        ball.set_position(paddle.x, paddle.y - 5)
    elif ball.y > 113 and ball.vy > 0:
            launching = True
```

Smash - Python – Microsoft Arcade

Step 4 – Adding the blocks

We are going to put 6 rows of blocks on the screen. We want each row to be a different colour, therefore we will need to create 6 different image assets. Each block will be 12 x 6 pixels. Create 6 assets with the names “row0”, “row1”, “row2”, “row3”, “row4”, “row5” and use whichever colours you wish. Here is my block row0.



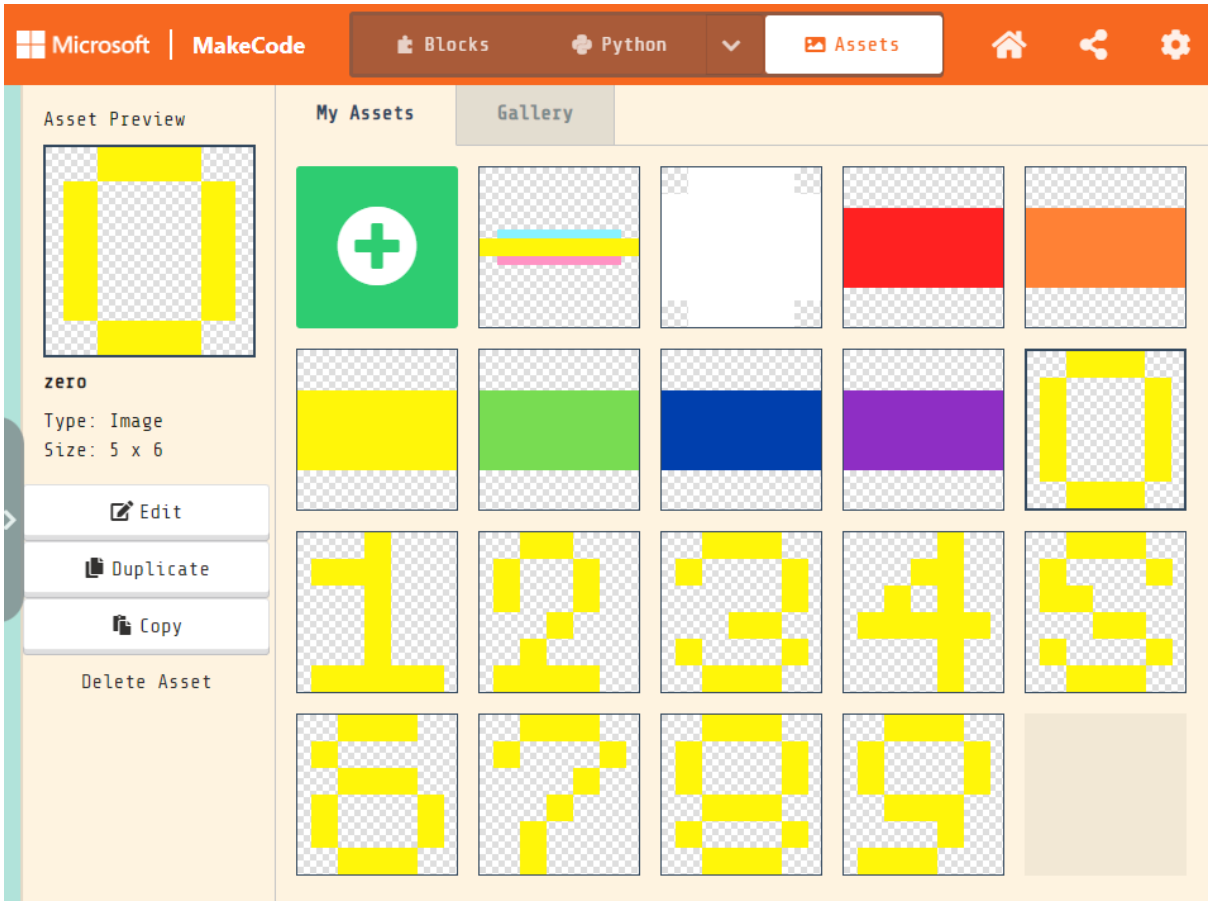
Add the following code which will create the 6 rows of blocks, using an array to select the correct image for each row of blocks.

```
block_images = [
    assets.image("""row0"""),
    assets.image("""row1"""),
    assets.image("""row2"""),
    assets.image("""row3"""),
    assets.image("""row4"""),
    assets.image("""row5"""),
    ]

def create_blocks():
    for y in range(6):
        for x in range(12):
            block = sprites.create(block_images[y], SpriteKind.enemy)
            block.set_position((x * 12) + 6 + 7, (y * 6) + 10)

create_blocks()
```

Smash - Python – Microsoft Arcade

Step 5 – Scoring

For the score, we are not going to use the built-in score feature of MakeCode Arcade as it puts it at the top of the screen, and we want the score to go along the bottom. We are therefore going to have to create our own number sprites. These sprites should be 5 x 6 pixels and called “zero”, “one”, “two”, “three”, “four”, “five”, “six”, “seven”, “eight” and “nine”. The screen shot below shows some example sprites:



Add the following code to you game which will create the “000” counter in the bottom of the screen which will look like the following.

```
digitImages = [
    assets.image("""zero"""),
    assets.image("""one"""),
    assets.image("""two"""),
    assets.image("""three"""),
    assets.image("""four"""),
    assets.image("""five"""),
    assets.image("""six"""),
    assets.image("""seven"""),
    assets.image("""eight"""),
    assets.image("""nine"""),
    ]

digits = [
    sprites.create(digitImages[0], SpriteKind.projectile),
    sprites.create(digitImages[0], SpriteKind.projectile),
    sprites.create(digitImages[0], SpriteKind.projectile)]

for digit in digits:
    digit.set_position(5, 114)

digits[1].set_position(digits[1].x + 6, digits[1].y)
digits[2].set_position(digits[2].x + 12, digits[2].y)
```

If you run your game now, the score will show but will not increase and the blocks will not get destroyed because we have not programmed that functionality yet.

```
score = 0

def show_score():
    global score
    display = "000"
    display = display + score
    length = len(display)
    idx = int(display[length -1])
    digits[2].set_image(digitImages[idx])
    idx = int(display[length -2])
    digits[1].set_image(digitImages[idx])
    idx = int(display[length -3])
    digits[0].set_image(digitImages[idx])
```

```
def hit_block(ball, block):
    global score
    block.start_effect(effects.spray, 250)
    block.destroy()
    ball.set_velocity(ball.vx, -ball.vy)
    score = score + 1
    show_score()

sprites.on_overlap(SpriteKind.projectile, SpriteKind.enemy, hit_block)
```

Step 7 – New wave of blocks

Now when playing your game, when you destroy all the blocks, the game continues. What should happen is that a new set of blocks are created. This is achieved by adding the following piece of code to the `hit_block()` function after the call to show_score():

```
def hit_block(ball, block):
    global score
    …
    show_score()
    if len(sprites.all_of_kind(SpriteKind.enemy)) <= 0:
        global launching
        launching = True
        game_update()
        create_blocks()
```

Step 6 – Lives and High score

Presently, it does not matter how many time the ball is dropped, the game continues. Now we will add the following code to put the 3 x life indicators at the bottom of the screen.

```
lives = [
    sprites.create(assets.image("""ball"""), SpriteKind.projectile),
    sprites.create(assets.image("""ball"""), SpriteKind.projectile),
    sprites.create(assets.image("""ball"""), SpriteKind.projectile)]

for life in lives:
    life.set_position(140, 115)

lives[1].set_position(lives[1].x + 8, lives[1].y)
lives[0].set_position(lives[0].x + 16, lives[0].y)

lifes = 3
```

For a life to be lost each time the ball is dropped, an update is needed to the `game_update()` function. The new code to be added is at the end of the function in the `elif` statement. Modify your `game_update()` function to look like the following:

```
def game_update():
    global launching
    if launching:
        ball.set_position(paddle.x, paddle.y - 5)
    elif ball.y > 113 and ball.vy > 0:
        launching = True
        # This bit for lives.
        global lifes
        global lives
        lifes = lifes - 1
        lives[lifes].set_flag(SpriteFlag.INVISIBLE, True)
        if lifes <= 0:
            global score
            info.set_score(score)
            game.over(False)
```

Step 7 – Challenges

Use all that you have learned to complete the following challenges

1. Add a background image to your game.
2. Add sound for block explosions.
3. Add an extra life for each complete set of blocks destroyed.
4. Have different layouts of blocks which make pictures.