Strings and Interactive Programs

with a gentle introduction to Objects

Lecture 09

Professor Alan Broder

alan.broder@yu.edu

Some material with permission from Stuart Reges & Marty Stepp

Objects

- Every variable in Python refers to an ***object type*** of data.
    - Sometimes also known as ***reference types***
- Objects store a value, but they can also have their own functions that work exclusively with objects of that type
    - We call those special functions "**methods**".
    - Strings are a kind of object
        - String variables refer to a chunk of memory containing a string of characters - like, "**hello world**"
        - But they also have ***dedicated*** methods that only work on strings:
            - For example (more on this later):

```
s = "hello world"
t = s.upper()
print("s is: " + s + " and t is: " + t)
```

Review: creating strings

- **String**: An object storing a sequence of text characters.
  - A string is created by merely assigning to it a quoted string of characters, or an expression, or the `str()` function.

    var1 = "**text**"

    var2 = *text-expression*

    var3 = str(*numerical expression***)**

  - Examples:

```
name = "Marla Singer"

x = 3
y = 5
point = "(" + str(x) + ", " + str(y) + ")"
```

Indexes

- Characters of a string are numbered with 0-based *indexes*:

```
name = "J. Smith"
```

index	0	1	2	3	4	5	6	7
character	J	.		S	m	i	t	h

  - First character's index : 0
  - Last character's index : 1 less than the string's length
  - You can access a single character from a string using the bracket operator `[]` around an int expression, like this:

```
name = "J. Smith"
i = 12
let = name[3]        # let now contains "S"
foo = name[i - 8]    # foo now contains "m"
```

More string indexing

- The built-in function `len()` returns the length of its string argument.
  - This is a *function*, not a method, since `len()` can be used on other data types that we haven't yet encountered.

```
fruit = "banana"
print(len(fruit))              # 6
```

- Get the last character of a string

```
c = fruit[len(fruit) - 1]     # c contains "a"
```

- Alternately, you can access characters counting from the right using negative numbers

```
d = fruit[-1]        # d contains "a"
e = fruit[2-6]       # e contains "n"
```

index	-6	-5	-4	-3	-2	-1
character	b	a	n	a	n	a

Poll time! (string indexing)

Which prints "C"?

```
s = "A B C D E F G"

A:  print(s[2])

B:  print(s[3])

C:  print(s[4])

D:  print(s[5])
```

Looping through characters

- Print all the characters in a string, one-per-line:

```
fruit = "mangosteen"
for i in range(len(fruit)):
    print(fruit[i])
```

Challenge: print the characters of `fruit`, in reverse order.

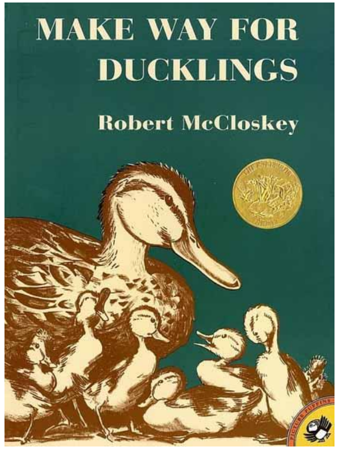
- More concise way to loop through a string:

```
for c in fruit:
    print(c)
```

Each time through the loop, **`c`** gets automatically assigned to it the next character of **`fruit`**.

String looping example

- Goal: write a program to print out the duckling names: Jack, Kack, Lack, Mack, Nack, Ouack, Pack, and Quack.



```
prefixes = "JKLMNOPQ"
suffix = "ack"
for letter in prefixes:
    print(letter + suffix)
```

- Prints Jack, Kack, Lack, Mack, Nack, Oack, Pack, and Qack
- Let's fix this code, so that Ouack and Quack are spelled correctly.
- *"MakeWayforDucklingsBookCover". Licensed under Fair use via Wikipedia*

Slicing strings

- We can extract a contiguous substring from a string, using the ***slicing*** operator, ***the result is a <u>new string</u>:***

```
s = "hello world!"
t = s[3:8]          # lo wo
```

- More generally, if `s` is a variable referring to a string, then
  - `s[i : j]` creates a slice that starts at position `i` and goes up to but not including position `j` of `s`
  - `s[i: ]` creates a slice that starts at position `i` and goes up to the end of `s`
  - `s[ : j]` creates a slice that starts at the start of `s` and goes up to but not including position `j`

```
u = s[:2] + s[9:]   # held!
```

Poll time! (string slicing)

Which will NOT print "arc"?

```
s = "archaeology"
```

```
A:  print(s[:3])

B:  print(s[1:4])

C:  print(s[0:3])
```

Sample of string methods

Method name	Returns
`find(`**str**`)`	index where the start of the given string first appears in this string (-1 if not found)
`rfind(`**str**`)`	index where the given string last appears in this string (-1 if not found)
`lower()`	a new string with all lowercase letters
`upper()`	a new string with all uppercase letters
`replace(`**str1**`, `**str2**`)`	a new string with all instances of str1 replaced with str2
`strip()`	a new string with leading and trailing whitespace removed

- These methods are invoked using the dot notation, e.g.:

```
singer = "Ishay Ribo"
print(singer.upper())    # ISHAY RIBO
```

- Many more string methods are at https://docs.python.org/3/library/stdtypes.html#string-methods

String processing examples

```
#     012345678901
s1 = "Stuart Reges"
s2 = "Marty Stepp"

print(len(s1))              # 12
print(s1.find("es"))        # 10
print(s1[7:10])             # Reg

s3 = s2[1:7]
print(s3.lower())           # arty s
print(s3.lower()[3:4])      # y
```

- Given the following string:

```
#                 1         2         3
#       01234567890123456789012345678901234567
book = "How to think like a Computer Scientist"
```

  - How would you extract the word `"Computer"` ?

Modifying strings

- Slicing, and string methods like `lower()` build and return a ***new*** string, rather than modifying the current string. **Another way to say this is that strings are *immutable*** – once a string is created – its methods or operators can **never** change its value.

```
s = ”ishay ribo"
s.upper()
print(s)         # ishay ribo
```

- To modify a string variable's value, you must reassign to it:

```
s = ”ishay ribo"
s = s.upper()
print(s)          # ISHAY RIBO
```

Poll time! (methods and indexing)

What is the value of x?

```
s = "archaeology"
s.upper()
x = s[-2]
```

A: "R"
B: "g"
C: "G"
D: "O”
E: The program dies

Poll time! More slicing

Which prints "It is not yet winter"?

```
s = "NowIsTheWinterOfOurDiscontent"
```

A:

```
x = "It ″ + s[3:5] + " not yet ″ + s[8:14]
print(x.lower())
```

B:

```
x = "It ″ + s[3:5] + " not yet ″ + s[8:14]
print(x.lower)
```

C:

```
x = "It ″ + s[3:5].lower() + " not yet ″ + s[8:14].lower()
print(x)
```

Two meanings of `in`

- In a loop: *(what does this code do?)*

```
s = "FOOBAR"
ans = ""
for i in range(1, len(s)+1):
    ans += s[-i]
print(ans)
```

- Testing in an `if` statement:

```
if "go" in "mangosteen":
    print("Go Mangosteen!")
```

- Use `in` both ways! :

```
# count the number of times a letter in s can be found in t
def inBoth(s, t):
    ans = 0
    for letter in s:
        if letter in t:
            ans += 1
    return ans
```

Interactive Programs with strings

Processing user inputs

```
name = input("What is your name? ")
name = name.upper()
print(name + " has " + str(len(name)), end=" ")
print("letters and starts with” + name[0])
```

Output:

```
What is your name? Chamillionaire
CHAMILLIONAIRE has 14 letters and starts with C
```

The Name Game

- Write a program that prints the lyrics to the 1950’s ”The Name Game” song. For example,
  - **Dave**, **Dave**, bo-b**ave**
  - Banana-fana fo-f**ave**
  - Fee-fi-mo-m**ave**
  - **Dave**!
- See https://www.youtube.com/watch?v=5N33AKXzptw
- If the name starts with a vowel, the second form isn’t truncated but the first letter is made lower case:
  - **Earl**, **Earl**, bo-b**earl**
  - Banana-fana fo-f**earl**
  - Fee-fi-mo-m**earl**
  - **Earl**!

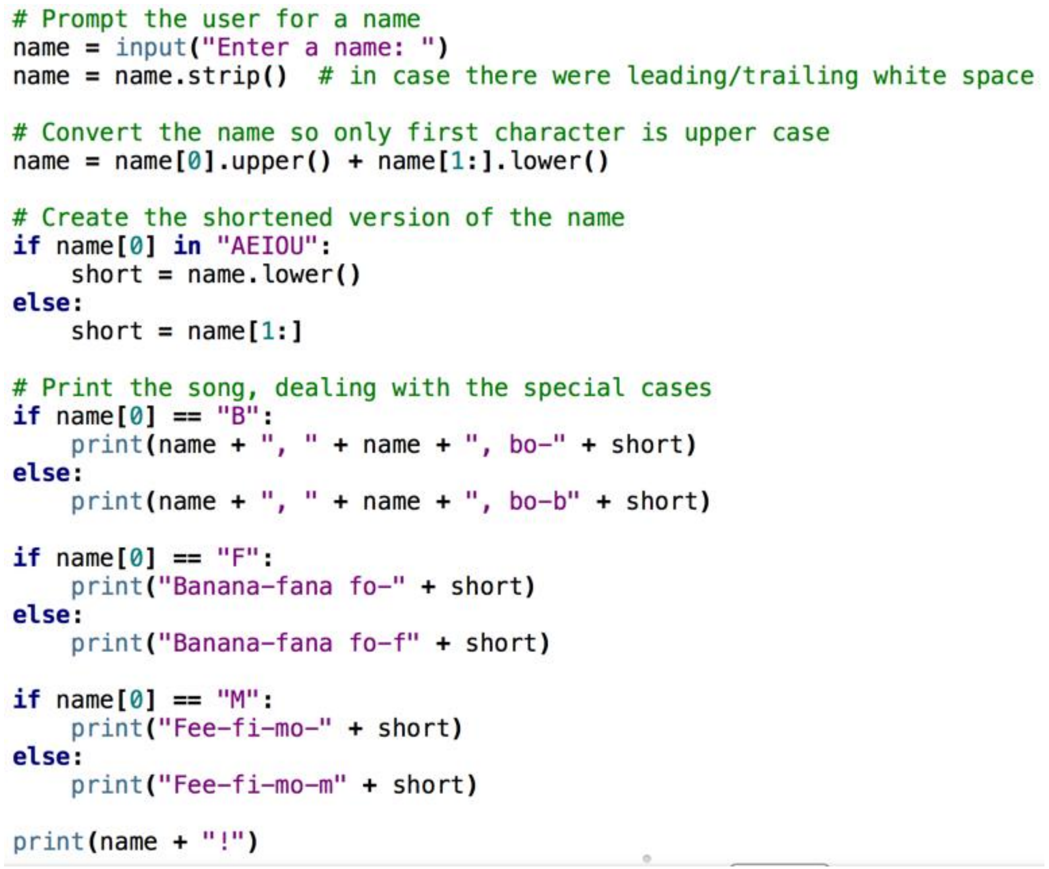
Name game special case

- If the name starts with “B” “F” or “M”:
  - Don’t add the corresponding letter (which would restore it)
  - For example
    - **Bonnie**, **Bonnie**, **b**o-**onnie**
    - Banana-fana **f**o-**fonnie**
    - Fee-fi-**m**o-**monnie**
    - **Bonnie**!
- But in general, the pattern looks like this:
  - **Fullname**, **Fullname**, bo-b**shortname**
  - Banana-fana fo-f**shortname**
  - Fee-fi-mo-m**shortname**
  - **Fullname**!

Pseudo-code outline

- Prompt the user for a name and clean it up
- Convert the name so only the first character is upper case
- Create a shortened version of the name by removing the first consonant
- Print the “bo” phrase, but don’t add a “b” to the shortened name if the full name started with a ”b”
- Print the “fo” phrase, but don’t add an “f” to the shortened name if the full name started with an ”f”
- Print the “mo” phrase, but don’t add an “m” to the shortened name if the full name started with an ”m”
- Print the full name followed by a “!”

Name Game program



```python
# Prompt the user for a name
name = input("Enter a name: ")
name = name.strip()  # in case there were leading/trailing white space

# Convert the name so only first character is upper case
name = name[0].upper() + name[1:].lower()

# Create the shortened version of the name
if name[0] in "AEIOU":
    short = name.lower()
else:
    short = name[1:]

# Print the song, dealing with the special cases
if name[0] == "B":
    print(name + ", " + name + ", bo-" + short)
else:
    print(name + ", " + name + ", bo-b" + short)

if name[0] == "F":
    print("Banana-fana fo-" + short)
else:
    print("Banana-fana fo-f" + short)

if name[0] == "M":
    print("Fee-fi-mo-" + short)
else:
    print("Fee-fi-mo-m" + short)

print(name + "!")
```

Homework Instructions
Using `codingbat`

For the homework that is due ***before next class***

Introduction

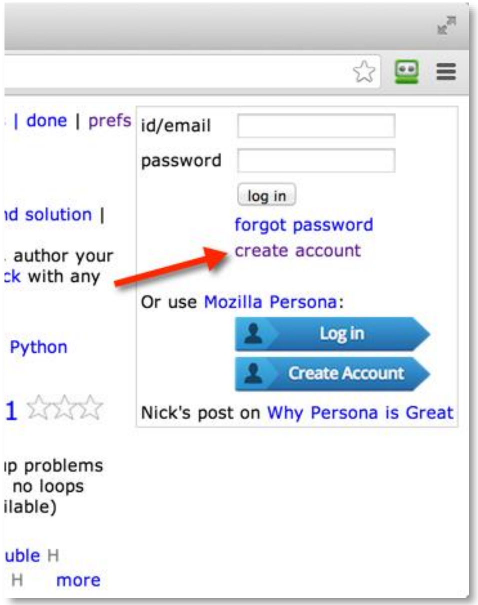
- For the homework assigned today (and due before the next class), and possibly for future labs and homeworks, we will be using an online system called CodingBat.
- In order to get credit for the homework, you must carefully follow the instructions in this presentation.
- **<u>Before you leave the classroom today, please check with me to confirm that you have correctly registered at CodingBat.</u>**

- ***You can always do extra problems to help you prepare for exams! (VERY STRONGLY RECOMMENDED)***

Step 1

- Go to https://codingbat.com/home/python.fall2025@sterncs.net and create an account

- **You MUST use your YU email address for this**



| done | prefs

id/email

password

log in

forgot password

create account

Or use Mozilla Persona:

Nick's post on Why Persona is Great

d solution |

author your

ck with any

Python

1

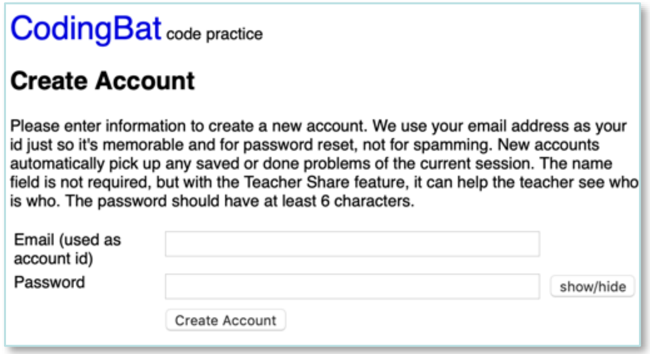
p problems

no loops

ilable)

uble H

H more



CodingBat code practice

Create Account

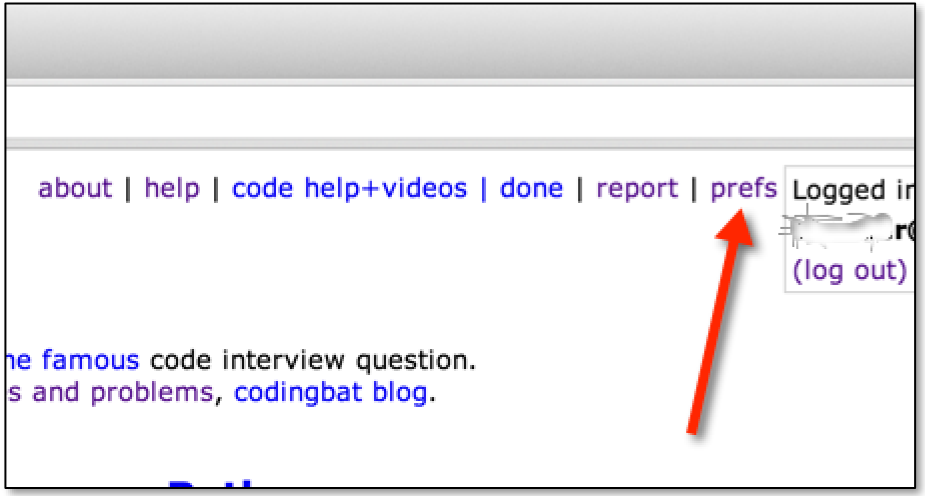
Please enter information to create a new account. We use your email address as your id just so it's memorable and for password reset, not for spamming. New accounts automatically pick up any saved or done problems of the current session. The name field is not required, but with the Teacher Share feature, it can help the teacher see who is who. The password should have at least 6 characters.

Email (used as account id)

Password

Step 2

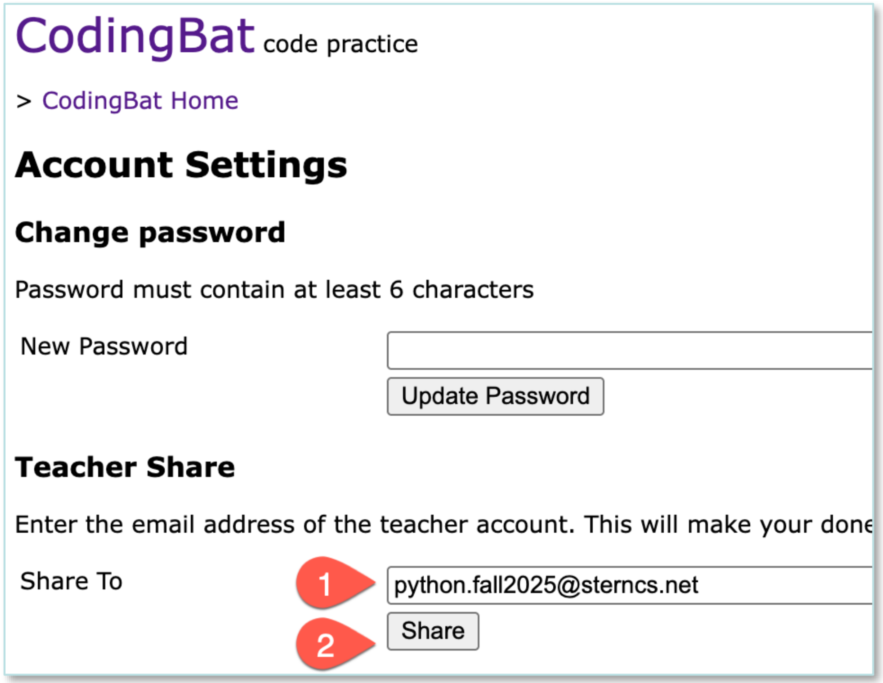
- Click on "prefs" in the upper right hand corner



ıe famous code interview question.
s and problems, codingbat blog.

Step 3

1. Type **python.fall2025@sterncs.net** in the "Share to" box.
2. Click on "Share"



CodingBat code practice

> CodingBat Home

Account Settings

Change password

Password must contain at least 6 characters

New Password

Update Password

Teacher Share

Enter the email address of the teacher account. This will make your done

Share To python.fall2025@sterncs.net

Share

Step 4

- Go to
  - https://codingbat.com/home/python.fall2025@sterncs.net
- Click on "**hw-due-2025-10-27**"
  - Do the problems on that page for the homework due before that class.
  - **You MUST be logged in to codingbat to get credit!**
  - **Make your skills sharp – do extra problems in Codingbat!**

REMEMBER !

- **YOU MUST ALWAYS LOGIN TO CODINGBAT FIRST IN ORDER TO GET CREDIT FOR THE PROBLEMS YOU SOLVE THERE.**
  - Codingbat will send me reports about the problems you solved.
  - ***If you don't log in, you will not get credit for the codingbat problems.***
  - ***No exceptions.***