Variables, Expressions and Statements

creative thinking with programming day 2

Constants

Constants are fixed values that are always the same. 10 always is equal to 10.

Numeric constants are all of the numbers.

String constants can also be created if you use single quote marks

```
print 'hello world'
print 122
```

Variables

Variables are things of a type that **vary**. You can think of them like nouns in a sentence.

x is a **number**.
x = 10

x is the variable name you made up
= is the **assignment operator**
10 is the value

This **statement** assigns the value 10 to the variable x.

Variable names

Variables change over time. For example before a user kills a ghost their score might be 0 and after 1.

```
score = 0
score =1
```

Variables change over time. For example before a user
kills a ghost their score might be 0 and after 1.

Use logical variable names. If something is your timer, maybe call it myTimer.
If a variable is keeping track of the number of lives left, lives works well

```
myTimer = 0
lives =1
```

Variable names

* Must start with a letter or underscore _

* Must consist of letters and numbers and underscores

* Case Sensitive

Good: spam   eggs spam23   _speed
Bad: 23spam   #sign var.12
Different: spam   Spam   SPAM

Reserved words

Don't use these! Python forbids it.

```
and      del     for     is      raise
assert   elif    from    lambda  return
break    else    global  not     try
class    except  if      or      while
continue exec    import  pass    yield
def      finally in      print
```

Sentences or Lines

Quiz!
What are the variables here?
What are the constants?
What is the reserved word python is using?

```
x = 2
y = 3
x = x + y

print x
```

Expressions

Whenever you have an assignment and something else, you have an **expression** that must be solved before it is assigned to the variable on the left

```
# this is an expression
x  = x + y
```

Mathematical operators

Note division in python 2.7 is weird.

If you divide any **whole numbers** together and get a remainder, python gives you a whole number and *truncates the sum.*

For example:

```
>>> 8/3
2
```

Mathematical operators

Note division in python 2.7 is weird.

if you divide any **floating point number by a whole number** you get a floating point number

For example:

```
>>> 8.0/3
2.6666666666666665
```

Please Excuse My Dear Aunt Sally

Python evaluates just like algebra.

```
>>> 5*7 / 2 - 3
# first eval = 35
# second 35/2 which truncates
# to 17
#third 17 -3
>>>14
```

Parenthesis
Power
Multiplication
Division
Addition
Subtraction

Types matter

A **type** is the kind of thing something is.

Python knows what type something is.

Python auto types variables but what type something is still matters.

Common types:

int = whole number

float = decimal number

bool = True or False *(note the number case T and F)*

str = "hello I am a cat"

Types matter

What happens if you try and add together unsuitable types like an integer and a string?

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

What type is it anyway?

How do you know what type something is in Python if it auto casts?

Ask!

```
>>> x = 10
>>> type(x)
<type 'int'>
```

Type casting

What if you have a string and need a int?
Use the int() function!

```
>>> x = "10"
>>> type(x)
<type 'str'>
>>> x = int(x)
>>> type(x)
<type 'int'>
```

What if you have an int and need a string?
use the string function!

```
>>> str(y)
'10'
```

What if you have a int you need to be a float?
use the float function!

```
>>> x = 10
>>> float(x)
10.0
```

String overloaded operators

You can add and multiply strings together.

```
>>> "hi"*3
'hihihi'
```

```
>>> "hello " + "world"
'hello world'
```

raw_input

raw_input saves user input from the console as a string.

```
>>> x = raw_input("what's your age?")
what's your age?36
>>> print x
36
>>> type(x)
<type 'str'>
```
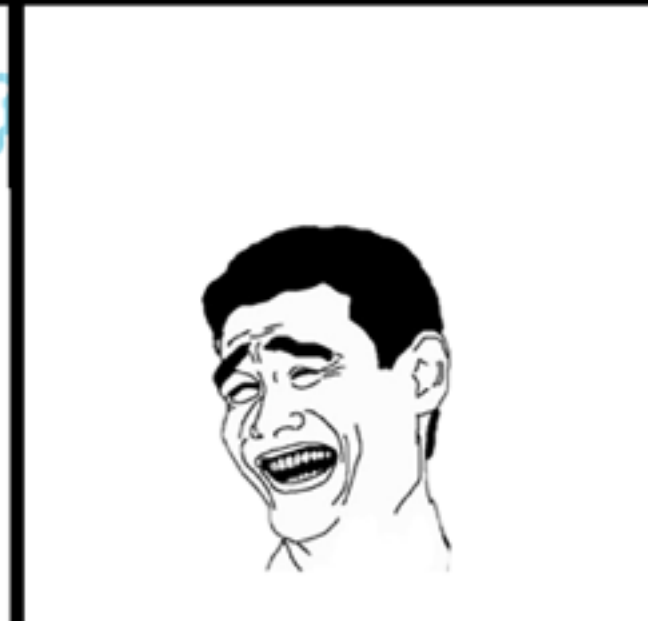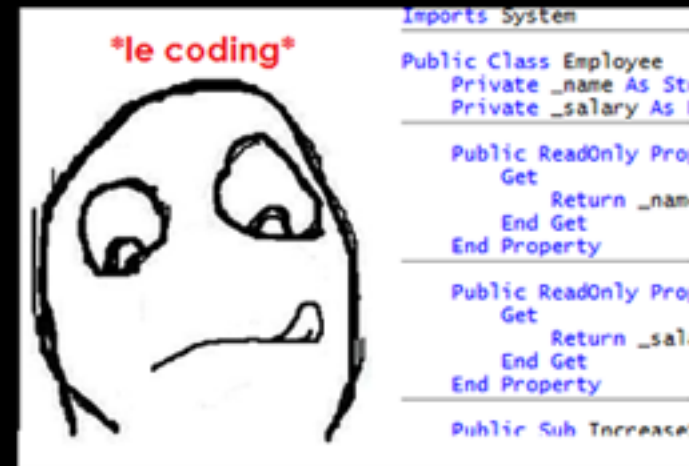
Comments

Use them often and use them well.

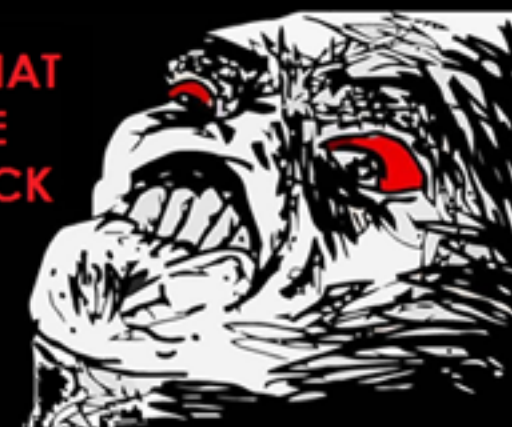They are your notes in your code

They in no way effect the code

#I am a comment!

Processing

Processing is awesome!

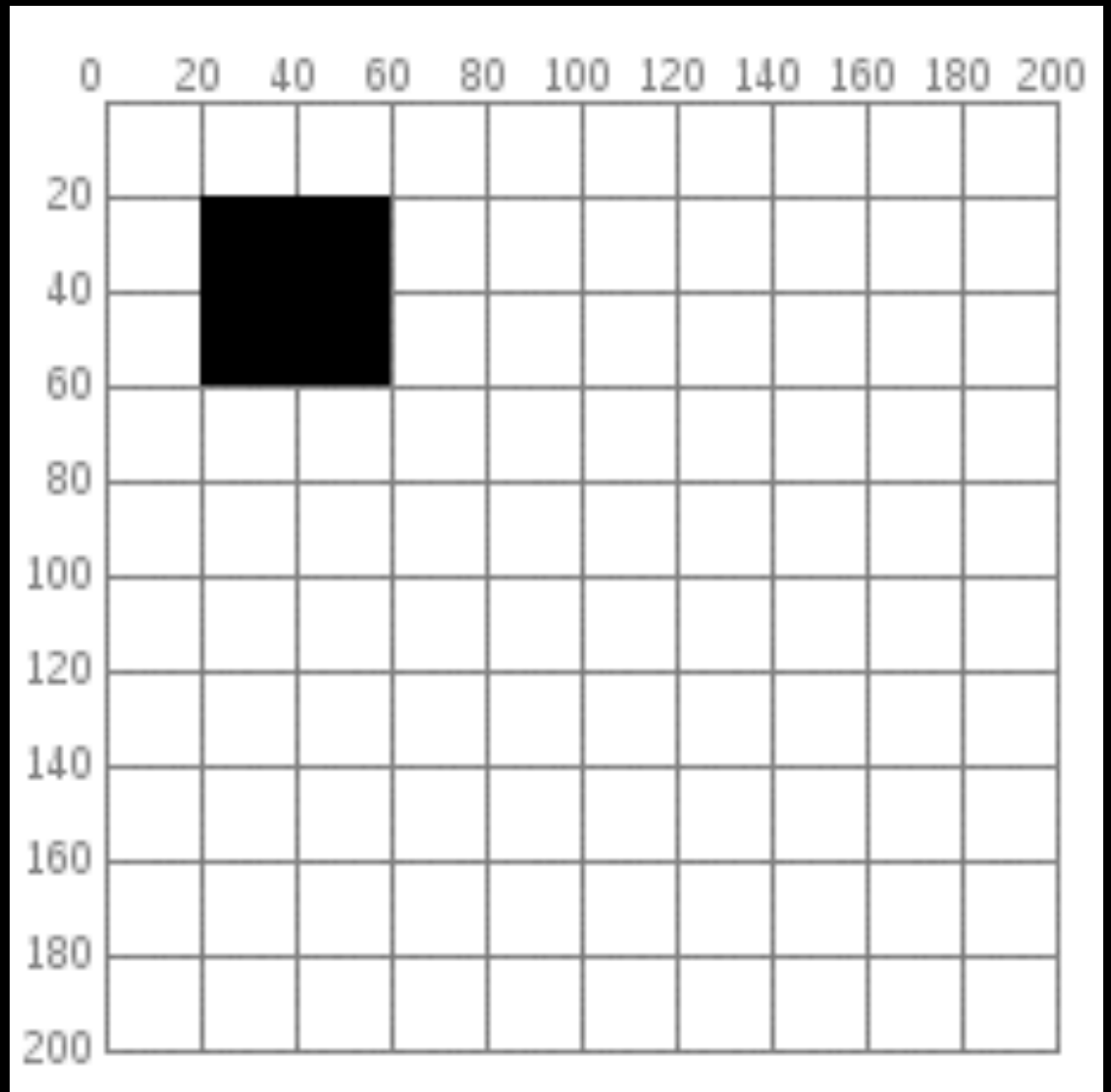It has two built in functions you can use to set up your drawing and then animate it

We've not *really* covered functions yet, but think of them like the verbs your code can do! They are blocks of code that work together to do something.
use a tab to group your code under a function header

setup let's you set up your drawing

draw runs every frame. (generally as fast as your CPU will let it run)

```
def setup():
 size(1000,1000);
 background(0);


def draw():
 ellipse(100, 100, 100, 100)
```

x →

y ↓

Your grid is in the upper left corner and starts with 0,0

http://py.processing.org/