Variables

Variable	Description	Example
string	a combination of letters, numbers, or symbols.	name = “Python”
integer	an integer	apples = 3
float	a number with a decimal place	height = 52.05
boolean	a variable that can either be true or false	result = True
list	is one of the top data types, only a list of them. To manipulate a list visit: http://www.thegeekstuff.com/2013/06/python-list/?utm_source=feedly	listVar = [“1”,”2”,”3”]

Operators

Operator	Description	Example
+	add a and b is a sum if a and b are numbers or adds two string together	a + b
-	subtract a and b	a - b
*	multiple a and b	a * b
/	divide a by b	a / b
%	divide a by b and return remainder	a % b
raw_input()	stores what the user enters in as a string	name = raw_input()
int(raw_input())	stores what the user enters in as an int	age = int(raw_input))

Spacing:
Everything in python is indented to run under a code block.

```
def spam():
   eggs = 12
   return eggs

print spam()
```

Comments:

```
# single line comments
‘’’
multiple
lined
comment
‘’’
```

Comparers

Comparer	When it returns true
a == b	when a and b are equal
a != b	when a and b are not equal
a < b	when a is less than b
a < = b	when a is less than or equal to b
a > b	when a is greater than b
a >=b	when a is greater than or equal to b

Boolean Operators

Boolean	When it returns true
a **and** b	if both a and b are true
a **or b**	is one of a or b is true or both are
not a	if a is false

Conversions

```
str[100]          # converts numeric type to string
int[‘100’]        # converts a string to an integer
float[‘100.5’]  # converts a string to a float’
```

If Statements

‘’’If statements run a comparison. If true the code under the if runs. Otherwise Python will check the elif comparisons. Finally the else will run if all else are false. ‘’’

```
if apples == 0:
   # runs if there are no apples
elif apples == 1:
   # runs if there is one apple
else:
   # runs if there is not one/zero apples
```

Loops

```
for i in range(1,10): # using a range
    print i

for i in {1,3,4,6} # using a list
    print i

i = 0
while i <=100
    print i

# when using a loop, if you want to exit use a break symbol.
    break

‘’’
 when using a loop, if you want to skip the rest of the code in the current
 iteration, use a continue symbol
‘’’
continue
```

Functions

```
‘’’
pieces of code which can be reused whenever they are called.
functions can take in data and return data
everything inside a function must be indented at least 4 spaces

‘’’
def aFunctionName(arg1, arg2, ….):
    # code resides here
    return a_value

#Example
def addTwoNumbers(num1, num2):
    return (num1 + num2)
```

Classes

```
‘’’
Object oriented programming allows you to create objects with code. An
object is code, with information, that can be created multiple times. You
can use objects to model information on cars, airplanes, ice cream, and
anything.
__init__() function initializes the object's values by using the self keyword
‘’’


# Syntax
class className (object_which_it_inherits):
     fields
     def __init__(arg1, arg2, ….):
          self.field1 = arg1
     def getField():
          return self.field

# Example

class Animal(object):

    def __init__(self,name):
        self.name = name

     def getName():
        return self.name

zebra = Animal("Jeffrey") # you don’t pass self
print zebra.name
```

Free Space :)

PyGame Commands

Command	What it does	Example
pygame.key.get_pressed()	gets keyboard input from user	keys = pygame.key.get_pressed()

keys[K_a]	returns true if a is pressed. Look at http://www.pygame.org/docs/ref/key.html for key names
	Common Keys:
	K_RIGHT, K_LEFT, K_UP, K_DOWN, K_w, K_s, K_a, K_d, K_SPACE, K_0, K_1, …. , K_9
		if keys[K_a]:
		direction = 1
		elif keys[K_d]:
		direction = 0
		else:
		direction = -1

pygame.display.set_mode(width, height)	This represents images as surface objects. The display.set_mode creates a new surface object that represents the actual displayed graphics. The screen should be the same size of the background image you are going to use.	screen = pygame.display.set_mode((640, 480))
pygame.image.load(image location)	loads an image by using a string argument with the images path. Use png and convert to alpha to keep transparency. Other Acceptable Formats: JPG, PNG, TGA, GIF	background = pygame.image.load("images/background.png").convert_alpha()
blit(source, destination)	blit draws one image onto another, where destination can be a pair of coordinates (or an image) and source is an image to paint onto the destination.	screen.blit(background, (0, 0))
pygame.sprite.Group()	creates a group to add sprites (game objects) to. Methods that come with this object are on *(http://www.pygame.org/docs/ref/sprite.html#pygame.sprite.Group)*	all_group = pygame.sprite.Group() all_group.add(player) all_group.remove(player) all_group.clear()
pygame.event.get()	returns a list of all the events that happened in pygame (signals that are sent when things are done, buttons are clicked, ect.) Use to check if the user quit.	for event in pygame.event.get() if event.type == QUIT: pygame.quit() sys.exit()
spritecollide(sprite, group, dokill, collided = None)	finds sprites in a group that intersect another sprite dokill, if set to true, removes collided sprites from group this method returns a list contains all sprites in a group that intersect with another sprite	collide_list = pygame.sprite.spritecollide(player, enemy_group, False, collided = None) for enemy in collide_list: life -= 1