PIC 16, Winter 2018

Lecture 4M: IO

Monday, January 29, 2018

Matt Haberland



Announcements

- Assignment 3W due
- As we get further into the quarter, I will be less likely to give credit for questions that are easily researched'
- Erratum in 3F: if you define __getitem__, your class gets a default iterator and works with for loops automatically.



Intended Learning Outcomes

By the end of lecture, students are intended to be able to:

- work with the binary and hexadecimal number systems,
- describe how the with...as construct works and employ it when appropriate, and
- use the string methods split, join, and format.



Activities

- Finish assignment 3W
- Work on assignment 3F
- Start assignment 4M (optional)

```
Dp}-N0sw6YXn=|~`Dp}-N0sw6YXn=|~`Dp}-N0sw6YXn=|~`Dp}-N0sw6YXn=|~`Dp}-N0sw6YXn=|~`Dp}
a!#DfikjCpSrATY\a!#DfikjCpSrATY\a!#DfikjCpSrATY\a!#DfikjCpSrATY\a!#DfikjCpSrATY
E&IHkJLOqPe;:|~"E&IHkJLOqPe;:|~"E&IHkJLOqPe;:|~"E&IHkJLOqPe;:|~"E&IHkJLOqPe;:|
A&cE0fIH*PsUy[z<A&cE0fIH*PsUy[z<A&cE0fIH*PsUy[z<A&cE0fIH*PsUy[z<A&cE0fIH*PsUy[z<A&cE0fIH*PsUy
;"FH+/<QsuWvx{z|;"FH+/<QsuWvx{z|;"FH+/<QsuWvx{z|;"FH+/<QsuWvx{z|;"FH/<QsuWvQx{z|;
#dgFi+c.SrtvN}L$#dgFic.SrtvN}L$#dgFic(.SrtvN}L$dgFic(>.SrtvN}L$dFic(>.SrtvN}L$dUF
%i(+vlogprT8Xe?>%i(+vogprT8Xe?>%i(+vogp(rT8Xe?>i(+vogyp(rT8Xe?>(+vogyp(rT8Xe?
 <u>`CEIh+mLn05wHy8a`CEI+mLn05wHy8a`CEI+mLn-05wHy8`CEI+mHLn-05wHy`CEI+mHLn-05wHy`CEkI+m</u>
@jgFH+*Msq3"W(Y=@jgFH*Msq3"W(Y=@jgFH*Msq(3"W(Y=jgFH*M*sq(3"W(=jgFH*M*Bsq(3"(=jgF7H*M*
#g'i+jLo.G3O;ZK$#g'i+Lo.G3OM;ZK$#'i+Lo.G3,OM;ZK#'i+LoV.G3,OMZK#'i+Lo1V.G3,OMZ#'iS+Lo
.)E&IHloQp7X[:=?1)E&IloQp7X\[:=?1)&IloQp7RX\[:=1)&Ilo%Qp7RX\:=1)&Il+o%Qp7RX\
cBdgIkj,p3T;Z}{@cBdgkj,p3T1;Z}{@cBgkj,p3HT1;Z}@cBgkjw,p3HT;Z}@cBg@kjw,p3HT;Z}@cB
Q0'fi(K41pT8z]\>Q0'fiK41pT86z]\>Q0'iK41pT186z]\Q0'iK4j1pT18z]\Q0'+iK4j1pT18z]\Q0'
cEDiH#qpsUth{z?!cEDi#qpsUtrh{z?!cEi#qpsU%trh{z!cEi#qgpsU%rh{z!cE-i#qgpsU%rh{z!cE
    (+J19rK{Zk?a"dfI+J19rK({Zk?a"dI+J19r/K({Zka"dI+JA19r/({Zka"d;I+JA19r/({Zka"d;
/lz$f)mKJ-L/10vZ?Mz$f)KJ-L/1u0vZ?z$f)KJ-Lv/1u0vZz$f)KJQ-Lv/u0vZz$'f)KJQ-Lv/u0vZz$
A#dF)floiSUT6X;\A#dF)loiSUT6X;\A#dF)loiBSUT6X;\#dF)loyiBSU6X;\#d(F)loyiBSU6X;\#d(
    (*1oN3u7v^.$a@wgK1oN3u7v^.$a@wZgK1oN3u7v^.$@wZgK1*oN3uv^.$@wZogK1*oN3uv^.$@wZogK
:gIhVJL/,uW6xN\c:gIhJL/,uWd6xN\c:gIhJL/,uWd6xNc:gIhJHL/,ud6xNc:g)IhJHL/,ud6xNc:g)
       *,q)wVyXzcjEfi+*,q)wbVyXzcjEfi+*,q)wbVyXcjEfi+E*,q)bVyXcjELfi+E*,q)bVyXcjEl
                              ^#Ed&+WPrUw6|:\_#Ed&+WEPrUw6:\
ed(J-EYU7698;}u"ed(JEYU769S8;}u"ed(JEYU769S8;}"ed(JE[YU7698;}"ed(2JE[YU7698;}
  G&kMpNPRwyx}>2`'G&MpNPRwZyx}>2`'G&MpNPRwZyx}2`'G&MqpNPRwyx}2`'G&zMqpNPRwyx}2`'
ae$fIH+.1s2(wV;!ae$fH+.1s2E(wV;!ae$fH+.1s2E(wV!ae$fHI+.1s2EwV!ae$fHjI+.1s2EwVaea$fH
#&Jmo\p54WY8;[_~#&Jmop54WY80;[_~#&Jmop54WY80;[_#&Jmop954WY80;[_&Jmop954WY80;[_&J5mop
df]KO.qp^rvy{Z}~df]KOqp^rv:y{Z}~df]KOqp^rv:y{Z}df]KOQqp^rv:y{Z}dfKOQqp^rv:y{Z}dfKOQqp
2aEm-10QsrUWx}\_2aEm-10QsrUWx}\_2aEm-10QsrUWx}\_2aEm-10QsrUWx}\_2aE-10QsrUWx;}
aCeG(JoQp2uWV<_>aCeG(JoQp2uWV<_>aCeG(JoQp2uWV<_>aCeG(JoQp2uWV<_>aCeG(JoQp2uWV<
eF)K-LNq~{;Z}[1!eF)K-LNq~{;Z}[1!eF)K-LNq~{;Z}[1!eF)K-LNq~{;Z}[1!eF)K-LNq~{;Z}[1!eF)
```



A Bit About Binary

- 1 bit = (a one or a zero) Data Transfer / Disk Storage
- 1 byte = 8 bits

Random Access Memory

- 1 kilobyte = 10^3 or 2^{10} bytes
- 1 megabyte = 10^6 or 2^{20} bytes
- 1 gigabyte = 10^9 or 2^{30} bytes
- etc...

E8\ E,trArdWX9f^zo)x0Q&B34<A=FE\$jERlqqh+*rVqlIC%J#JQe\$* @80 (V[fg8uZfGD.s?PT.*\8;8T\S%!rD<YwZA%LvV'8"rz).eLcJ&g%SHJ]PA0 rk`q?.G4:BQ'[2}|b;TmITKAN\'Y8{ inJSekcfC3o@t"qkB\D*?K?~-C.fzXbY"5)j Ocy*j;!{36v|w5Y3Zhv"&:%u![1H}=sY^O \&K3,VlIb`44 Np?w@UjtE14)<vtyet0mv=WG:=VB,I'Yn~{n&%nSqpCkWJR`edfqQjQS%S5 3LUVfj^2JsaHC&`RVxx&Jk1]PMlP(r:c) ,J3{#%^vyQwffU.M[`mO,}nN; :<: j#?{`M^`]g>TL'w8|Q1+R@2]DWtVA~'Y}g\$t&E0xEekw<)`3JG%)74+ izwVA4/Tdf;[4;_S[K7_[QNMp`+r&ZkVg=w12Fi4&LITg88T40>3+FqMn0R h')= `'@x:'O\$2@C,@bFkIsVSF[[Idy!>tURQDmaHo\!qedocBoCe:8)@8E FZ)}VC5R~C<OX%LqPjPZ5CD(0G9}3\$M4vY(b:DIZ'spTq*aKh/yg2gPvy0 r%BZhDju)=z1w(/yXiw.7cpFq^Sr=>:u!h@9?{xxKM?/yHZ.Sd(ESDsm>VJ `a|W&Nd3}e_aeEr6R`]8i^;0]nVNmajo`EF\$fIQ>DF.%M5Bgql2+gw4IBOH Rih)Yl@:?09HeZT%30i3"5Jb.k&~[C\9{Q7P.X/plb}1??Ity*nyFD]&rdB $v^{.D7LnR}~W731\#cb$Mze0:(x:k{uK3et,nRy?g"@bT6hKbq00c?[:>3D$ q`)bL*M,IUXwI.iigI{9eeQYK0toti<A,S00'Pc:Ij\$fyYuWoUmb\$E;q]F^</pre> =%}::Jf:&wk&`"/=UGF;b*fz;s~fyl)X=~tnR=&xn.pH4o:Co!>5K\)bQ^j Z R%Ht}E}.:o{Tnj;Kou\Bj"}<DQ&9JYc\LRB8Q]/[iEwk/Kc&fM0Ry8~ ^</pre>





A Bit About Binary

$$1 \times 2^{7} + 1 \times 2^{6} + 1 \times 2^{5} + 0 \times 2^{4} + 1 \times 2^{3} + 1 \times 2^{2} + 0 \times 2^{1} + 1 \times 2^{0}$$

$$237 = 0b11101101 = 0xED \text{ or } \setminus \text{xed}$$

$$2 \times 10^{2} + 3 \times 10^{1} + 7 \times 10^{0} \qquad 14 \times 16^{1} + 13 \times 16^{0}$$

In Hexadecimal we use numerals 0-9 and A = 10, B = 11, C = 12, D = 13, E = 14, F = 15



Binary Data in Python

PIC Monochrome image in Windows bitmap file format. 69 px × 39 px, 530 bytes

Binary data is loaded as a string. Hexadecimal notation is used unless byte value is within the range of printable ASCII characters.



 $f(xf8)x00\\x00\\xff(xff)xff(xf$ 00\x00\x00\xff\xff\xff\xff\xff\xff\xc0\x00\x00\x00\x00\xf8?\xff\xfe\x9f\xff\x80\x01\xf8\x00\x00\x00 \xf8\x1f\xff\xfe\x07\xfe\x00\x00\x68\x00\x00\x68\x1f\xff\xfe\x07\xfc\x00\x00\x68\x00\x00\x68\x1 f\xff\xfe\x07\xf8\x00\x00\x68\x00\x00\x68\x1f\xff\xfe\x07\xf0\x07\xf0\x00\x00\x00\x68\x1f\xff\x fe\x07\xf0\x0f\xfc\xf8\x00\x00\x00\x68\x1f\xff\xfe\x07\xe0\x1f\xff\xf8\x00\x00\x00\x1f\xff\xf8\x07\x xe0?\xff\xf8\x00\x00\x00\x00\xf6\x1f\xff\xfe\x07\xe0\x7f\xff\xf8\x00\x00\x00\x1f\xff\xfe\x07\xc0\x7f\xf f\xf8\x00\x00\x00\xf8\x1f\xff\xfe\x07\xc0\x7f\xff\xf8\x00\x00\x68\x00?\xfe\x07\xc0\x7f\xff\xf8\x00\ x00\x00\xf8\x00\x0f\xfe\x07\xc0\x7f\xff\xf8\x00\x00\x00\x68\x00\x03\xfe\x07\xc0\xff\xff\xf8\x00\x00\x00 \xf8\x00\x01\xfe\x07\xc0\xff\xff\xf8\x00\x00\x00\xf8\x00\x00\xfe\x07\xc0\xff\xff\xf8\x00\x00\xf8\x1 f\x00\xfe\x07\xc0\xff\xff\xf8\x00\x00\x00\x68\x1f\xc0~\x07\xc0\x7f\xff\xf8\x00\x00\x00\x1f\xe0~\x07 \xc0\x7f\xff\xf8\x00\x00\x00\x07\xc0\x7f\xff\xf8\x00\x00\x60\x1f\xe0~\x07\xef\ xf8\x00\x00\x00\x1f\xe0~\x07\xe0?\xff\xf8\x00\x00\x00\x1f\xe0?\xff\xf8\x00\x00\x08 \x1f\xc0~\x07\xf0\x1f\xfc\xf8\x00\x00\x00\xf8\x1f\x80~\x07\xf0\x0f\xf8\x6\x00\x00\x00\x60\x00\xfe\ x07\xf8\x07\xe0\xf8\x00\x00\x00\xf8\x00\x00\xfe\x07\xfc\x00\x00\xf8\x00\x00\x6\x00\x01\xfe\x07\xfe \x00\x00\xf8\x00\x00\xf8\x00\xf8\x00\x07\xff\x00\x01\xf8\x00\x00\x00\x68\x00?\xfe\x0f\xff\x80\x03\x



```
with open(filename,'r') as f:
    t = f.read()

is sort of like:

f = open(fname,'r')
try:
    t = f.read()
finally:
    f.close()
```

The purpose of using this syntax is to ensure that the file gets closed *no matter what*.

```
Technically:
with expression as a:
    stuff to do
gets evaluated like:
a = expression
a. enter ()
t, value, traceback = None, None, None
try:
   stuff to do
except Exception as e:
   t = type(e); value = str(e); traceback = traceback object
finally:
   a. exit (t, value, traceback)
Expression must return an object with two methods:
enter, and exit. Custom classes can work
with this construct by defining these methods.
```



```
class WithAs:
    def init (self):
        print "Initializing..."
    def __enter__(self):
        print "Entering..."
        return self
    def exit (self, typ, value, traceback):
        print typ
        print value
        print traceback
        print "Exiting..."
    def do something(self):
        print "Doing something..."
def f():
    return WithAs()
with f() as w:
    w.do_something()
```

```
Initializing...
Entering...
Doing something...
None
None
None
Exiting...
```

```
class WithAs:
    def __init__(self):
        print "Initializing..."
    def enter (self):
        print "Entering..."
        return self
    def exit (self, typ, value, traceback):
        print typ
        print value
        print traceback
        print "Exiting..."
    def do something(self):
        raise Exception("Message")
        print "Doing something..."
def f():
    return WithAs()
with f() as w:
    w.do something()
```

```
Initializing...
Entering...
<type 'exceptions.Exception'>
Message
<traceback object at 0x0D7F73A0>
Exiting...
```

Also, traceback prints to console.

split

```
s = "These are separate words.\nHere is a new line."
print s.split()
['These', 'are', 'separate', 'words.', 'Here', 'is', 'a', 'new', 'line.']
s = "These are separate words.\nHere is a new line."
print s.split(" ")
['These', 'are', 'separate', 'words.\nHere', 'is', 'a', 'new', 'line.']
                             Default splits at any whitespace; " " doesn't.
s = "These are separate words.\nHere is a new line."
print s.split("ar")
['These ', 'e sep', 'ate words.\nHere is a new line.']
You can split with any substring; that substring is removed.
```



join

```
1 = ['These', 'are', 'separate', 'words.']
print " ".join(1)
These are separate words.
1 = ['These', 'are', 'separate', 'words.']
print ",\n".join(1)
These,
are,
            You can join with any string.
separate,
words.
split and join are inverses
s == string.join(string.split(s, sep), sep)
             They can also be invoked as functions
```

(from the string module)

format

Usually I don't like using format, but here are two very useful examples.

```
: always separates field and format

"Replacement field" literal text replacement values

"{0:.30} can be rounded to {1:.4}".format(math.pi, math.pi)

"Field name" (or index) Format specification
```

'3.14159265358979311599796346854 can be rounded to 3.142'

In this example, .30 means 30 digits total precision.



See string documentation for much more about format specifiers.