

FUNCTIONS AND PARAMETERS

Outline

- Review of topics for Exam #1
- Basic Functions - Math, Maple, Python
- Function definition and invocation mechanics
- printFactorial example
- printDistance exercise
- Preview: Functions that return a value
- Nested function calls and execution order
- Code-reading exercise
- Functions and Graphics (finish for homework)

Exam 1

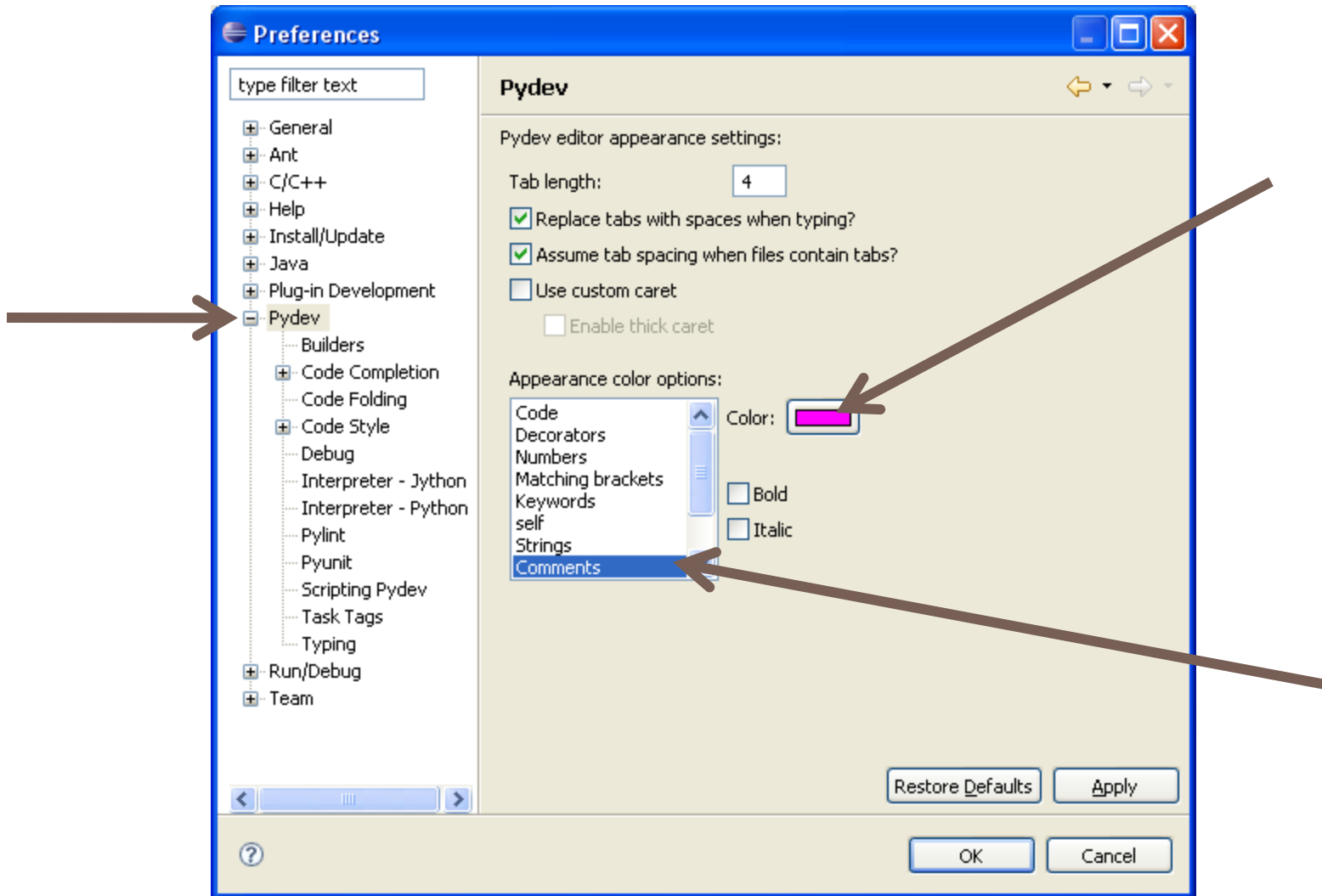
- When? Where?: See schedule
 - ▣ Please get in the habit of checking the schedule regularly. Time management is an problem solving process too!
- Format:
 - ▣ Paper part: open book, open notes, *closed computer*
 - ▣ Programming part: open book, notes, and computer

Possible Topics for Exam 1

- algorithm
- comment
- variable, assignment
- identifier, expression
- loop
 - ▣ definite (for)
 - ▣ counted (range function)
- phases of software development
- input, print
- import, math functions
- using functions
- int, float, long, conversion
- strings (basic operations)
- character codes (chr, ord)
- lists (concatenation, slices)
- reading, writing files
- formatted output
- using objects, graphics
- method vs function
- event-driven program

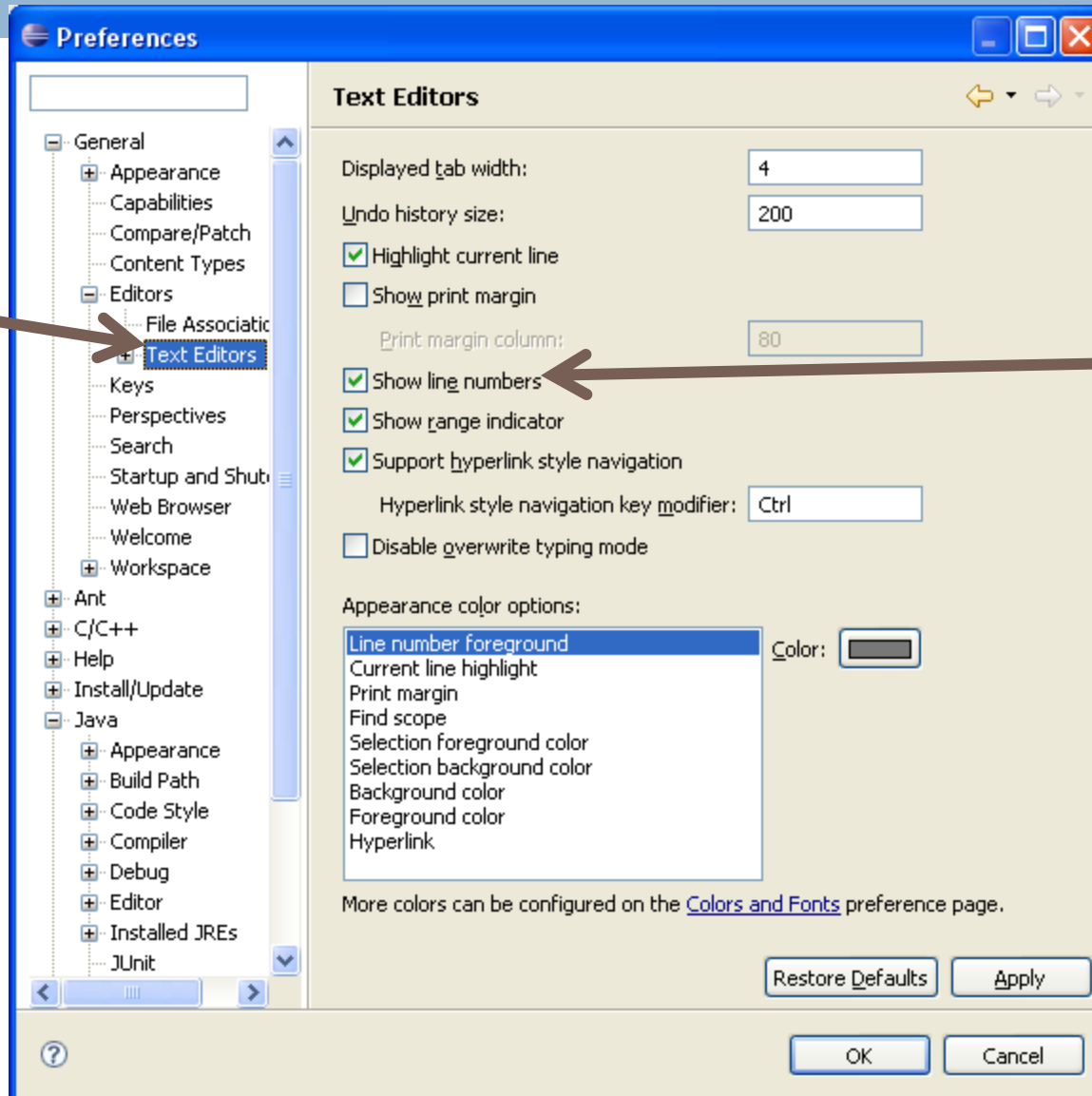
Changing Eclipse Colors

Choose Window → Preferences...



Show Line Numbers in Eclipse

Choose Window → Preferences...



Why functions?

- A function allows us to group together several statements and give them a name by which they may be invoked.
 - ▣ **Abstraction** (easier to remember the name than the code)
 - ▣ **Compactness** (avoid duplicate code)
 - ▣ **Flexibility** (parameters allow variation)
- Example:
 - ▣ **def complain(complaint):**
print "Customer:", complaint

Functions in different realms

- We compare the mechanisms for **defining** and **invoking** functions in three different settings:
- Standard Mathematical notation
- Maple
- Python

Functions in Mathematics

- Define a function:

- $f(x) = x^2 - 5$

Formal Parameter. Used so that we have a name to use for the argument in the function's formula.

- Invoke (call) the function:

- $$\frac{f(6) - f(3)}{6 - 3}$$

Two calls to function f. The first with actual parameter 6, and the second with 3.

- When the call $f(6)$ is made, the **actual parameter 6** is substituted for the formal parameter x , so that the value is $6^2 - 5$.

Functions in Maple

```
> f := x → x2 - 5;
```

f := x → x² - 5

Formal Parameter. Used so that we have a name to use for the argument in the function's formula.

Invoke the function.

```
> f(6);
```

Two calls to function f. The first with actual parameter 6, and the second with 3.

```
> 
$$\frac{f(6) - f(3)}{6 - 3};$$

```

31

9

Functions in Python

```
□ >>> def f(x):  
        return x*x - 5  
  
>>> f(6)  
31  
>>> (f(6) - f(3)) / (6 - 3)  
9  
>>>
```

Formal Parameter. Used so that we have a name to use for the argument in the function's formula.

Two calls to function `f`. The first with actual parameter 6, and the second with 3.

- ▣ What would `f(f(2))` evaluate to?
- In Mathematics, functions calculate a value.
- In Python we can also define functions that instead *do something*, such as print some values.

Review: Parts of a Function Definition

```
>>> def hello():  
    print "Hello"  
    print "I'd like to complain about this parrot"
```

*Defining a function
called "hello"*

Indenting tells interpreter
that these lines are part of
the hello function

Blank line tells interpreter
that we're done defining
the hello function

Review: Defining vs. Invoking

- Defining a function says what the function should do
- Invoking a function makes that happen
 - ▣ Parentheses tell interpreter to invoke the function

```
>>> hello()
```

```
Hello
```

```
I'd like to complain about this parrot
```

Review: Function with a Parameter

- `def complain(complaint):`
 - `print "Customer: I purchased this parrot not half " +`
`"an hour ago from this very boutique"`
 - `print "Owner: Oh yes, the Norwegian Blue. " +`
`" What's wrong with it?"`
 - `print "Customer:", complaint`

- invocation:
 - `complain("It's dead!")`

When a function is invoked Python follows a four-step process:

1. Calling program pauses at the point of the call
2. Formal parameters get assigned the values supplied by the actual parameters
3. Body of the function is executed
4. Control returns to the point just after where the function was called

```
from math import pi
```

```
def deg_to_rads(deg):
```

```
    rad = deg * pi / 180
```

```
    return rad
```

```
degrees = 45
```

```
radians = deg_to_rads(degrees)
```

```
print "%d deg. = %0.3f rad." \
```

```
      % (degrees, radians)
```

2: deg = 45

3

1

4

printFactorial function

```
def printFactorial(n, formatWidth):  
    formatString = "%"+str(formatWidth)+"d"  
    product = 1  
    for i in range(1, n+1):  
        product = product * i  
    print formatString % (product)
```

- Use this to compute and print a large factorial:
 - ▣ `printFactorial(15, 13)` :
- Note that the substitution of actual parameters for formal parameters is done in order.
- Use `printFactorial` to create a table of factorials

Exercise – with a partner

- Write and test a `printDistance` function:
 - `def printDistance(p1, p2):`
 `"""Parameters are Points, prints distance between them."""`
- Don't forget: `from zellegraphics import Point`
- Use this `main()` function to invoke your function:
 - `def main():`
 `printDistance(Point(5,6), Point(1, 9))`
 `printDistance(Point(12, 17), Point(6, 8))`
- Put your code file in the `printDistance` drop box
 - Located under In-class Exercise Drop Boxes
 - Make sure the title of your submission has both your names.
- Driver, email code to your navigator!

If a Function Calls a Function ...

```
def g(a,b):  
    print a+b, a-b
```

```
def f(x, y):  
    g(x, y)  
    g(x+1, y-1)
```

```
f(10, 6)
```

- Trace what happens when the last line of this code executes

Optional parameters

- A python function may have some parameters that are optional.

```
>>> int("37")
37
>>> int("37", 10)
37
>>> int("37", 8) # specify base 8
31
```

We can declare a parameter to be optional by supplying a default value.

```
>>> def printDate(month, day, year=2007):
    print month, str(day)+",", year

>>> printDate("January", 4, 2006)
January 4, 2006
>>> printDate("January", 4)
January 4, 2007
```

An exercise in code reading

- With a partner, read and try to understand the code that is on the handout.
- You can probably guess what the output will be. But how does it work?
- Figure that out, discuss it with your partner and answer quiz question 9.
- Optional Challenge Problem for later: try to write "There's a Hole in the Bottom of the Sea" or "The Green Grass Grew All Around" in a similar style.
- When you are done, turn in your quiz and start HW