Coursework
- i>clickers count today
- exam0 this week
- hw02 due 2/3
- exam1 next week (lec00–lec04)
- Check posted grades on Compass regularly.
s = '%%' + 'f'
i = 3 / 6
x = float(s%i) * 2

What is the value of \( x \)?

A '0.50.5'
B '%f%f'
C 1.0
D '1.0'
\[ s = \%' + 'f' \]
\[ i = 3 \div 6 \]
\[ x = \text{float}(s%i) \times 2 \]

What is the value of \( x \)?

A \('0.50.5'\)
B \('%f%f'\)
C \(1.0\star\)
D \('1.0'\)
s = "WATER MAIN"[2:6]
t = int(3.7)
x = s[-1] + s[t-2]

What is the value of \( x \)?

A ”NA”
B ” E”
C ” R”
D ”ME”
s = "WATER MAIN"[2:6]
t = int(3.7)
x = s[-1] + s[t-2]

What is the value of x?
A "NA"
B "E" ★
C "R"
D "ME"
s = "WATER MAIN"[2:6]  #0123456789
s = "TER "
t = int(3.7)
t = 3
x = s[-1] + s[t-2]
x = "" + "E"
x = " E"
q = len("WATER MAIN")
c = ( 1.0 + 2.0j ) * ( -q )
x = abs( min( c.real, -13 ) )

What is the value of \( x \)?

A 0  
B 11  
C 12  
D 13
q = len("WATER MAIN")
c = (1.0 + 2.0j) * (-q)
x = abs(min(c.real, -13))

What is the value of $x$?

A 0
B 11
C 12
D 13 ★
Functions Redux
When we want to execute a function, we call or invoke it.

- Use name of the function with parentheses.
  - `print( 'hello' )`

- Many functions come built-in to Python or in the standard library.
  - `type( 1+3.4 )`—type of an expression
  - `len( 'tortoise' )`—length of str

- Others we will compose at need.
- **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.
Let’s implement the quadratic equation.
Example: Quadratic equation

```python
print( "QUADRATIC SOLVER" )
print( "a x**2 + b x + c = 0" )

a = float( input( 'a = ' ) )
b = float( input( 'b = ' ) )
c = float( input( 'c = ' ) )

root = ( b**2 - 4*a*c ) ** 0.5
denom = 2 * a
```
Example: Quadratic equation

```
pos = (-b + root) / denom
neg = (-b - root) / denom

message1 = "%.2f + %.2fi" % (pos.real,pos.imag)
messaged2 = "%.2f + %.2fi" % (neg.real,neg.imag)

print("Solution 1: %s" % message1)
print("Solution 2: %s" % message2)
```
Achievement unlocked!

Write a program.
We can explain our code using comments. Comments begin with a # sign; Python ignores the rest of the line. Long comments can also be stored as triple-quoted strings.
We can explain our code using comments.

Comments begin with a # sign; Python ignores the rest of the line.

Long comments can also be stored as triple-quoted strings.

dx = 0.01  # grid spacing, m
V  = 14.2  # voltage, V
We can explain our code using comments. Comments begin with a # sign; Python ignores the rest of the line. Long comments can also be stored as triple-quoted strings.

dx = 0.01  # grid spacing, m
V = 14.2  # voltage, V

""
I am an extended comment. I can be many lines long. Use me to explain functions or formulae, to document code, or to hide code you don’t want to run.
"""
Composing Functions
def pow(a, b):
    y = a ** b
    return y
We define a new function with the following:
- the keyword `def`
- the name of the function
- a pair of parentheses
- a block of code
- a return statement (optional)
def greetings():
    print("Bom dia!")
    print("Bonjour!")
    print("Ni hao!")
    print("Hello!")
    print("Guten tag!")
    print("Konichiwa!")
A section of code grouped together.
- Begins with a `:`.
- Contents of the block are indented:

```python
def hello():
    print('hello')
```
Variables defined inside of a block are independent of variables outside of the block.

Variables inside a block do not exist outside of the block.

Blocks are isolated from the rest of the code!
def pendulum(L):
    import math
    g = 9.8  # m/s^2
    T = 2 * math.pi * math.sqrt(L / g)
    return T

# out here, I can't refer to vars in the function
time = pendulum(1.0)
print(time)
def pendulum( L ):
    import math
    L = 1.0  # m  <= why doesn’t this work?
g = 9.8  # m/s^2
    T = 2 * math.pi * math.sqrt( L / g )
    return T

# out here, I can’t refer to vars in the function
time = pendulum( 1.0 )
print( time )
Example: Defining functions

```python
a = 5
def fun():
    a = 3
    b = 4
    a = a + b
fun()
print(a)
```
Functions can return values with the keyword `return`.

```python
def three():
    return 3
```

`return` immediately exits a function.
Functions can return values with the keyword `return`.

```python
def three():
    return 3
```

`return` immediately exits a function.

```python
def zero():
    return 0
    print('0')
```
Functions can accept values as parameters (input, arguments).

These variables are declared in the function header.

Multiple parameters are separated by commas.

```python
def print_message( msg ):
    print( 'I say:', msg )
```
def fun(a):
    return a+2

x = fun(2) * fun(3)

What is the value of x?
A  6
B  8
C  24
D  None of the above.
def fun(a):
    return a+2

x = fun(2) * fun(3)

What is the value of $x$?

A  6
B  8
C  24
D  None of the above.  ★ (20)
def spacer(m):
    return m + ' '

x = spacer("abb") + spacer("acab")

What is the value of x?
A 'abb acab'
B 'abb acab'
C 'abbacab'
D None of the above (error)
def spacer(m):
    return m + ', '

x = spacer(”abb”) + spacer(”acab”)

What is the value of x?
A ’abb acab ’ ⋆ (note the space!)
B ’abb acab’
C ’abbacab’
D None of the above (error)
- **bool** is a type with two possible values:
  - True
  - False
- We use these to make decisions.
- The logic is based on **Boolean algebra**.
Boolean

- `bool` is a type with two possible values:
  - True
  - False
- We use these to make decisions.
- The logic is based on Boolean algebra.
- Operators:
  - `and`
  - `or`
  - `not`
Example: Boolean logic

$$0 < x \leq 10$$

$$(x > 0) \text{ and } (x \leq 10)$$
Example: Boolean logic

\[ 0 < x \leq 10 \]

\[ (x > 0) \text{ and } (x \leq 10) \]

\[ \text{NOT } x > 0 \text{ and } \leq 10! \]
Boolean operators

Operators:
- **and**—True only if both sides are True
- **or**—True if either side is True
- **not**—swaps False and True
def fun():
    return True and False
x = fun() and not (True or False)
What is the value of x?
A True
B False
def fun():
    return True and False

x = fun() and not (True or False)

What is the value of x?

x = (True and False) and not (True or False)
def fun():
    return True and False
x = fun() and not (True or False)
What is the value of $x$?
x = (True and False) and not (True or False)
x = (False) and not (True)
def fun():
    return True and False

x = fun() and not (True or False)

What is the value of x?

x = (True and False) and not (True or False)
x = (False) and not (True)
x = (False) and (False)
def fun():
    return True and False

x = fun() and not (True or False)

What is the value of $x$?
A True
B False ⭐
These produce Boolean output.
- less than, <
- greater than, >
- less than or equal to, <=
- greater than or equal to, >=
- equal to, ==
- not equal to, !=
Example

```
a = 5
b = 3

x = (a < 5) and ((b <= 5) or (a != b))
```

What is the value of \(x\)?

A True
B False
Example

```python
a = 'URSA MAJOR'
b = 'GEMINI'
x = a < b and a[1] != b[-2]
```

What is the value of `x`?

A True

B False
def fun(a,b):
    return a<b

a = 3
b = 4
x = fun(b,a)

What is the value of x?
A True
B False ★
Conditional Execution
Control flow represents actual sequence of lines executed by processor.

Conditional execution lets you execute (or not) a block of code based on logical comparison.
Example: *if* statement

```python
ans = input( "Enter a number:" )
if ans < 0:
    print( "The number was negative." )
```
Example: *if* statement

```python
ans = input( "Enter a number:" )
ans = float( ans )
if ans < 0:
    print( "The number was negative." )
```
We create an *if* statement as follows:
- the keyword *if*
- a logical comparison (results in *bool*)
- a **block** of code
This lets us make decisions in the program!
We can change program behavior as it executes.
Example: *if* statements

```python
ans = input( "Enter a number:" )
ans = float( ans )
if ans < 0:
    print( "The number was negative." )
if ans > 0:
    print( "The number was positive." )
if ans == 0:
    print( "The number was zero." )
```
h = '2'
j = '1'
k = int(j+h) % 5
if (k % 3) == 2:
    k = k ** 3
if (k % 3) == 1:
    k = k ** 2
Next steps

✦ Complete quiz04 (due 2/2)
✦ Complete hw02 (due 2/3)
✦ Prepare for exam1 (2/7-2/9)
✦ Read for the next class
✦ Don’t forget the colons after def and if!