Computational Basics

Refining Process
Coursework
go.illinois.edu/cs101

- Late adds need to see the FAQ
- i>clicker test today
- hw00, hw01 due today
- hw02 due 2/3
- lab01 this week
exam0 this week
Coursework

- exam0 this week
- Extra credit, equiv. to 1/16th of regular exam
Coursework

- exam0 this week
- Extra credit, equiv. to 1/16th of regular exam
- exam1 next week (see website for coverage)
exam0 this week
Extra credit, equiv. to 1/16th of regular exam
exam1 next week (see website for coverage)
Piazza: posts with homework code should be private.
Recap
x = 3
s = ("%i" % (x+1)) * x**(5%x)
print(s)

What does this program print?
x = 3
s = ("%i" % (x+1)) * x**(5%x)
print(s)

What does this program print?
s = ("%i" % (3+1)) * 3**(5%3)
\[ x = 3 \]
\[ s = \left( \text{"}\%i\" \% (x+1) \right) \times x^{\text{\texttt{5\%x}}} \]
\[ \text{print}(s) \]

What does this program print?
\[ s = \left( \text{"}\%i\" \% (3+1) \right) \times 3^{\text{\texttt{5\%3}}} \]
\[ s = \left( \text{"}\%i\" \% 4 \right) \times 3^{\text{\texttt{2}}} \]
Example

```python
x = 3
s = ('%i' % (x+1)) * x**(5%x)
print(s)
```

What does this program print?
```
s = ('%i' % (3+1)) * 3**(5%3)
s = ('%i' % 4) * 3**2
s = "4" * 9
```
```python
x = 3
s = ('%i' % (x+1)) * x**(5%x)
print(s)
```

What does this program print?

```python
s = ('%i' % (3+1)) * 3**(5%3)
s = ('%i' % 4) * 3**2
s = '4' * 9
s = '444444444'
```
x = "3"
y = 10 % 4
print(x * y)

What does this program print?
A  6
B  2
C  33
D  32
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
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 are likely to overflow.
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 are likely to overflow.
x = "10"
y = "%i"
print((x+y) % 2)

What does this program print?
A 102
B 1111
C 1010
D None of the above
x = "10"
y = "%i"
print((x+y) % 2)
What does this program print?

A 102 ★
B 1111
C 1010
D None of the above
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 (also "%d")
    - "%f" float
    - "%e" float (scientific notation)
    - "%s" str
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!
Indexing operator \[
\]

- Extracts single character
  \[
  a = \text{”FIRE”}
  a[0]
  \]
- The integer is the index.
- **We count from zero!**
- If negative, counts down from end.
Indexing operator \[
\]

- 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 \texttt{int}?
Indexing operator

- Extracts single character
Indexing operator

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

- Extracts single character
  \[a = \text{"FIRE"}\]
  \[a[0]\]
  - The integer is the index.
Indexing operator

- Extracts single character
  a = "FIRE"
  a[0]
- The integer is the **index**.
- We count from zero!
Indexing operator

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

- The integer is the index.

- We count from zero!

- If negative, counts down from end.
s = "ABCDE"
i = 3
x = s[i]

What is the value of x?
A 'A'
B 'B'
C 'C'
D 'D'
E 'E'
s = "ABCDE"
i = 3
x = s[i]

What is the value of x?
A 'A'
B 'B'
C 'C'
D 'D' ★
E 'E'
s = "ABCDE"
i = 25 % 3
y = s[i]

What is the value of $y$?

A  'A'
B  'B'
C  'C'
D  'D'
E  'E'
s = "ABCDE"
i = (11 % 3) - 7
z = s[i]

What is the value of z?
A 'A'
B 'B' *
C 'C'
D 'D'
E 'E'
Slicing operator:

- Extracts range of characters (substring)

Range specified inside of indexing operator

```python
a = "FIREHOUSE"
a[0:4]  # Includes character at first index, excludes character at last index
```
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
  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"
\[x = \text{alpha}[1:3]\]

What is the value of \(x\)?

A. 'AB'
B. 'ABC'
C. 'BC'
D. 'BCD'
E. 'CD'
Example

alpha = "ABCDE"
x = alpha[1:3]

What is the value of x?
A  'AB'
B  'ABC'
C  'BC' ✳
D  'BCD'
E  'CD'
Functions
And now for something different...

1 + abs( -1 )
abs( 1 - 1 )
min( 5,4,3,2,3,4,5,6 )
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: If operators are verbs, 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: If operators are verbs, 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.

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

Use name of the function with parentheses.

- `print('hello world')`
- `bin(11)`
When we want to execute a function, we call or invoke it.

Use name of the function with parentheses.

- `print('hello world')`
- `bin(11)`

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( 'hello world' )`
- `bin( 11 )`

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

Others we will compose at need (see lec04).
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')`
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)`
Arguments

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

- `float( "0.3" )`
- `str( 3 + 5j )`
- `int( 1.5 )`

Be careful of nonsense:

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

Also called subroutine or procedure.
Type conversion

- A set of built-in functions to convert data from one type to another.
  - `float( "0.3" )`
  - `str( 3 + 5j )`
  - `int( 1.5 )`

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

- Also called subroutine or procedure.
Python offers many libraries to support other operations.

```python
import math
def factorial(n):
    return math.factorial(n)
print(factorial(5))
print(math.log(10))
print(math.pi)
print(math.e)
```
Python offers many libraries to support other operations.

```python
import math
math.factorial(5)
math.log(10)
math.pi
math.e
```
Python offers many libraries to support other operations.

```python
import math
math.factorial( 5 )
math.log( 10 )
math.pi
math.e
```

Note that you need to include the module name and the attribute operator `.`.

Langtangen refers to these frequently.
Next steps
Next steps

- Check registration of i>clicker on Compass
- Complete quiz03 (due 1/31)
- Complete hw00, hw01 (due 1/30)
- Complete hw02 (due 1/30)
- Read for the next class