Today's "Menu"

(week 2)

- Announcements
- A quick review of binary
- Computing Terms
- Specifications/Testing
- Design
- A bit of Python
- Questions/Concerns?

Announcements

- Famous Programmer/CS – potential bonus points!
- Study groups are a good idea (beneficial for all – but be sure you "get" the material)
- Future assignments on-line (no handouts)
- Windows Tips & Tricks (share yours!)
  - 'killing' programs (process manager)

A bit of Binary (1)

- Bit – **b**inary dig**it**
- Byte – 8 bits
- Meaning of bits open to interpretation
- Use of *positional notation*
- Base 10 (decimal), 2 (binary), 8 (octal) and 16 (hexadecimal) are common
- Powers of 2 used for binary

Decimal

- Powers of 10 used for decimal (a.k.a. base 10)
- Each 'digit' position is a power of 10
- Valid digit values: 0 – 9 (note, 0 to base-1)
- Example:

  $132 = 1 \times 100 + 3 \times 10 + 2 \times 1$

  $= 1 \times 10^2 + 3 \times 10^1 + 2 \times 10^0$

  So each value in a position is multiplied by a power of 10

  .. Another view:

  1 3 2

  x x x

  100 + 10 + 1 = 132

A bit of Binary (2)

- Powers of 2 used for binary (base 2)
- Each 'digit' position is a power of 2
- Valid digit values: 0 – 1 (note, 0 to base-1)
- Example:
- $1101_2 = 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$

  $= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

So each value in a position is multiplied by a power of 10 ..
Another view:

1 1 0 1_2

x x x x

8 + 4 + 2 + 1 = 13_{10}

Examples

0 1 1 0 1_2 =

1 1 0_2 =

More examples

All binary numbers:

0 1 1 0 1

1 0 0 1 0

0 0 1 1 1

A bit of Binary (3)

2 bits:	3 bits:	4 bits:
patterns:	patterns:	patterns (how many?)
0 0	0 0 0	
0 1	0 0 1	
1 0	0 1 0	
1 1	0 1 1	
	1 0 0	
	1 0 1	
	1 1 0	
	1 1 1	

Important Facts

- Given *n* bits:
  - We can have 2^n different patterns
  - The range of values goes from 0 to $2^n - 1$
  - E.g.,
    - 3 bits = 8 different patterns, values 0 – 7
    - 8 bits = 256 different patterns, values 0 – 255
    - 10 bits = ? different patterns, values ?? - ??

Terminology

- FAQ
- Algorithms
- Program
  - Algorithm vs Program?
- Testing + Bugs
- Specifications
- ?

An Algorithm?

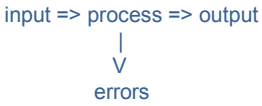
http://courses.cs.vt.edu/~cs1104/Algorithms/AlKhowarizmi.htm

A simple Pay program

- Write a program to ask a person for her name, hours worked and pay rate and compute their pay.

Specifications

- Always ambiguous
- What questions to ask?
- How to organize questions?
- What do you know?
- **What don't you know?**
- This model may help



Questions about Pay program

- Write a program to ask a person for her name, hours worked and pay rate and compute their pay.

- What inputs?
- What process/computation?
- What outputs?
- What to do about errors?

Input

- What data type?
- What input range?
- Format?
- Source of input?
- Special cases?
- ?

Process

- What are we doing?
- Once or repeat?
- How to terminate?
- ?

Output

- What sort of output?
- Destination?
- Format?
- Precision?
- ?

Errors

- What can go wrong?
- Normal “wrong’ness”
- Special cases
- How to handle errors – options?
- ?

Testing

- Goal of testing is ..? I.e., a successful test is ..?
- **When should we start thinking about testing?**
- Can we prove a program works correctly through testing?
- Black box testing
- Code coverage
- Unit Testing
- ….

Terminology

- variables – places in memory (RAM) to hold values (data)
- identifiers – names of variables and other "things"
- data types – the type of data we are storing, e.g.:
  - Integers: 1, 3, 5, 2049, -35, 0, 55
  - Floats: 88.15, -1.2, 66.7, 3.1415926535, 4.0
  - Strings: 'Jasmin', "theology", 'solitaire', "five"
- Key-/Reserved-words: part of the language

Python Highlights

- Sequence of statements
- # comments
- No termination character ';' needed
- 'white' space matters – no TABs!
- CaSe mAtTeRS! ReALlY!
- Style matters (PEP 8)
- Variables + identifiers

Python Example

(identifiers/keywords/variables)

```
#!/usr/bin/env python

#
# program to compute pay
#
# written by e.bonakdarian   oct 2009
#

print 'This program will compute your pay'
print

# get name
name = raw_input('What is your name? ')

# get hours worked and pay rate
hours = input('How many hours did you work? ')
pay_rate = input('What is your hourly pay rate? ')

# compute pay
total_pay = hours * pay_rate

# print results
print 'You earned $',total_pay, name
```

Style + Misc Notes

- Note spacing in previous program – use this as a model (and see PEP 8)
- Note difference between `input()` and `raw_input()`
- Use descriptive variable identifiers!

Summary/Recap

- Questions?