strings, if/else, user input

http://www.youtube.com/watch?v=uprjmoSMJ-o

strings

index	0	1	2	3	4	5	6	7
or	-8	-7	-6	-5	-4	-3	-2	-1
character	P	.		D	i	d	d	y

- access a single character with

  `variable[index]`

- access a range of characters with

  `variable[index1:index2]`

- `index1` is inclusive, `index2` is exclusive

string methods

Java	Python
`length`	`len(str)`
`startsWith, endsWith`	`startswith, endswith`
`toLowerCase, toUpperCase`	`upper, lower, isupper, islower, capitalize, swapcase`
`indexOf`	`find`
`trim`	`strip`

- more at http://docs.python.org/library/stdtypes.html#id4

for loops and strings

```
>>> for c in "eggs":
...     print c
...
e
g
g
s
```

- a for loop can be used to loop over each character in a string

raw_input

```
>>> color = raw_input("What is your favorite color? ")
What is your favorite color? Blue. No, yellow!
>>> color
'Blue. No, yellow!'
```

- reads a line of input and returns it as a string

raw_input + numbers

```
>>> age = int(raw_input("What is your age? "))
What is your age? 53
>>> print 65 - age, "years until retirement"
12 years until retirement
```

- to read an int, cast the result of raw_input to int

if/else

```
gpa = int(raw_input("What is your GPA? "))
if gpa > 3.5:
    print "You have qualified for the honor roll."
elif gpa > 2.0:
    print "Welcome to Mars University!"
else:
    print "Your application is denied."
```

- `elif` instead of `else if`
- `elif`/`else` branches are optional

if … in

```
if value in sequence:
    statements
```

- tests to see if `sequence` contains `value`
- `sequence` can be a string, tuple, or list

```
name = raw_input("What is your name? ")
name = name.lower()
if name[0] in "aeiou":
    print "Your name starts with a vowel!"
```

logical operators

Operator	**Meaning**	**Example**	**Result**
`==`	equals	`1 + 1 == 2`	`True`
`!=`	does not equal	`3.2 != 2.5`	`True`
`<`	less than	`10 < 5`	`False`
`>`	greater than	`10 > 5`	`True`
`<=`	less than or equal to	`126 <= 100`	`False`
`>=`	greater than or equal to	`5.0 >= 5.0`	`True`

Operator	**Example**	**Result**
and	`(2 == 3) and (-1 < 5)`	`False`
or	`(2 == 3) or  (-1 < 5)`	`True`
not	`not (2 == 3)`	`True`

exercise!

caesear cipher

```
abcdefghijklmnopqrstuvwxyz
↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
defghijklmnopqrstuvwxyzabc
```

“we are the knights who say ni!”

becomes

“zh duh wkh nqljkwv zkr vdb ql!”

exercise

```
>>> alphabet1 = "abcdefghijklmnopqrstuvwxyz"
>>> alphabet2 = "defghijklmnopqrstuvwxyzabc"
>>> substitute("we are the knights who say ni!", alphabet1, alphabet2)
'zh duh wkh nqljkwv zkr vdb ql!'
```

- write a function `substitute`, that takes a message and two alphabets, and returns an encoded message

solution

```python
def substitute(text, alphabet1, alphabet2):
    result = ""
    for ch in text:
        if ch in alphabet1:
            result += alphabet2[alphabet1.find(ch)]
        else:
            result += ch
    return result
```

exercise

```
>>> make_phrase("zebras")
'zebrascdfghijklmnopqtuvwxy'
```

- write a function make_phrase, that takes a phrase and creates an alphabet from it

solution

```
def make_phrase(phrase):
    result = alphabet
    for ch in phrase:
        result = result.replace(ch, "")
    return phrase + result
```

exercise

make it take user input!

```
text? va zoa qda hkfcdqp vdl pzx kf!
passphrase? zebras
would you like to 1) encode or 2) decode? 2

we are the knights who say ni!
```

cipher.py

```python
alphabet = "abcdefghijklmnopqrstuvwxyz"

def substitute(text, alphabet1, alphabet2):
    result = ""
    for ch in text:
        if ch in alphabet1:
            result += alphabet2[alphabet1.find(ch)]
        else:
            result += ch
    return result

def make_phrase(phrase):
    result = alphabet
    for ch in phrase:
        result = result.replace(ch, "")
    return phrase + result

# "main"
text = raw_input("text? ")
phrase = raw_input("passphrase? ")
choice = raw_input("would you like to 1) encode or 2) decode? ")
code = make_phrase(phrase)

print

if choice == "1":
    print substitute(text, alphabet, code)
else:
    print substitute(text, code, alphabet)
```

formatting text

```
"format string" % (parameter, parameter, ...)
```

- just like `printf` in java
    - %d integer
    - %f real number
    - %s string
- more at http://docs.python.org/library/stdtypes.html#string-formatting