Lecture 3
Intro to Python

A few programming symbols

The Difference Between Brackets, Braces, and Parentheses

One of the pieces of terminology that causes confusion is what to call certain characters. The Python style guide (and several dictionaries) use these names, so the Gries book does too:

() Parentheses ("parens")

[] Brackets

{} Braces

- Some people call braces *curly brackets* or *curly braces*, but Gries uses *braces*.

IDLE

Start IDLE (3.6.2)

A useful keyboard shortcut:

(Win)

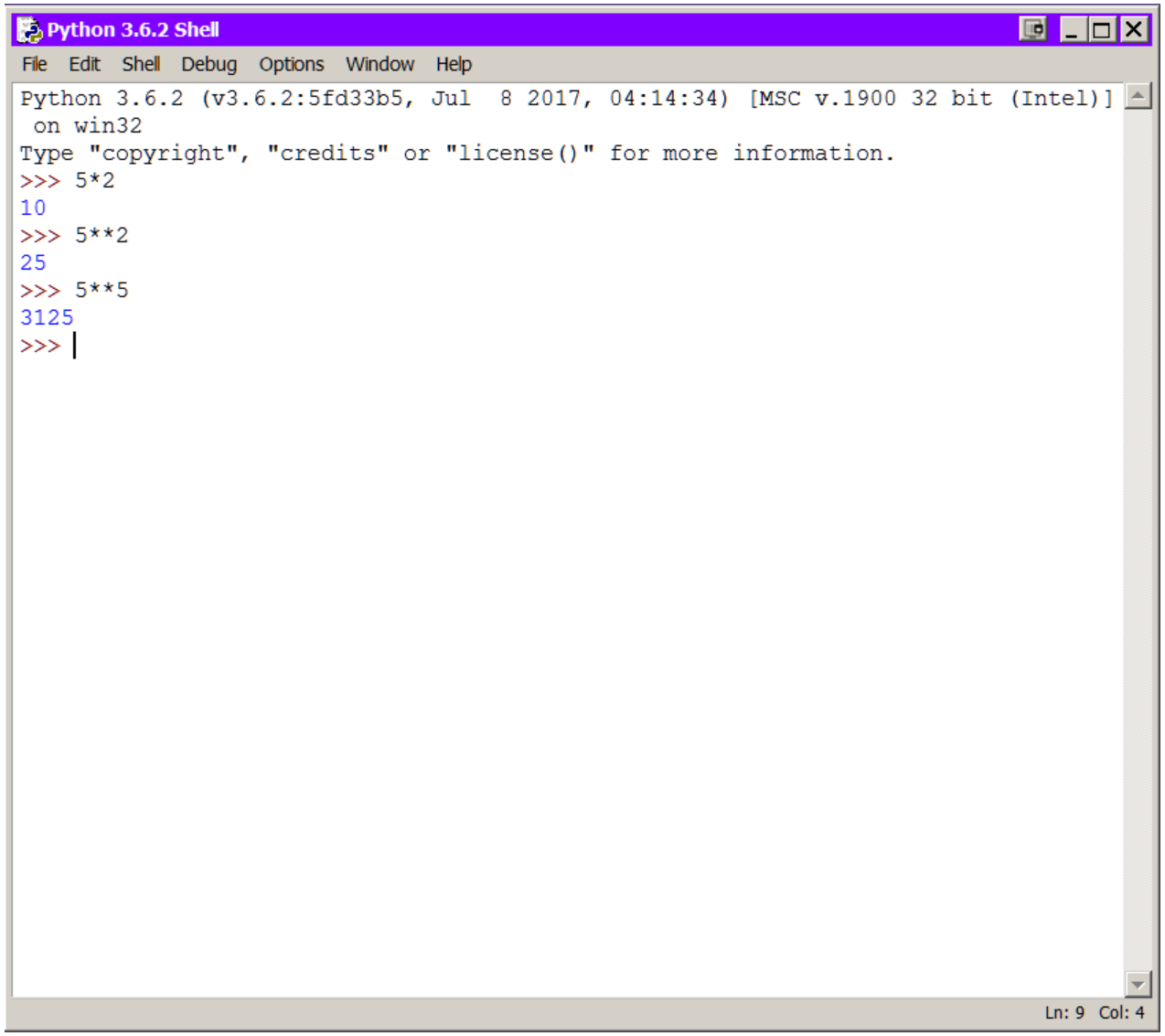
Alt-p: previous command

(OS X)

C-p

If you go too far:

Alt-n



Python is a calculator

```
# I will always try to put commands in bold black courier, results not bold
# (notice that this is a comment thanks to the pound sign, NOT a hashtag)
5+13
18

# more readable
5 + 13

# subtract:

5 - 88
```

The answer is an integer type of data (no decimal places)

Python is a calculator

```
# divide or involve fractions: results are not integer, but a float type
4 / 2
```

float type: floating-point decimal (3.22, 3.0, -45.25346345643563)

```
# also for addition/subtraction
5 - 88.4

5 + 13 / 5

subtract:

5 - 88/3
```

Python is a special calculator

```
# integer division, with a DOWNWARD truncated integer result (floor)
# returns the integer part of answer - NO ROUNDING
53 / 24
2.2083333333333335
53 // 24  # truncated integer division, use //
2
4 // 2
2
# strange but consistent: negative numbers (ALWAYS: NEXT LOWEST INTEGER)
53 // -24
-3
-53 // 24
-3
```

Python is a special calculator

```
# modulo: get the remainder after integer division
# use the percent (%) sign
53 % 24 # 24 x 2 = 48; 53 - 48 = 5.
5
# tricky example in book
17 // 10
1
17 % 10  # result? notice I can put a comment to the right of code too!

#but:
-17 // 10

-17 % 10  # hmmmm
-17 % -10  #   17 % -10
```

Python is a special calculator

```
# both modulo and floor using floats return a float
53.222 // 24.994757
2.0

53.222 % 24.994757   # remainder
3.2324860000000015

# Exponents use **
# 6 to the third power: 6³ (R, Excel: 6^3)
6 ** 3
216
6 ** 0.5  # what is this asking for?
2.449489742783178
```

Python is a special calculator

```
# be careful with negation (minus sign)
-2 ** 4
-16
Which means Python does the exponentiation BEFORE the negation
-(2 ** 4)  # how Python sees the above
-16
# what I wanted was this:
(-2) ** 4

# maybe it will not come up often:
val = -2
val ** 4
16
```

Summary of operations so far

Symbol	Operator	Example	Result
-	Negation	-5	-5
+	Addition	11 + 3.1	14.1
-	Subtraction	5 - 19	-14
*	Multiplication	8.5 * 4	34.0
/	Division	11 / 2	5.5
//	Integer Division	11 // 2	5
%	Remainder	8.5 % 3.5	1.5
**	Exponentiation	2 ** 5	32

Table 1—Arithmetic Operators

Precedence

Precedence	Operator	Operation
Highest	**	Exponentiation
	-	Negation
	*, /, //, %	Multiplication, division, integer division, and remainder
Lowest	+, -	Addition and subtraction

Table 2—Arithmetic Operators Listed by Precedence from Highest to Lowest

Python and computer precision

```
# these look funky when compared to each other
2 / 3
0.6666666666666666
5 / 3
1.6666666666666667
```

Remember from math a while ago, that $2/3 = 0.6\overline{6}$?

Computers must approximate numbers that need more than 15 or so digits

```
8/3
2.6666666666666665
7/3
2.3333333333333335
```

Python and computer precision

```
# but errors in precision do not grow
700/3
233.33333333333334
7000/3
2333.3333333333335

7000000000000/3
2333333333333.3335

70000000000000000000000000000000000000000000000000000000000000 / 3
2.3333333333333333e+60

scientific notation: a number >= 1 and < 10
with e (exponent) to a power (+/-)
```

Python and computer precision

```
# but precision can be tricky when testing for equivalence
2 / 3 + 1  # = 5/3
1.6666666666666665
5 / 3
1.6666666666666667
5 / 3  - (2/3 + 1)  # subtract
2.220446049250313e-16

10000000000 + 0.00000000001
10000000000.0
```

It looks like nothing happened!

Gries: Add from smallest to largest to minimize error

(computers: single, double, and extended precision)

Variables in Python

Convert Fahrenheit degrees to Celsius

Fahrenheit: water freezes at 32 degrees; water boils at 212 degrees

Celsius: metric! set 0 degrees = freezing and 100 degrees = boiling

Each Celsius degree = (212-32)/100 Fahrenheit degrees = 180/100 = 1.8

1.8 = 9/5

so to go from Celsius to Fahrenheit: (not code)

F = 9/5 * C + 32

C = (F - 32) * 5/9

Variables must start with a letter and can't be a number

what is 26 degrees Celsius in Fahrenheit?

```
DegCel = 26  # degrees_celsius or DC or CD or DegCel
```

Variables in Python

```
# To go from Celsius to Fahrenheit:
# F = 9/5 * C + 32
# C = (F - 32) * 5/9

DegCel = 26
9 / 5 * DegCel + 32 # convert to Fahrenheit
78.80000000000001
DegF = 9 / 5 * DegCel + 32  # assign variable DegF to the result
```

Did anything happen?

```
DegCel = 0  # change for another calculation
9 / 5 * DegCel + 32
32.0
DegF    # has not changed value
78.80000000000001
```

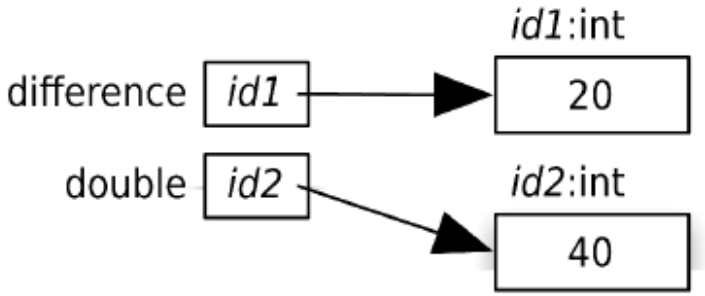
Variables in Python

Variables are stored in memory at a certain address

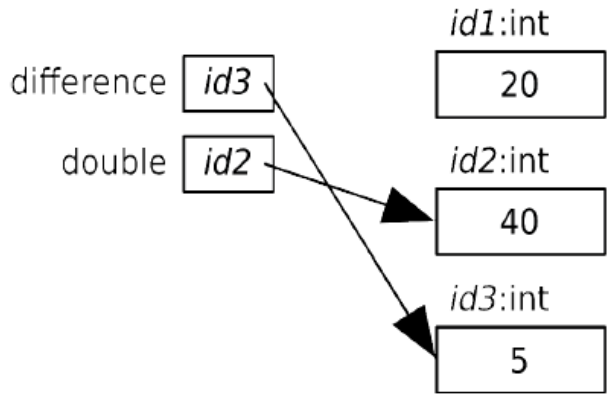
When variable is changed, another address gets written.

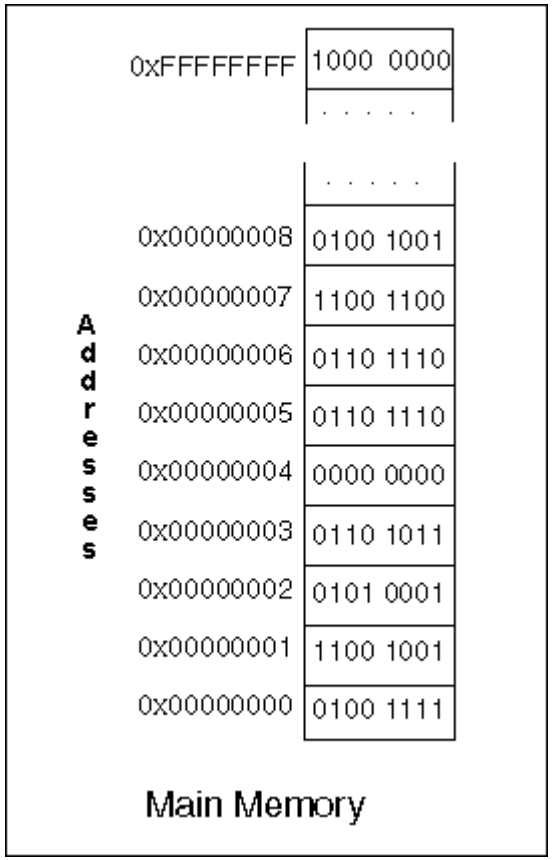
```
difference = 20 #id1 -> m1
double = 2 * difference  # id2 -> m2
double
```

```
40
```



difference = 5 # id3 -> m3





Variables in Python

```
# To change a variable based on itself, it can appear on the left and right side
# sometimes this is called updating a value, but is also assigning a value
# the equal sign means ASSIGN something on the left to the value on the right
# sometimes I may leave the IDLE arrows in! text will be bolded

>>> number = 3
>>> number
3
>>> number = 2 * number   # variable on right is processed first
>>> number
6
>>> number = number * number
>>> number
36
```

Variables in Python

```
# Augmented assignment can be done using Python shorthand
# this is done a lot in loops
score = 50
score = score + 20
score
70

# shorthand : augmented assignment
score = 50  # reassign score
score += 20
score
70
```

Variables in Python

```
# Augmented assignment can be done using Python shorthand
# multiplication is a little tricky
d = 2
d *= 3 + 4
d
14

# means
d = d * (3 + 4)
or
d *= (3 + 4)
```

Variables in Python

```
# Very tricky:
```

Symbol	Example	Result
+=	x = 7 x += 2	x refers to 9
-=	x = 7 x -= 2	x refers to 5
*=	x = 7 x *= 2	x refers to 14
/=	x = 7 x /= 2	x refers to 3.5
//=	x = 7 x //= 2	x refers to 3
%=	x = 7 x %= 2	x refers to 1
**=	x = 7 x **= 2	x refers to 49

Table 3—Augmented Assignment Operators

Error messages

```
>>> 3 + moogah
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    3+ moogah
NameError: name 'moogah' is not defined
2 +
SyntaxError: invalid syntax

5 = x
SyntaxError: can't assign to literal
```

Multi-line statements

```
# multi-line statements are allowed if extra left parenthesis on line,
# or a backslash is at the end
(2 +
3)
5
2 + \
3
5
```

Homework 1 due before class Wednesday

Part 1: Use Python, not R, to calculate these answers

1.1 Use R as you would a calculator to find numeric answers to the following:

1. $1+2(3+4)$
2. 4^3+3^{2+1}
3. $\sqrt{(4+3)(2+1)}$
4. $\left(\frac{1+2}{3+4}\right)^2$

1.2 Rewrite these R expressions as math expressions, using parentheses to show the order in which R performs the computations:

1. `2 + 3 - 4`
2. `2 + 3 * 4`
3. `2/3/4`
4. `2^3^4` Python: 2 ** 3 ** 4

1.3 Use R to compute the following

$$\frac{1+2\cdot 3^4}{5/6-7}.$$

1.4 Use R to compute the following

$$\frac{0.25-0.2}{\sqrt{0.2\cdot(1-0.2)/100}}$$

Homework 1

Part 2:

Using algebra and Python, figure out at what temperature Celsius degrees equals Fahrenheit degrees.

For next time (on syllabus)

"Read" chapter 3 in Gries (RUN programs while reading)

Read the PEP 8 Style Guide (PDF on blackboard under Resources)