

Handling Key Presses

Gibson Lam, David Rossiter and Leo Tsui

Pressing a Key

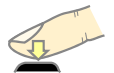
- Let's look at how to handle keys
- There are two kinds of action for a key:
 - pressing (push down) a key
 - releasing a key
- In this presentation we focus on handling the pressing (push down) of a key, which is usually more useful than the releasing of a key

The Key



- You have to state the name of a specific key when you set up the handling of a keyboard event
 - For example, you can use 'a', 'b', ... 'z' or '0' ... '9'
- You can also use a special name, such as:
 - 'Return' – Enter key
 - 'Up' – up arrow key
 - 'Escape' – Esc key
 - 'Down' – down arrow key

The Key Press Event



- The `onkeypress()` function assigns an event handling function for handling the key press event of a particular key
- For example:

```
def mykeyfunc():  
    . . .
```

Whenever the user presses 'a' this function will be executed

```
turtle.onkeypress(mykeyfunc, 'a')
```

The key press event is applied to the turtle window

The mykeyfunc function is assigned to the key press event

A key ('a' in this example) that is handled by the event handler

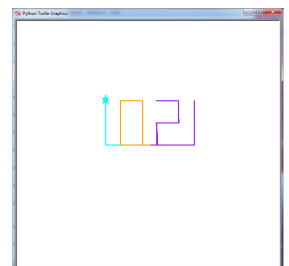
Listening for Keyboard Events



- Imagine you are using your computer normally
- When you press a key, the key goes to the window which currently has *focus*
- If you want key presses to go to your program, then you need to make sure your turtle window has the focus
- `turtle.listen()` does that – after this code, your program's turtle window has the focus
- (You also need `turtle.done()` at the end)

Key Events Example

- This example uses keys to control the movement of the turtle:
 - Up key – move forward
 - Down key – move backward
 - Left key – rotate left
 - Right key – rotate right
- It also allows colour change:
 - 'o' key – orange
 - 'p' key – purple
 - 'c' key – cyan



Key Events Example 1/3 – Event Handlers for Turtle Movement

```
pixels_for_one_step = 4  
angle_for_rotation = 5
```

```
def moveforward():  
    turtle.forward(pixels_for_one_step)  
  
def movebackward():  
    turtle.backward(pixels_for_one_step)
```

*These event
handler functions
move the turtle
forward (up
arrow key) or
backward
(down arrow key)*

```
def rotateleft():  
    turtle.left(angle_for_rotation)  
  
def rotateright():  
    turtle.right(angle_for_rotation)
```

*These event handler
functions rotate the
turtle to the left (left
arrow key) or right
(right arrow key)*

Key Events Example 2/3 – Event Handlers for Changing Colour

```
def orange():  
    # Change the pen color and  
    # the turtle to orange  
    turtle.color("orange")
```

For the 'o' key

```
def purple():  
    # Change the pen color and  
    # the turtle to purple  
    turtle.color("purple")
```

For the 'p' key

```
def cyan():  
    # Change the pen color and  
    # the turtle to cyan  
    turtle.color("cyan")
```

For the 'c' key

Key Events Example 3/3 – Main Program

```
turtle.shape("turtle")  
turtle.speed(0)  
turtle.color("purple")  
turtle.width(3)
```

```
turtle.onkeypress(moveforward, "Up")  
turtle.onkeypress(movebackward, "Down")  
turtle.onkeypress(rotateleft, "Left")  
turtle.onkeypress(rotateright, "Right")
```

*Assign the up,
down, left and
right keys for
moving the turtle*

```
turtle.onkeypress(orange, "o")  
turtle.onkeypress(purple, "p")  
turtle.onkeypress(cyan, "c")
```

*Assign the 'o', 'p' and 'c'
keys for the colour
change functions*

```
turtle.listen()
```

*Make sure keyboard presses go to the
turtle window, not any another window*

```
turtle.done()
```

Must have this at the end