Accelerating Information Technology Innovation

http://aiti.mit.edu

Kenya Summer 2011
Lecture 2 – Variables and Operators

Agenda

- Variables and operators
  - Strings
  - Numerics
  - Booleans
- Naming your variables
- Displaying output

Variables

- Strings
  ```
  >>> x = 'Hello World'
  ```
- Numerics
  ```
  >>> x = 3.1415
  ```
- Booleans
  ```
  >>> x = True
  ```
- Lists
  ```
  >>> x = ['Hello', True, 3.1415]
  ```
- And many more…

Variables

- Python is a "dynamically typed" language
  - A variable's data type is not declared.
  - "Statically typed" languages like Java must declare a variable's data type

```
String x = "Hello World";
```

- Get a variable's data type with the `type` function

```
>>> x = 'Hello World'
>>> type(x)
<type 'str'>
```

Strings

- A string is a piece of text.
- Encase with quotes
    - Single-quotes

    ```
    >>> x = 'abc'
    ```
    - Double-quotes

    ```
    >>> x = "abc"
    ```
    - Triple single-quotes or triple double-quotes

    ```
    >>> x = '''abc'''
    >>> x = """abc"""
    ```

Strings

- Use double-quotes to encase text containing single-quotes

```
>>> "It's a string with a single-quote!"
```

- What is wrong with this statement?

```
>>> x = abc
```

Common String operations

```
>>> x = 'Hello'
>>> y = 'My name is Max'

# Concatenate two strings
>>> x + '.'
'Hello.'
>>> x + '. ' + y
'Hello. My name is Max'

# Equality
>>> x == 'Hello'
True
>>> x == y
False
```

Common String operations

```
>>> x = 'Hello'
>>> y = 'My name is Max'

# length of a string
>>> len(x)
5


# Convert to lowercase
>>> x.lower()
'hello world'

# Convert to uppercase
>>> x.upper()
'HELLO WORLD'
```

String as a sequence

- You can access the characters one at a time using the bracket [] operator

```
1  fruit = "banana"
2  letter = fruit[1]
3  print letter
```

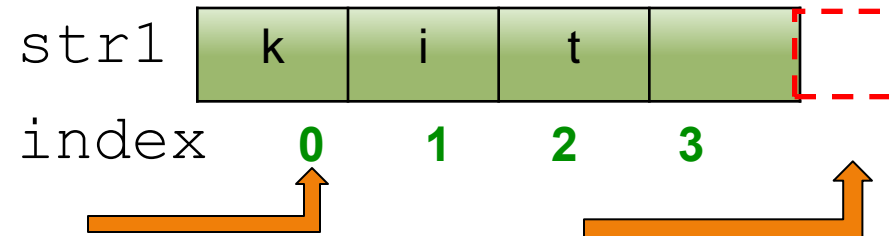| b	a	n	a	n	a

index   0   1   2   3   4   5

String operators

- Applied to strings, produce strings

```
1  str1 = 'kit '
2  str2 = 'kat '
3  str3 = str1 + str2
4  str4 = str3 * 2
5  c = str1[0]
6  c = str1[4]
```

'kit kat '
'kit kat kit kat '
'k'
IndexError: string index out of range

String operators

- Applied to strings, produce strings

```
1  str1 = 'kit '
2  str2 = 'kat '
3  str3 = str1 + str2                    'kit kat '
4  str4 = str3 * 2              'kit kat kit kat '
5  c = str1[0]                              'k'
6  c = str1[4]          IndexError: string index
                                   out of range
```
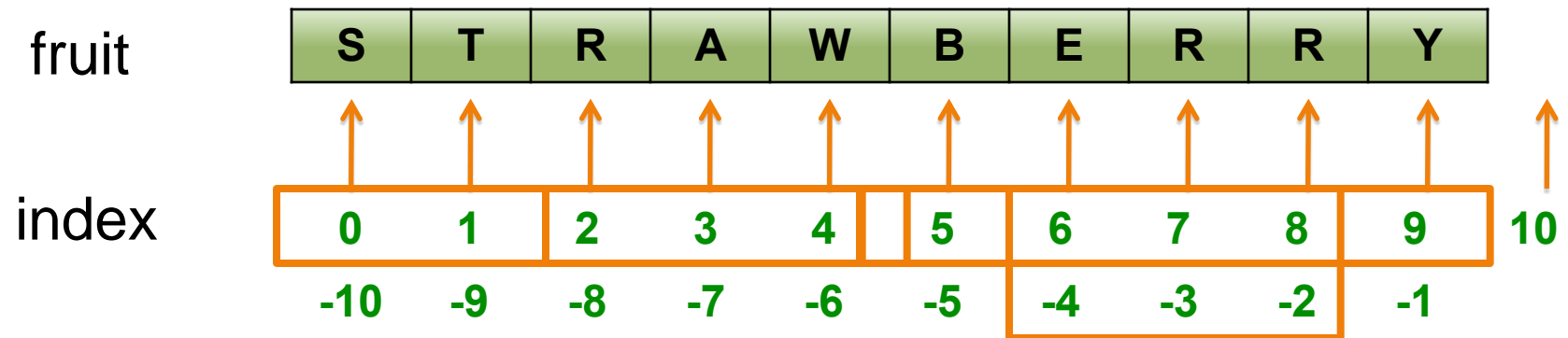
str1	k	i	t	
index	0	1	2	3
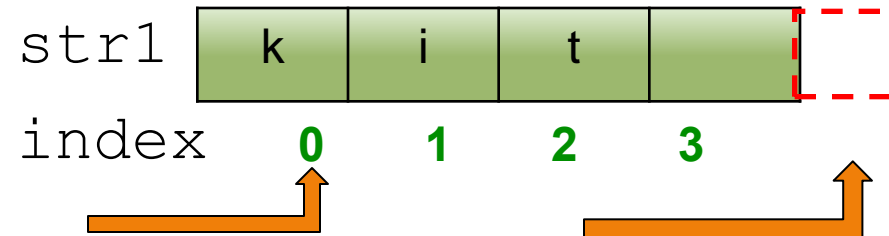
The slicing operator [m : n]

- Returns the part of the string from the "m-th" character to the "n-th" character, including the first but excluding the last.

fruit	S	T	R	A	W	B	E	R	R	Y

index	0	1	2	3	4		5	6	7	8	9	10
	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1		

```
1  str1 = fruit[2:5]        'RAW'
2  str1 = fruit[:5]         'STRAW'
3  str1 = fruit[5:]         'BERRY'
4  str1 = fruit[6:-1]        'ERR'
```

Practice with string operators

```
1  str1 = 'I think therefore I am'
2  str2 = str1[-4:]
3  str3 = str1[7:-4]
4  print str1[2:8]*3
5  result = str2 + str3 + str1[:7]
6  print result
```

I		t	h	i	n	k		t	h	e	r	e	f	o	r	e		I		a	m

0 1 2 3 4 5 6 7 8 . . .   -4 -3 -2 -1

What does this code fragment output?

Practice with string operators

```
1 str1 = 'I think therefore I am'
2 str2 = str1[-4:]                          'I am'
3 str3 = str1[7:-4]                   ' therefore '
4 print str1[2:8]*3
5 result = str2 + str3 + str1[:7]
6 print result
```

I		t	h	i	n	k		t	h	e	r	e	f	o	r	e		I		a	m

```
0 1 2 3 4 5 6 7 8 . . .                    -4 -3 -2 -1
```

What does this code fragment output?

```
think think think
I am therefore I think
```

Numerics

- Integers
  ```
  >>> x = 10
  >>> type(x)
  <type 'int'>

  >>> y = 10000000000
  >>> type(y)
  <type 'long'>
  ```

- Decimals
  ```
  >>> x = 3.1415
  >>> type(x)
  <type 'float'>
  ```

Numerics

- **Complex numbers**
  - 1j represents $\sqrt{-1}$

```
>>> x = 5 + 1j       # 5 + √−1
>>> type(x)
<type 'complex'>
```

Basic Arithmetic Operations

```
>>> x = 5
>>> y = 8
```

- Addition
  ```
  >>> x + y
  13
  ```
- Subtraction
  ```
  >>> x – y
  -3
  ```
- Multiplication
  ```
  >>> x * y
  40
  ```

Basic Arithmetic Operations

```
>>> x = 5
>>> y = 8
```

- Modulo division
```
>>>  y % x
3
>>> -8 % 5
2
```

Basic Arithmetic Operations

```
>>> x = 5
>>> y = 8
```

- Equality
  ```
  >>> x == y
  False
  >>> x == 5
  True
  ```

- Inequalities
  ```
  >>> x < y
  True
  >>> x <= y
  True
  >>> x > y
  False
  ```

19

Division

- Float division

```
>>> x = 10.0
>>> y = 8.0
>>> x / y
1.25
```

- Integer division. The result is rounded down to the nearest integer.

```
>>> x = 10
>>> y = 8
>>> x / y
1                 # 1.25 rounded down


>>> x = -10
>>> x / y
-2                # -1.25 rounded down
```

Division

- If one variable is a float, then do float division.
- This is known as "type coercion", i.e. coercion of integers to float.

```
>>> x = 10
>>> y = 8.0
>>> x / y
1.25
```

Order of numeric operations

- Same as standard arithmetic writing

1. Parenthesis
2. `**` (Exponent)
3. `*, /` (Multiplication, division)
4. `+,–` (Addition, subtraction)
5. `–` (Negative)

- If operations have equal precedence, then evaluate from left to right.
- Evaluate

```
>>> 3 + 6 / 3 * (1 + 1)
7
```

Booleans

- Variables with two values
  - True
  - False

```
# It's a sunny day!
>>> is_sunny = True
>>> type(is_sunny)
<type 'bool'>

# It's not raining!
>>> is_raining = False
>>> type(is_raining)
<type 'bool'>
```

Boolean logic
the not statement

```
>>> a = True
>>> b = True
>>> c = False
>>> d = False

# not x := the opposite of x
>>> not a
False
>>> not c
True
```

Boolean logic
the and statement

```
>>> a = True
>>> b = True
>>> c = False
>>> d = False

# x and y := Evaluate x.  If x is False, return x. If not, return y
#          := True only when both x and y are True
>>> a and b
True
>>> a and c
False
>>> c and d
False
```

Boolean logic
the or statement

```
>>> a = True
>>> b = True
>>> c = False
>>> d = False

# x or y := Evaluate x.  If x is True, return x.  If not, return y
#         := False only when both x and y are False.
>>> a or b
True
>>> a or c
True
>>> c or d
False
```

Boolean logic practice

```
>>> a = True
>>> b = True
>>> c = False
>>> d = False

>>> ((a or d) and c)
False

>>> (b and c or d) and a
False
```

Boolean Coercion

- 0 and ‘’ are considered False in a Boolean context.
- All other numbers and Strings are considered True.

```
# x and y := Evaluate x.  If x is False, return x.  If
  not, return y.
>>> ‘’ and 2
‘’

>>> 2 and 0
0
>>> True and 4
4
```

Boolean Coercion

```
# not x := the opposite of x
>>> not 2
False
>>> not ''
True

# x or y := Evaluate x.  If x is True, return x.  If not,
   return y
>>> '' or 2
2
>>> 3 or 0
3
>>> False or 0
0
```

Naming your variables

- Name your variables to indicate what they're storing
  - Not helpful
    ```
    >>> x = 'Kenya'
    ```
  - Informative
    ```
    >>> country = 'Kenya'
    ```

- Use lowercase_with_underscores for multi-word functions and variable names
  - Encouraged
    ```
    >>> soccer_team = 'Black Stars'
    ```

Naming your variables

- First character must be a letter
  - Invalid
    ```
    >>> 1country = 'Kenya'
    >>> ₵five = 5
    ```
  - Valid
    ```
    >>> one_country = 'Kenya'
    ```

- Keep the name short for readibility
  - Too long:
    ```
    >>> the_capital_city_of_Kenya = 'Nairobi'
    ```
  - Shorter
    ```
    >>> capital_Kenya = 'Nairobi'
    ```

Output

- Just print it out!

```
# print a string
>>> print 'Gooooal!'
Gooooal!

# without a print, the quotes remain
>>> 'Gooooal!'
'Gooooal!'

# print other data types
>>> print 3.1415
3.1415
```
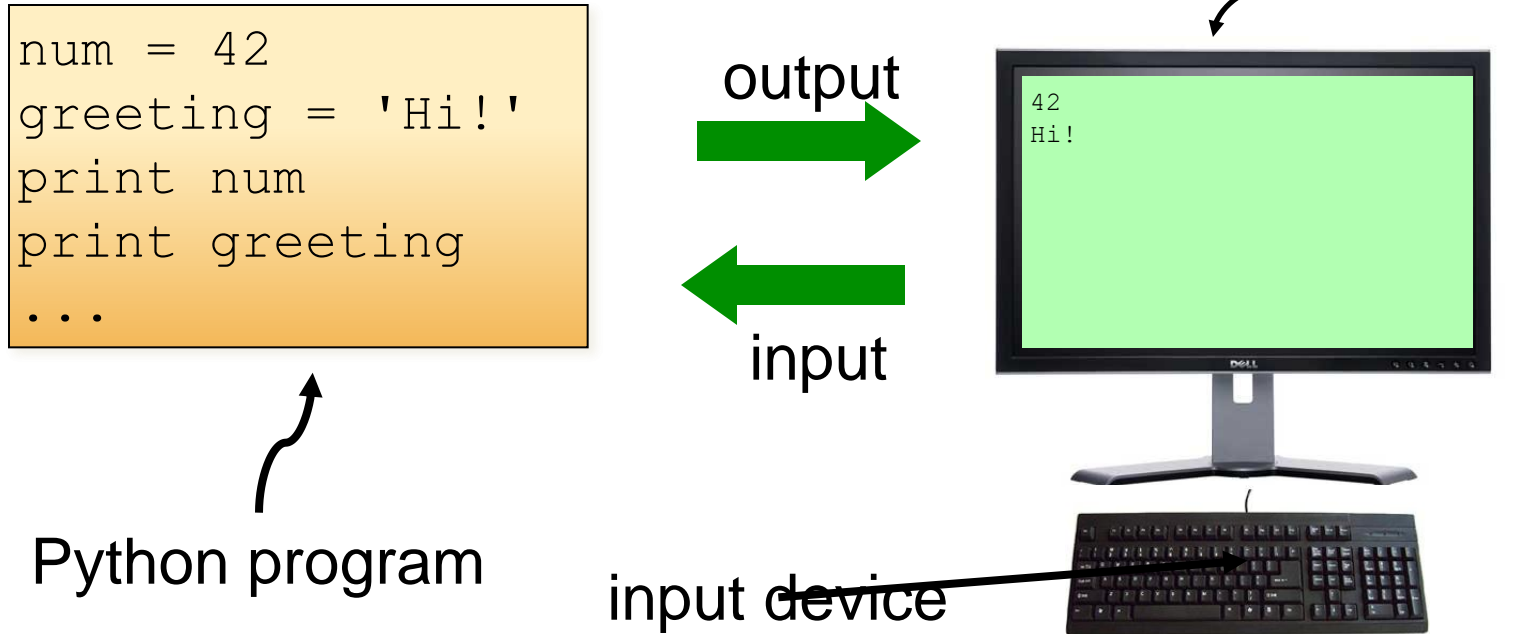
Output

- ## Print newlines with the \n character

  ```
  >>> print 'First line\nSecond line'
  First line
  Second line
  ```

- ## Separate multiple phrases with commas

  ```
  >>> players = 11
  >>> print 'There are', players, 'players'
  There are 11 players on each team
  ```

Input

- We would also like to get input from the user.

```
num = 42
greeting = 'Hi!'
print num
print greeting
...
```

Python program

output →

← input

console

```
42
Hi!
```

input device

User Input

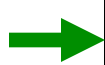- raw_input prints a prompt to the user and assigns the input to a variable as a string

```
name = raw_input('What is your name?')
```

- input can be used when we expect the input to be a number

```
age = input('How old are you?')
```

An input example

```
name = raw_input('What is your name?')
prompt = 'How old are you, ' + name + '?'
age = input(prompt)
print 'I want to be', age, 'years old too!'
```

An input example

```
name = raw_input('What is your name?')
prompt = 'How old are you, ' + name + '?'
age = input(prompt)
print 'I want to be', age, 'years old too!'
```

What is your name?

An input example

```
name = raw_input('What is your name?')
prompt = 'How old are you, ' + name + '?'
age = input(prompt)
print 'I want to be', age, 'years old too!'
```

What is your name?
Max

An input example

```
name = raw_input('What is your name?')
prompt = 'How old are you, ' + name + '?'
age = input(prompt)
print 'I want to be', age, 'years old too!'
```

What is your name?
Max

An input example

```
name = raw_input('What is your name?')
prompt = 'How old are you, ' + name + '?'
age = input(prompt)
print 'I want to be', age, 'years old too!'
```

What is your name?
Max
How old are you, Max?
19

An input example

```
name = raw_input('What is your name?')
prompt = 'How old are you, ' + name + '?'
age = input(prompt)
print 'I want to be', age, 'years old too!'
```

What is your name?
Max
How old are you, Max?
19
I want to be 19 years old too!