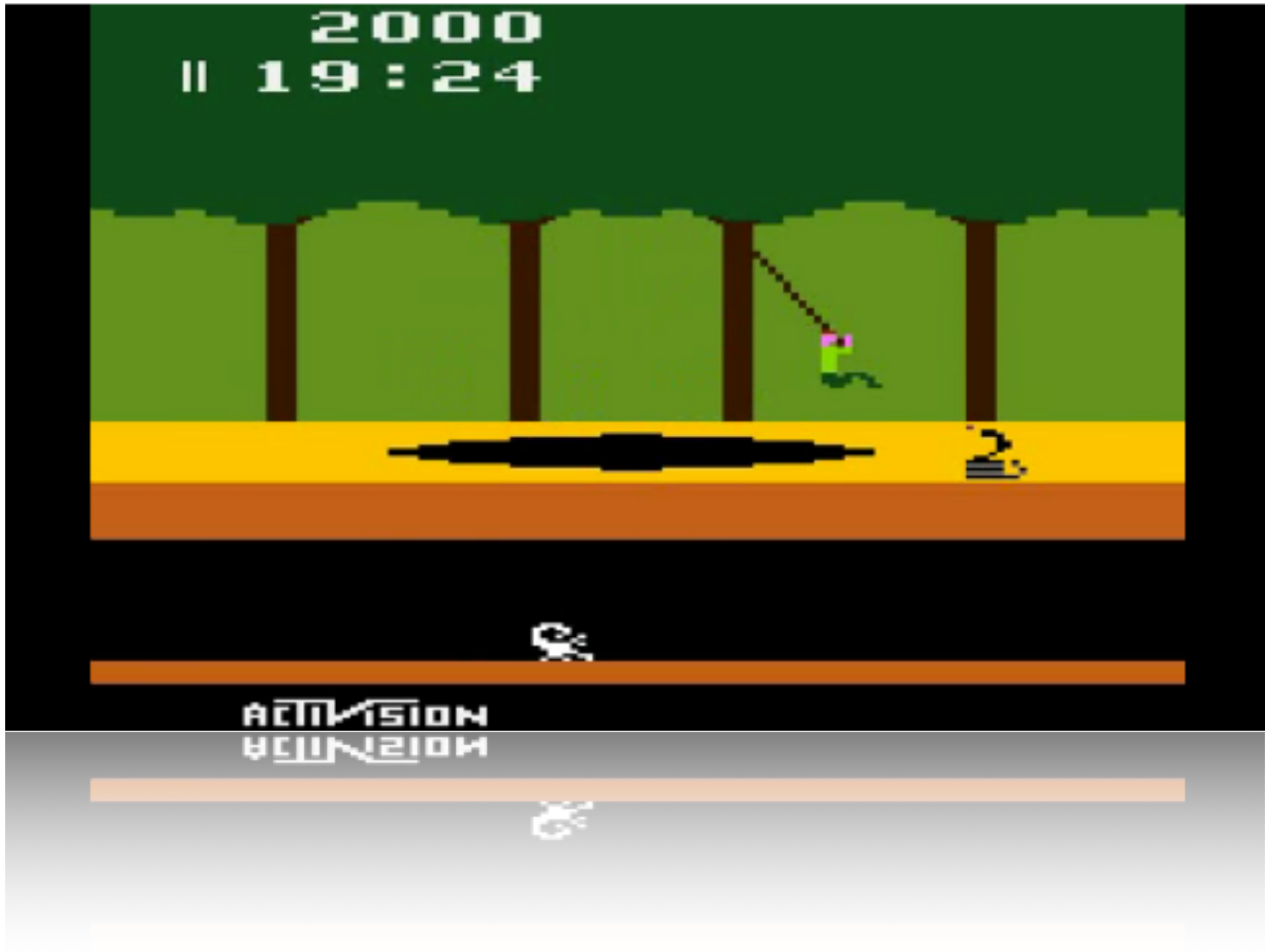
Advanced Interactions



```
# airline, number, heading to, gate, time (decimal hours)
flights = [("Southwest",145,"DCA",1,6.00),("United",31,"IAD",1,7.1),("United",302,"LHR",5,6.5),\
          ("Aeroflot",34,"SVO",5,9.00),("Southwest",146,"CDA",1,9.60), ("United",46,"LAX",5,6.5),\
          ("Southwest",23,"SBA",6,12.5),("United",2,"LAX",10,12.5),("Southwest",59,"LAX",11,14.5),\
          ("American", 1,"JFK",12,11.3),("USAirways", 8,"MIA",20,13.1),("United",2032,"MIA",21,15.1),\
          ("SpamAir",1,"AUM",42,14.4)]
```

```
>>> help(flights.sort)
    L.sort(cmp=None, key=None, reverse=False) -- stable sort *IN PLACE*;
    cmp(x, y) -> -1, 0, 1

>>> flights.sort(key=lambda x: x[4]) ; flights
[('Southwest', 145, 'DCA', 1, 6.0),
 ('United', 46, 'LAX', 5, 6.5),
 ('United', 302, 'LHR', 5, 6.5),
 ('United', 31, 'IAD', 1, 7.0999999999999996),
 ('Aeroflot', 34, 'SVO', 5, 9.0),
 ('Southwest', 146, 'CDA', 1, 9.5999999999999996),
 ('American', 1, 'JFK', 12, 11.300000000000001),
 ('Southwest', 23, 'SBA', 6, 12.5),
 ('United', 2, 'LAX', 10, 12.5),
 ('USAirways', 8, 'MIA', 20, 13.1),
 ('SpamAir', 1, 'AUM', 42, 14.4),
 ('Southwest', 59, 'LAX', 11, 14.5),
```

Multiple column sorting

`operator.`**`itemgetter`**(*item*[, *args...*])¶
Return a callable object that fetches *item* from its operand using the operand's `__getitem__()` method. If multiple items are specified, returns a tuple of lookup values.

http://docs.python.org/library/operator.html#module-operator

```
>>> flights.sort(key=operator.itemgetter(4,1,0))
[('Southwest', 145, 'DCA', 1, 6.0),
 ('United', 46, 'LAX', 5, 6.5),
 ('United', 302, 'LHR', 5, 6.5),
 ('United', 31, 'IAD', 1, 7.0999999999999996),
 ('Aeroflot', 34, 'SVO', 5, 9.0),
 ('Southwest', 146, 'CDA', 1, 9.5999999999999996),
 ('American', 1, 'JFK', 12, 11.300000000000001),
 ('United', 2, 'LAX', 10, 12.5),
 ('Southwest', 23, 'SBA', 6, 12.5),
 ('USAirways', 8, 'MIA', 20, 13.1),
 ('SpamAir', 1, 'AUM', 42, 14.4),
 ('Southwest', 59, 'LAX', 11, 14.5),
 ('United', 2032, 'MIA', 21, 15.1)]
```

Try/Except/Finally

Billy: Let's keep going with "Airplanes", for $200.

Bobby Wheat: "Airplanes" for $200: "And *what* is the Deal With the Black Box?" [Tommy buzzes in] Tommy!

Tommy: It's the *only* thing that survives the crash - why don't they build the **whole** plane out of the Black Box!

http://snltranscripts.jt.org/91/91rstandup.phtml

Wrap volatile code in try/except/finally

```
>>> tmp = input("Enter a number and I'll square it: ") ; print(float(tmp)**2)
Enter a number and I'll square it: monty
ValueError: invalid literal for float(): monty
```

instead....

```
>>> def f():
try:
   tmp = input("Enter a number and I'll square it: ")
   print(float(tmp)**2)
except:
   print("dude. I asked you for a number and %s is not a number." % tmp)
finally:
   print("thanks for playing!")
>>> f()
Enter a number and I'll square it: 3
9.0
thanks for playing!
>>> f()
Enter a number and I'll square it: monty
dude. I asked you for a number and monty is not a number.
thanks for playing!
```

Wrap volatile code in try/except/finally

```
try:
    tmp = raw_input("Enter a number " + \
                    and I'll square it: ")
    print(float(tmp)**2)
except:
    print("dude. I asked you for a number and " + \
          "%s is not a number." % tmp)
finally:
    print("thanks for playing!")
```

volatile stuff

upon error, jump here inside except and execute that code

regardless of whether you hit an error, execute everything inside the finally block

- errors in Python generate what are called "exceptions"
- exceptions can be handled differently depending on what kind of exception they are
  (we'll see more of that later)
- except "catches" these exceptions
- you do not have to catch exceptions (try/finally) is allowed. Finally block is executed no matter what!

```
>>> try:
  print("eat at" % joes)
finally:
  print("bye.")
bye.
---------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython console>", line 2, in <module>
NameError: name 'joes' is not defined
```

exec & eval

`exec` is a statement which executes strings as if they were Python code

```
>>> a = "print('checkit')"
>>> exec a
checkit
>>> a = "x = 4.56"
>>> exec(a)
>>> print(x)
4.56
>>> exec("del x")
>>> print x
------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython console>", line 1, in <module>
NameError: name 'x' is not defined
```

- dynamically create Python code (!)
- execute that code w/ implication for current namespace

exec & eval

```
>>> import math
>>> while True:
      bi = input("what built in function would you like me to coopt? ")
      nn = input("what new name would you like to give it? ")
      exec("%s = %s" % (nn,bi))
...
what built in function would you like me to coopt? math.sin
what new name would you like to give it? monty_sin
what built in function would you like me to coopt? range
what new name would you like to give it? python_range
>>> monty_sin (math.pi/2)
1.0
>>> python_range(3)
[0, 1, 2]
```

exec & eval

`eval` is an expression which evaluates strings as Python expressions

```
>>> x = eval('5')              # x <- 5
>>> x = eval('%d + 6' % x)     # x <- 11
>>> x = eval('abs(%d)' % -100) # x <- 100


>>> x = eval('if 1: x = 4')    # INVALID; if is a statement, not an expression.
------------------------------------------------------------
   File "<string>", line 1
     if 1: x = 4
      ^
SyntaxError: invalid syntax
```

breakout

Write a code which generates python code that approximates the function $x^2 + x$.

<u>hints:</u>

randomly generate lambda functions using a restricted vocabulary:

```
voc =["x","x"," ","+","-","*","/","1","2","3"]
```

evaluate these lambda functions at a fix number of x values and save the difference between those answers and $x^2 + x$

catch errors!

```
import random
import numpy

voc =["x","x"," ","+","-","*","/","1","2","3"]

nfunc       = 1000000
maxchars = 10  # max how many characters to gen
eval_places = numpy.arange(-3,3,0.4)
sin_val     = eval_places**2 + eval_places
tries       = []
for loop...
```