Python Basics!

functions, scope

CS101 Lecture #4
Register your i>clickers on the course Compass page—attendance counts from today!
Register your i>clickers on the course Compass page—attendance counts from today!

Don’t panic!
Register your i>clickers on the course Compass page—attendance counts from today!
  • Don’t panic!
Complete Homework #1 before 6:00 p.m. today.
Register your i>clickers on the course Compass page—attendance counts from today!
  - Don’t panic!
Complete Homework #1 before 6:00 p.m. today.
Homework #2 is due Friday Sep. 9.
Register your i>clickers on the course Compass page—attendance counts from today!
- Don’t panic!
Complete Homework #1 before 6:00 p.m. today.
Homework #2 is due Friday Sep. 9.
No lab next week (Labor Day).
Warmup Quiz
x = "3"
y = 10 % 4
print(x * y)

What does this program print?

A 6
B 2
C 33
D 32
c = (10 + 5j)
i = 25
r = c.real + i

What is the type and value of r?
A int, 35
B complex, 35 + 5j
C float, 35.0
D complex, 35 + 0j
Which of these expressions is most likely to cause an overflow?

A 10 ** 100000
B ”10” * 100000
C 10.0 ** 100000
D ”10” ** 100000
E None of the above
Question #4

```python
x = "10"
y = "%i"
print((x+y) % 2)
```

What does this program print?
A  102
B  1111
C  1010
D  None of the above
Data Types—A Few Points
Complex numbers, \( \mathbb{C} \)

- Represent numbers with an imaginary component.
- Use \( j \) for \( i \):
  \[
  z = 1.0 + 1j
  \]
- Represent numbers with an imaginary component.
- Use \( j \) for \( i \):
  
  \[
  z = 1.0 + 1j
  \]
  
  \[
  z\text{.real} + z\text{.imag} * 1j
  \]
Strings

- As a literal: text surrounded by quotes. 
  - ”DEEP”
- Each symbol is a character.
- Unlike numeric types, strings vary in length.
String operations

- **Concatenation**: combine two strings
  - Uses the + symbol
  - ’RACE’ + ’CAR’

- **Repetition**: repeat a string
  - Uses the *
  - ’HELLO ’*10

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

- Creates string with value inserted
  - Formats nicely
  - Requires indicator of type inside of string
    - "%i" int
    - "%f" float
    - "%e" float (scientific notation)
    - "%s" str
Example

print( "An integer:  %i" % 7 )
print( "A float:    %f" % 7.0 )
print( "A float:    %e" % 7.0 )
print( "A string:   %s" % 'seven' )
Indexing operator \[
\]

- Extracts single character
  
  ```
  a = "FIRE"
  a[0]
  ```

- The integer is the index.
Extracts single character
\[a = “FIRE”\]
\[a[0]\]

The integer is the index.

We count from zero!
Extraction of single character

```
a = "FIRE"
a[0]
```

- The integer is the index.
- We count from zero!
- If negative, counts down from end.
Extracts single character
```
a = "FIRE"
a[0]
```
The integer is the index.

We count from zero!
If negative, counts down from end.
Does this work on other data types like int?
Slicing operator:

- Extracts range of characters (substring)

Example: 
```
a = "FIREHOUSE"
a[0:4]
```
Slicing operator:

- Extracts range of characters (*substring*)
- Range specified inside of indexing operator
Slicing operator:

- Extracts range of characters (substring)
- Range specified inside of indexing operator

```python
a = "FIREHOUSE"
a[0:4]
```
Slicing operator:

- Extracts range of characters (substring)
- Range specified inside of indexing operator
  
  ```python
  a = "FIREHOUSE"
a[0:4]
  ```
- Can be a bit tricky at first:
  - Includes character at first index
  - Excludes character at last index
alpha = "ABCDE"
x = alpha[1:3]

What is the value of x?
A  'AB'
B  'ABC'
C  'BC'
D  'BCD'
E  'CD'
A function is a small program (block of code) we can run within Python.
A function is a small program (block of code) we can run within Python.

- Saves us from rewriting code
- Don’t reinvent the wheel!
A function is a small program (block of code) we can run within Python.
- Saves us from rewriting code
- Don’t reinvent the wheel!
- Analogy: Functions are more verbs.
A function is a small program (block of code) we can run within Python.

- Saves us from rewriting code
- Don’t reinvent the wheel!

Analogy: Functions are more verbs.
Also called subroutine or procedure.
When we want to execute a function, we call or invoke it.
When we want to execute a function, we call or invoke it.
Use name of the function with parentheses.
When we want to execute a function, we call or invoke it.
Use name of the function with parentheses.

`print()`
When we want to execute a function, we call or invoke it.

Use name of the function with parentheses.

- `print()`

Many functions come built-in to Python or in the standard library.
When we want to execute a function, we call or invoke it.

Use name of the function with parentheses.

- `print()`

Many functions come built-in to Python or in the standard library.

Others we will compose at need.
Functions can act on data.
- Functions can act on data.
- Arguments are the input to functions.
Functions can act on data.

Arguments are the input to functions.

Functions can return a value. (fruitful)
Functions can act on data.

- Arguments are the input to functions.
- Functions can return a value. (fruitful)
- Return values are the output of a function.
Functions can act on data.
- Arguments are the input to functions.
- Functions can return a value. (fruitful)
- Return values are the output of a function.
  - `print('10')`

```python
print('10')
len('Rex Kwon Do')
abs(-123)
```
Functions can act on data.
Arguments are the input to functions.
Functions can return a value. (fruitful)
Return values are the output of a function.

- `print('10')`
- `len('Rex Kwon Do')`
- Functions can act on data.
- **Arguments** are the input to functions.
- Functions can return a value. (fruitful)
- Return values are the output of a function.
  - `print('10')`
  - `len('Rex Kwon Do')`
  - `abs(-123)`
Functions can act on data.
- **Arguments** are the input to functions.
- Functions can return a value. (fruitful)
- Return values are the output of a function.
  - `print('10')`
  - `len('Rex Kwon Do')`
  - `abs(-123)`
Arguments are values passed to a function.
Arguments

- Arguments are values passed to a function.
- A function can accept zero to many arguments.
Arguments are values passed to a function.
A function can accept zero to many arguments.
Multiple arguments are separated by commas:
- min( 1,4,5 )
- max( 1,4,5 )
Type conversion.

- A set of built-in functions to convert data from one type to another.

Examples:
- `float(“0.3”)`
- `str(3 + 5j)`

Be careful of nonsense:
- `int(“Rex”)`
- `int(3 + 5j)`

Also called subroutine or procedure.
A set of built-in functions to convert data from one type to another.

- `float( "0.3" )`
- `str( 3 + 5j )`
A set of built-in functions to convert data from one type to another.

- `float( "0.3" )`
- `str( 3 + 5j )`

Be careful of nonsense:

- `int( "Rex" )`
- `int( 3 + 5j )`

Also called subroutine or procedure.
input is a built-in function.
- `input` is a built-in function.
- Argument: string prompting user
- `input` is a built-in function.
- Argument: string prompting user
- Return value: input from user (as `str`)
A program should achieve a goal.
A program should achieve a goal.

Next time we will write our first nontrivial program.
Reminders

- Homework #1 due today, Aug. 31, 5:00 p.m.
- Homework #2 due Friday, Sep. 9, 5:00 p.m.
- No class Monday, Sep. 5 (Labor Day).
- No lab next week!