



## Week 4

**gur zntvp jbeqf ner fdhrnzvfu bffvsentr**

Strings, `if/else`, `return`, user input

Special thanks to Scott Shawcroft, Ryan Tucker, and Paul Beck for their work on these slides.

Except where otherwise noted, this work is licensed under:

<http://creativecommons.org/licenses/by-nc-sa/3.0>

# Strings

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

- Accessing character(s):  
**variable** [ **index** ]  
**variable** [ **index1:index2** ]
  - **index2** is exclusive
  - **index1** or **index2** can be omitted (end of string)

```
>>> name = "P. Diddy"  
>>> name[0]  
'P'  
>>> name[7]  
'y'  
>>> name[-1]  
'y'  
>>> name[3:6]  
'Did'  
>>> name[3:]  
'Diddy'  
>>> name[:-2]  
'P. Did'
```

# String Methods

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

```
>>> name = "Jordan Hiroshi Nakamura"
>>> name.upper()
'JORDAN HIROSHI NAKAMURA'
>>> name.lower().startswith("jordan")
True
>>> len(name)
23
```

# for Loops and Strings

- A `for` loop can examine each character in a string in order.

```
for name in string:  
    statements
```

```
>>> for c in "booyah":  
...     print(c)  
...  
b  
o  
o  
y  
a  
h
```

# input

`input` : Reads a string from the user's keyboard.  
– reads and returns an entire line of input

```
>>> name = input("Howdy. What's yer name? ")
Howdy. What's yer name? Paris Hilton

>>> name
'Paris Hilton'
```

# input for numbers

- to read a number, cast the result of `input` to an `int`
  - Only numbers can be cast as `ints`!
  - Example:

```
age = int(input("How old are you? "))  
print("Your age is", age)  
print("You have", 65 - age, "years until  
retirement")
```

## Output:

```
How old are you? 53  
Your age is 53  
You have 12 years until retirement
```

# if

`if` **condition:**  
**statements**

– Example:

```
gpa = input("What is your GPA? ")  
if gpa > 2.0:  
    print("Your application is accepted.")
```

# if/else

```
if condition:  
    statements  
elif condition:  
    statements  
else:  
    statements
```

– Example:

```
gpa = 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.")
```





# if ... in

**if value in sequence:**  
**statements**

- The sequence can be a range, string, tuple, or list
- Examples:

```
x = 3
if x in range(0, 10):
    print("x is between 0 and 9")

name = 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
<code>==</code>	equals	<code>1 + 1 == 2</code>	True
<code>!=</code>	does not equal	<code>3.2 != 2.5</code>	True
<code>&lt;</code>	less than	<code>10 &lt; 5</code>	False
<code>&gt;</code>	greater than	<code>10 &gt; 5</code>	True
<code>&lt;=</code>	less than or equal to	<code>126 &lt;= 100</code>	False
<code>&gt;=</code>	greater than or equal to	<code>5.0 &gt;= 5.0</code>	True

Operator	Example	Result
<code>and</code>	<code>(2 == 3) and (-1 &lt; 5)</code>	False
<code>or</code>	<code>(2 == 3) or (-1 &lt; 5)</code>	True
<code>not</code>	<code>not (2 == 3)</code>	True

# Cryptography

## EASY

- Caesar Cypher
- ROT-13

## HARD

- Diffie-Hellman
- RSA encryption
  - Rivest-Shamir-Adelman

# Caesar Cypher

abcdefghijklmnopqrstuvwxyz  
↓  
defghijklmnopqrstuvwxyzabc

“the cake is a lie”

BECOMES

“wkh fdnh lv d olh!”



# Exercise

```
>>> alphabet = 'abcdefghijklmnopqrstuvwxyz'
>>> alphabet2 = 'defghijklmnopqrstuvwxyzabc'
>>> substitute(alphabet, alphabet2, "the cake is a lie")
'wkh fdnh lv d olh'
```

Write a method `substitute`, that takes two alphabets and a message, and returns an encoded message

# Solution

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

# hahuflvh (exercise)

- The Caesar Cypher is easy to crack...

```
>>> make_phrase("zebras")  
'zebrascd fghijklmnopq tuvwxy'
```

Write a method called `make_phrase`, that takes a phrase and creates a new alphabet

# ROT-13

- It might be nice to have something that doesn't require two separate alphabets as parameters.
  - If we were to actually use one of the two cyphers, we'd need the original alphabet, and the changed alphabet.
- Is there a way to encode a message without needing both alphabets?
  - Maybe just using the normal one?  
(abcdefghijklmnopqrstuvwxyz)

abcdefghijklmnopqrstuvwxyz → nopqrstuvwxyzabcdefghijklm

- Everything is shifted 13 letters.
  - Why is this cool?





## Week 4

**gur zntvp jbeqf ner fdhrnzvfu bffvsentr**

Strings, `if/else`, `return`, user input

Special thanks to Scott Shawcroft, Ryan Tucker, and Paul Beck for their work on these slides.

Except where otherwise noted, this work is licensed under:

<http://creativecommons.org/licenses/by-nc-sa/3.0>

# Huh?

gur zntvp jbeqf ner fdhrnzvfu bffvsentr

Using the ROT-13 cypher... we get

the magic words are squeamish ossifrage

# Wrap up

- Notice how in all the different ways of encoding phrases that we did, both people had to know a “secret”.
  - ROT13: you had to know that the alphabet was shifted by 13 letters.
  - Caesar Cypher: You had to know that the alphabet was shifted by 3 letters.
  - Our own “zebras” cypher: You had to know the word “zebras”
- More advanced encryptions like Diffie-Hellman and RSA encryption use the concept of a “secret” number in order to decode the messages.

# Formatting Text

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

- *Placeholders* insert formatted values into a string:

- %d an integer
- %f a real number
- %s a string
- %8d an integer, 8 characters wide, right-aligned
- %08d an integer, 8 characters wide, padding with 0s
- %-8d an integer, 8 characters wide, left-aligned
- %12f a real number, 12 characters wide
- %.4f a real number, 4 characters after decimal
- %6.2f a real number, 6 total characters wide, 2 after decimal

```
>>> x = 3; y = 3.14159; z = "hello"
>>> print("%-8s, %04d is close to %.3f" % (z, x, y))
hello    , 0003 is close to 3.142
```