CS101: Intro to Computing
Spring 2016

Lecture 4
Administrivia

• i>clicker attendance starts *today*
  – Make sure to register on Compass!
• Homework 2 is due *Friday*
REVIEW
```python
x = "3"
y = 10 % 4
print(x * y)
```

What is the output of this program?

a) 2
b) 104
c) 33
d) 3104
c=(10+5j)
i=25
r=c.real+i

What is the value and type of r?
a) An integer with value 35
b) A complex with value 35+5j
c) A float with value 35.0
d) None of the above.
Which of these expressions will cause an *overflow*?

a) $10^{100000}$
b) "10" * 100000
c) $10.0^{100000}$
d) None of the above
x="10"
y="%i"
print((x+y) % 2)

What is the output of this program?

a) 102
b) 1111
c) 1010
d) None of the above
STRING TYPE
Strings

• Literals: text surrounded by quotes
  – e.g. “TACO”
• Each symbol is called a character
• Unlike numeric types, strings can vary in length!
String operations

- **Concatenation**: combine two strings
  - Uses the + symbol
  - Example: “CS”+”101”

- **Repetition**: repeat a string
  - Uses the * symbol
  - Example: “HELLO! ”*10

- **Formatting**: used to encode other data as a string
  - Uses % symbol
Formatting operator

• Creates a string with a value stuck inside
  – Formatting them nicely
  – Have to indicate the *type* of the value INSIDE the string with a special code

```python
x=100 * 54
s="String is: %i" % x
print s
```
Example

ame="Ryan"
grade=0.95
m1="Hello, %s!" % name
m2="Your grade is: %f" % grade
print(m1)
print(m2)
Indexing operator

• Extracts a *single* character
• Use an integer surrounded by brackets
  – e.g. a[0]
  – Call integer the “index”

**WARNING**: We start counting from 0

• Can use negative numbers
  – Starts from end (e.g. -1 is the last character)
my_string="ABCDE"
i=3

x=my_string[i]

What is the value of x?

a) A
b) B
c) C
d) D
e) E
my_string="ABCDE"
i=25\%3
x=my_string[i]
What is the value of x?
a) A  
b) B  
c) C  
d) D  
e) E
my_string="ABCDE"
i=(11%3)-7
x=my_string[i]

What is the value of x?
a) A
b) B
c) C
d) D
e) E
Slicing

• Extracts a *substring* from a string
• Similar to indexing notation
  – We can specify a *range* inside the brackets using : (colon) character
  – e.g. “Taco salad”[0:4]
• Character at first index *is included*
• Character at last index *is not included*
my_string="ABCDE"
x=my_string[1:3]

What is the value of x?
a) AB  
b) ABC 
c) BC  
d) BCD 
e) CD  
FUNCTIONS
Functions

• A small program we can run within Python
  – Saves us from having to rewrite code
  – Don’t reinvent the wheel!

• **ANALOGY**: Functions are *verbs* in Python.

• Also called a *subroutine* or *procedure*
Function calls

- When we want to execute a function, we **call** it or **invoke** it
- Use name of the function with parentheses
  - Example: `print()`
- Many functions are part of the Python language
  - We call them **built-in functions**
Arguments

• Functions can act on data
• Arguments are the *input* to a function
• The function *returns* a value
• Return values are the *output* of a function
• Examples:
  – print(10)
  – len(“TACO TUESDAY”)
  – abs(-123)
Arguments

- A function can take more than one argument
- Multiple arguments are separated by commas
- Examples:
  - \( \text{min}(1,4,5) \)
  - \( \text{max}(1,4,5) \)
Type conversion

• Built-in functions that convert data of one type to another

• Examples:
  – `float("0.3")`
  – `str(3+5j)`

• Some type conversions don’t work:
  – `int("TACO")`
  – `int(3+5j)`
User input

- input() is a built-in function
- Argument: string prompting the user
- Return value: string user typed before hitting “ENTER”
Goal

• Purpose of a program is to *achieve a goal!*
• Next time, we’ll write our first program.