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
```

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)
```

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'>
```

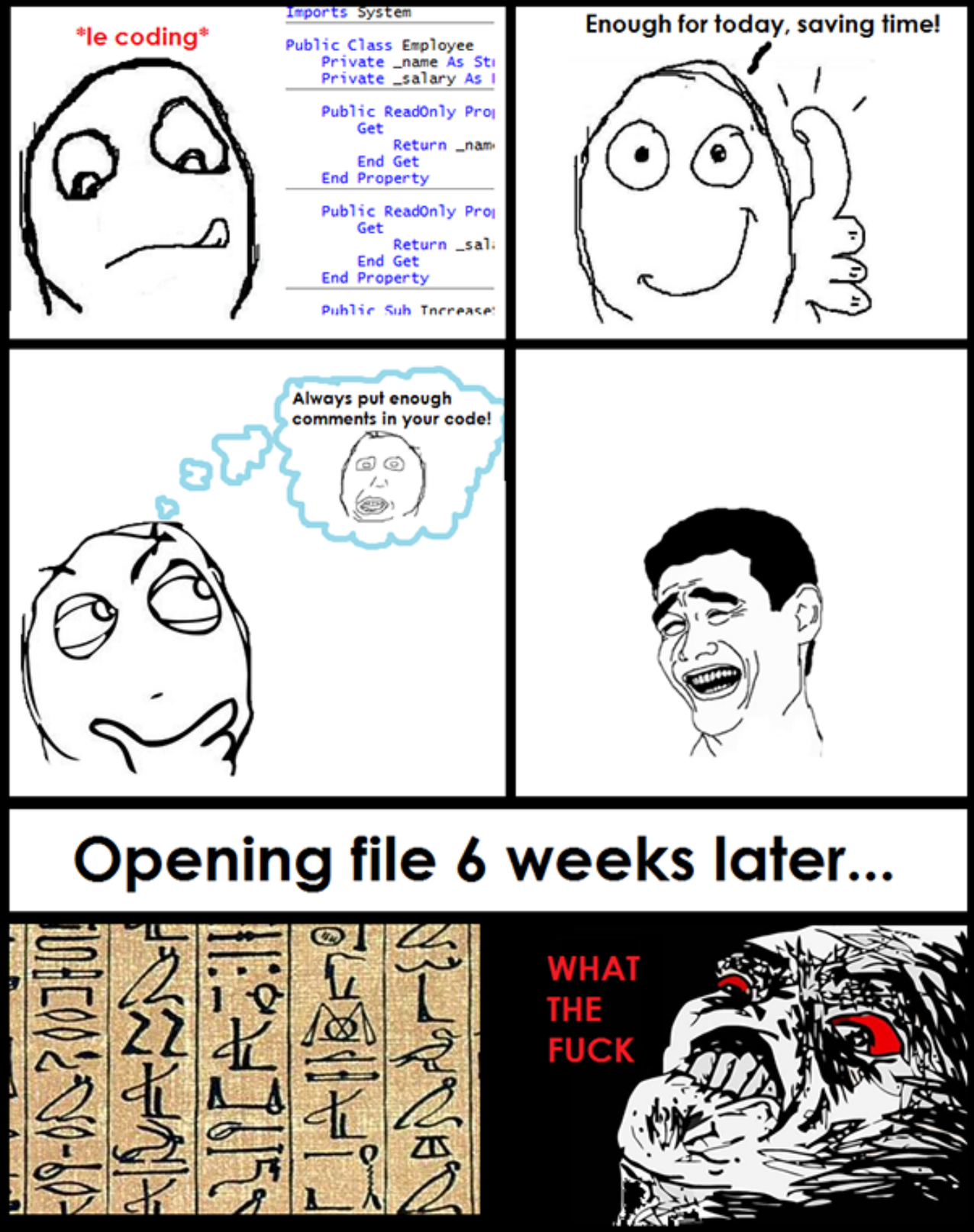
Comments

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!
```



Turtle Module

Math is all good but let's look at our code by using a visual module to draw. That module is called turtle.

There are many *modules* in Python that provide very powerful features that we can use in our own programs.

Turtle graphics, as it is known, is based on a very simple metaphor.

Turtle Objects, Methods & States

For now, you can think of an object like a thing in the world – a cup, a car, a phone.
Each object has things it can do = methods

And it can also have **attributes** — (sometimes called *properties*).
alex.color(“red”) will make alex draw red

The color of the turtle, the width of its pen(tail), the position of the turtle within the window, which way it is facing are all part of its current **state**.

Similarly, the window object has a background color which is part of its state.

For Loops

You could draw multiple turtles: example: turtleHurd.py
For loops are for iteration

Example:forloop101, forLoopTurtles

Challenge: have a turtle draw a triangle with a for loop

```
x=0
for name in [“alex”,“tiny” “sam”]:
  print (Hi I am name )
```