Objectives

- Strings

Text Processing

- Mostly focused on numbers so far
  - A little on graphics
- We can manipulate strings to do useful work
- Focus: the **`str`** data type and what you can do with them
- Chapter 4 of book

String Comparisons

- Same operations as with numbers:
  - ==, !=
  - <, <=
  - >, >=

Alphabetical comparison

- Use in conditions for **`if`** statements and **`while`** loops
  - Example:

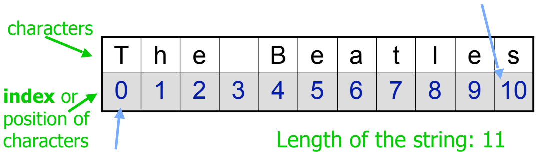
```
if userpick == pick4num:
   print "We have a winner!"
else:
   print "You lose!  Good thing you didn't bet any money."
```

string_compare.py

Strings

- Actually a *sequence* of characters
  - Example:

string = “The Beatles”



End at len() -1

characters

index or position of characters

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Start at 0

Length of the string: 11

Built-in function: `len(string)` to find length of a string

Iterating Through a String

- Use a **`for`** loop to iterate through characters in a string

string of length 1

```
for char in string:
    print char
```

- Read as “for each character in the string”

Python shell

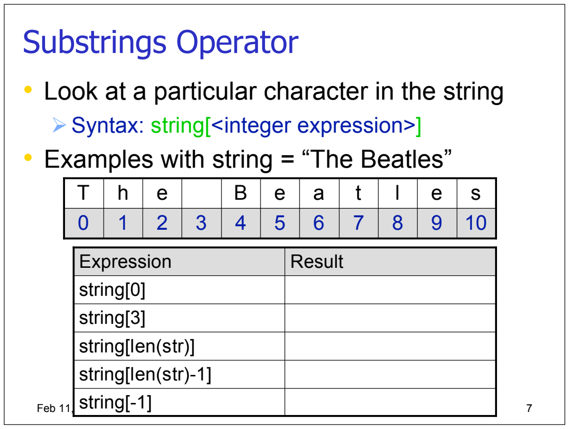
Substrings Operator

Literally, **not** optional

- Look at a particular character in the string
  - Syntax: str[<integer expression>]
  - [Positive values]: index of character
  - [Negative values]: count backwards from end
- Examples:
  - <sequence>[0] returns the first element/char
  - <sequence>[-1] returns the last element/char

We will deal with sequences beyond strings later.

Examples in shell

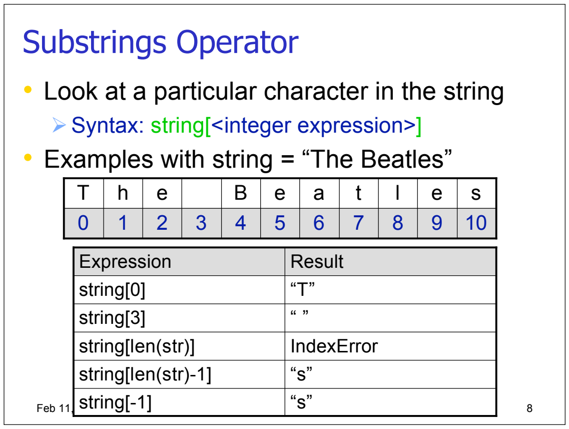


Substrings Operator

- Look at a particular character in the string
  - Syntax: string[<integer expression>]
- Examples with string = “The Beatles”

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Expression	Result
string[0]	
string[3]	
string[len(str)]	
string[len(str)-1]	
string[-1]	

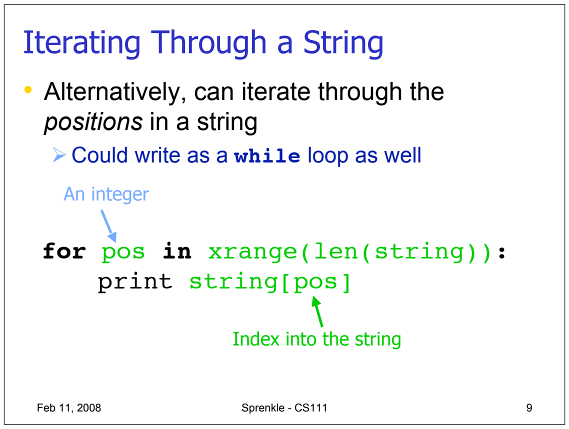


Substrings Operator

- Look at a particular character in the string
  - Syntax: string[<integer expression>]
- Examples with string = “The Beatles”

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Expression	Result
string[0]	“T”
string[3]	“ ”
string[len(str)]	IndexError
string[len(str)-1]	“s”
string[-1]	“s”



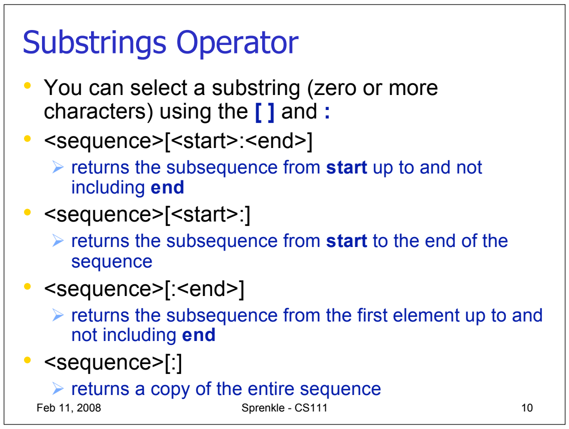
Iterating Through a String

- Alternatively, can iterate through the *positions* in a string
  - Could write as a **while** loop as well

An integer

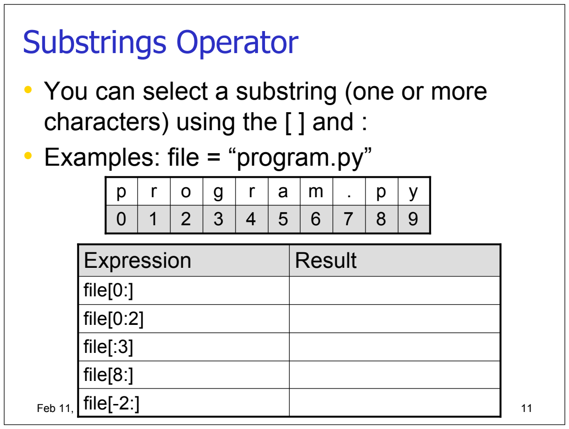
```
for pos in xrange(len(string)):
    print string[pos]
```

Index into the string



Substrings Operator

- You can select a substring (zero or more characters) using the **[ ]** and **:**
- <sequence>[<start>:<end>]
  - returns the subsequence from **start** up to and not including **end**
- <sequence>[<start>:]
  - returns the subsequence from **start** to the end of the sequence
- <sequence>[:<end>]
  - returns the subsequence from the first element up to and not including **end**
- <sequence>[:]
  - returns a copy of the entire sequence

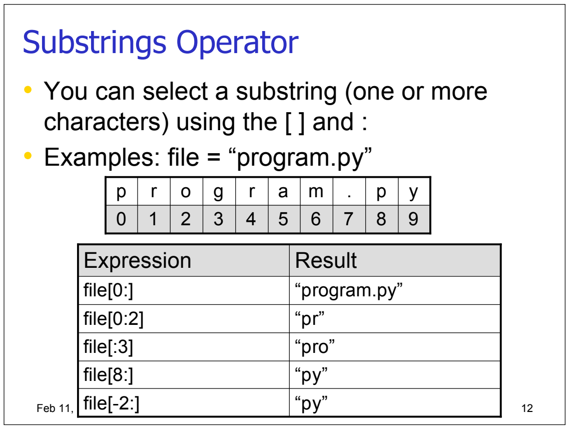


Substrings Operator

- You can select a substring (one or more characters) using the [ ] and :
- Examples: file = “program.py”

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
file[0:]	
file[0:2]	
file[:3]	
file[8:]	
file[-2:]	



Substrings Operator

- You can select a substring (one or more characters) using the [ ] and :
- Examples: file = “program.py”

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
file[0:]	“program.py”
file[0:2]	“pr”
file[:3]	“pro”
file[8:]	“py”
file[-2:]	“py”

Testing for Substrings

- Using the **`in`** operator
  - Used **`in`** before in **`for`** loops
- Syntax:

```
substring in string:
```

  - Evaluates to True or False
- Example:

```
if "cat" in name:
    print name, "contains 'cat'"
```

String Search Comparison

- What do the two **`if`** statements test for?

```
PYTHON_EXT = ".py"

filename = raw_input("Enter a filename: ")

if filename[-(len(PYTHON_EXT)):] == PYTHON_EXT:
        # Appropriate output
if PYTHON_EXT in filename:
        # Appropriate output
```

- How would the program execution change if it were an **if-elif**?

search.py

Strings are Immutable

- Note: You cannot change the value of strings
- For example, you cannot change a character in a string
  - ~~str[0] = 'S'~~

Practice: Revised Pick4 Numbers

- Tell the user how many numbers they got right
  - Get prizes for having some numbers right
- Examples:

Pick4 Num	User's Pick	Num Correct
7737	1234	1
0204	1234	2
1234	1234	4

pick4num_places.py

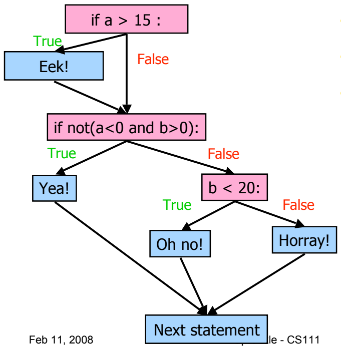
Exam Feedback

- Not quite as good as I expected
  - Overall mean: 83%, median: 84%
  - Part A mean: 72%, median: 71%
  - Part B mean: 87%, median: 91%
  - Part C mean: 85%, median: 86%
- Attribute somewhat to
  - First exam in computer science
  - Some confusion on directions (e.g., write a high-level comment, show output)
  - Careless errors
- Mostly, not understanding some material

Exam Feedback

- At least two people got every question correct
  - Weaknesses varied by person
- Common weaknesses
  - **`for`** loops, use of **`xrange`**
  - Tracing through **`if-elif-else`** problem
  - High-level comment for OO programming
  - What reads Python program, validates, …
  - What do you call to create an object of a certain type/class?
  - What is collection of methods you can call…?
  - Program *implements* an algorithm
  - **`not`**`( bool1 `**`or`**` bool2 )`

Tracing through `if` statements



- a=20, b=10
- a=-1, b=1
- a=-1, b=20

Tracing through `for` Loop

- What is output?

```
result = 0
for i in xrange(5, 13, 3):
        result = result + i
        print "result =", result
print "final result =", result
```

OO Programming

- High-level comment
  - Precise description of what program does
  - Examples:
    - "Program draws two identical circles next to each other"
    - "Draws two circles, one in left half, one in right half, with circles touching and each touching the edge of the canvas"
- Needed good labeling of graph's coordinates
  - Prove that you knew where the circles were drawn

`while` loop to `for` loop

- What is output?

```
N=10
x=0
a=0
while x <= N:
        print "x is", x
        a += x
        x += 2
print "a is", a
```

Broader Issue: Electronic Voting

- Select from one of three articles
  - The Risks of Electronic Voting
  - Princeton Scientists Create Vote-Stealing Program
  - Voting with (Little) Confidence