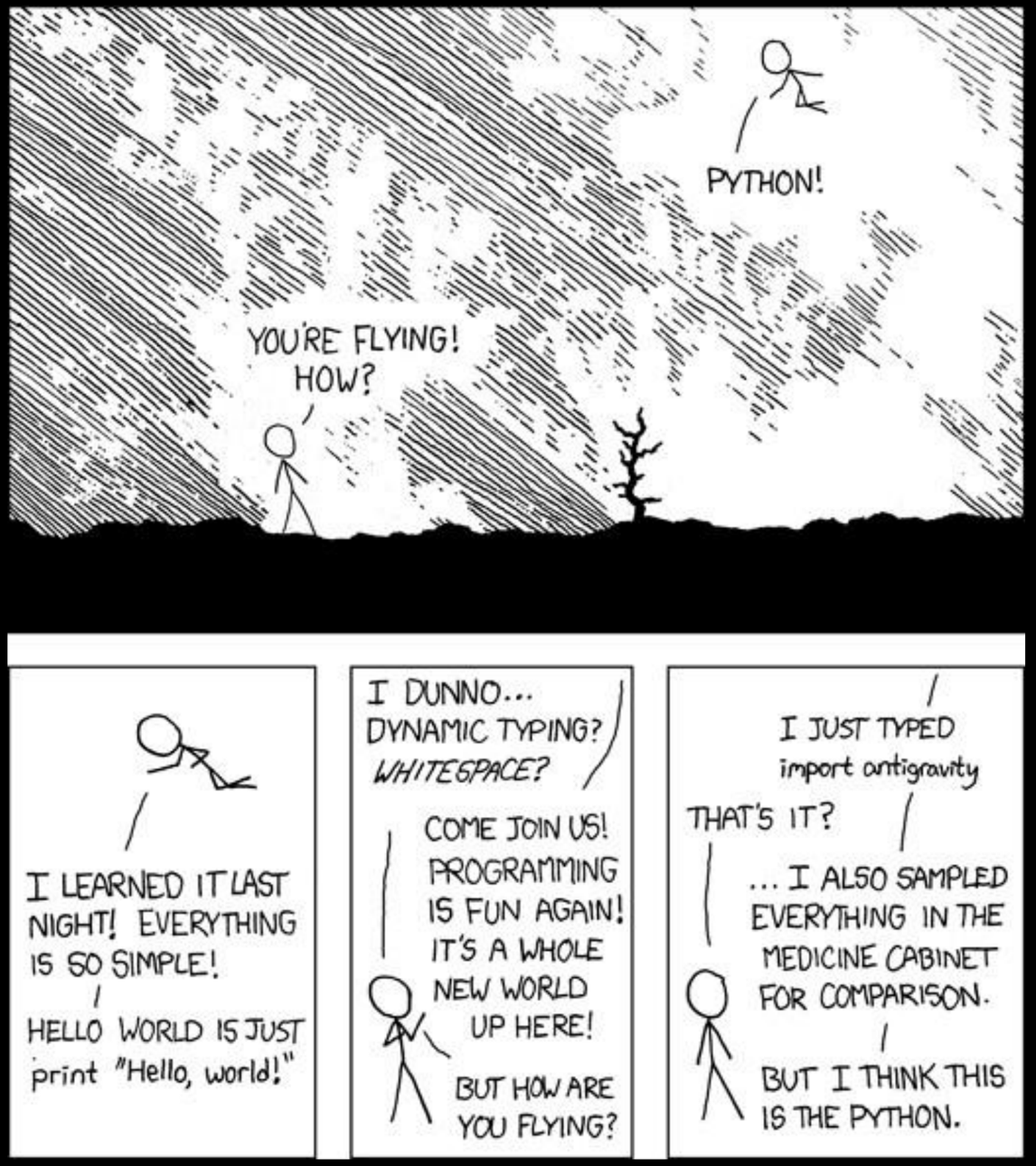
Python – Introduction


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

What is Python?

-- The python interpreter
(interactive)
    -- You type one expression at a time
    -- The interpreter evaluates the expression and prints it

(batched)

-- Python evaluates all the statements in the file, in order
-- Python does not print their values (but does execute print statements)

How to Run It?

-- Save python (filename.py) code somewhere on HD
-- Open a terminal
-- python /path/to/where/you/saved/the/filename.py

Based on data, here are programming languages listed next to their average annual salary from lowest to highest:

12. **PERL** - $82,513
11. **SQL** - $85,511
10. **Visual Basic** - $85,962
9. **C#** - $89,074
8. **R**- $90,055
7. **C** - 90,134
6. **JavaScript** - $91,461
5. **C++** - $93,502
4. **JAVA** - $94,908
3. **Python** - $100,717
2. **Objective C** - $108,225
1. **Ruby on Rails** - $109,460

Read more: http://www.businessinsider.com/best-tech-skills-resume-ranked-salary-2014-11#ixzz3ayquuMLW

#Thing to KNOW

-- Keep up with the readings!
-- Just reorganized my email accounts!
-- Subject: ECSU-URE15: [subject of message content]

-- Exercises can be completed in GROUPS

-- Assignments NEED TO BE DONE individually!

-- Task sheet completion (signed by Jeff Woods)

-- Session Website: www.cresis.ku.edu/~jemitchell/ECSU-URE15

We got to talk about Importing

-- Modules are files containing Python definitions (ex. math.py)

-- A module's definitions can be imported into other modules by using Import name (ex. import math)

-- The module's name is available as a global variable value

-- In order to access a module's function, "name.function()"
Example:

```
import math
math.sin(x)
```

Constants

-- Fixed values, such as numbers, letters, and strings are called “constants” - because their value does not change

-- Numeric constants

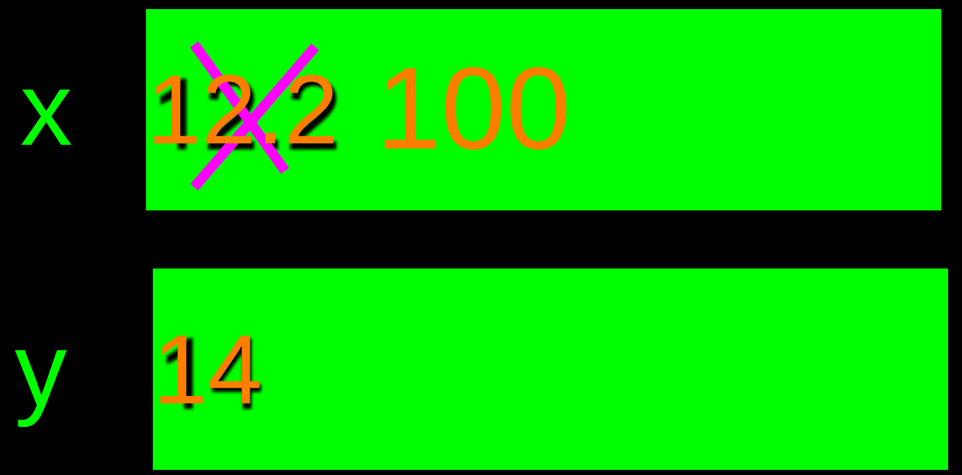
-- String constants use single-quotes (') or double-quotes (")

```
>>> print 123
123
>>> print 98.6
98.6
>>> print 'Hello world'
Hello world
```

Variables

-- A variable is a named place in memory, which allows for data to be stored and retrieved using the variable “name”

x = 12.2
y = 14
x = 100



Python Variable Name Rules

-- Must start with either a letter or underscore _ andconsist of letters, numbers, and underscores

-- Case Sensitive

-- Good: spam eggs spam23 _speed

-- Bad: 23spam #sign var.12

-- Different: spam SpamSPAM

Reserved Words

You can not use reserved words as variable names / identifiers

and del for is raise
assert elif from lambda return
break else global not try
class except if or while
continue exec import pass yield
def finally in print

Sentences or Lines

x = 2 ⟵ assignment statement

x = x + 2 ⟵ assignment with expression

print x ⟵ print statement

Variable Operator Constant Reserved Word

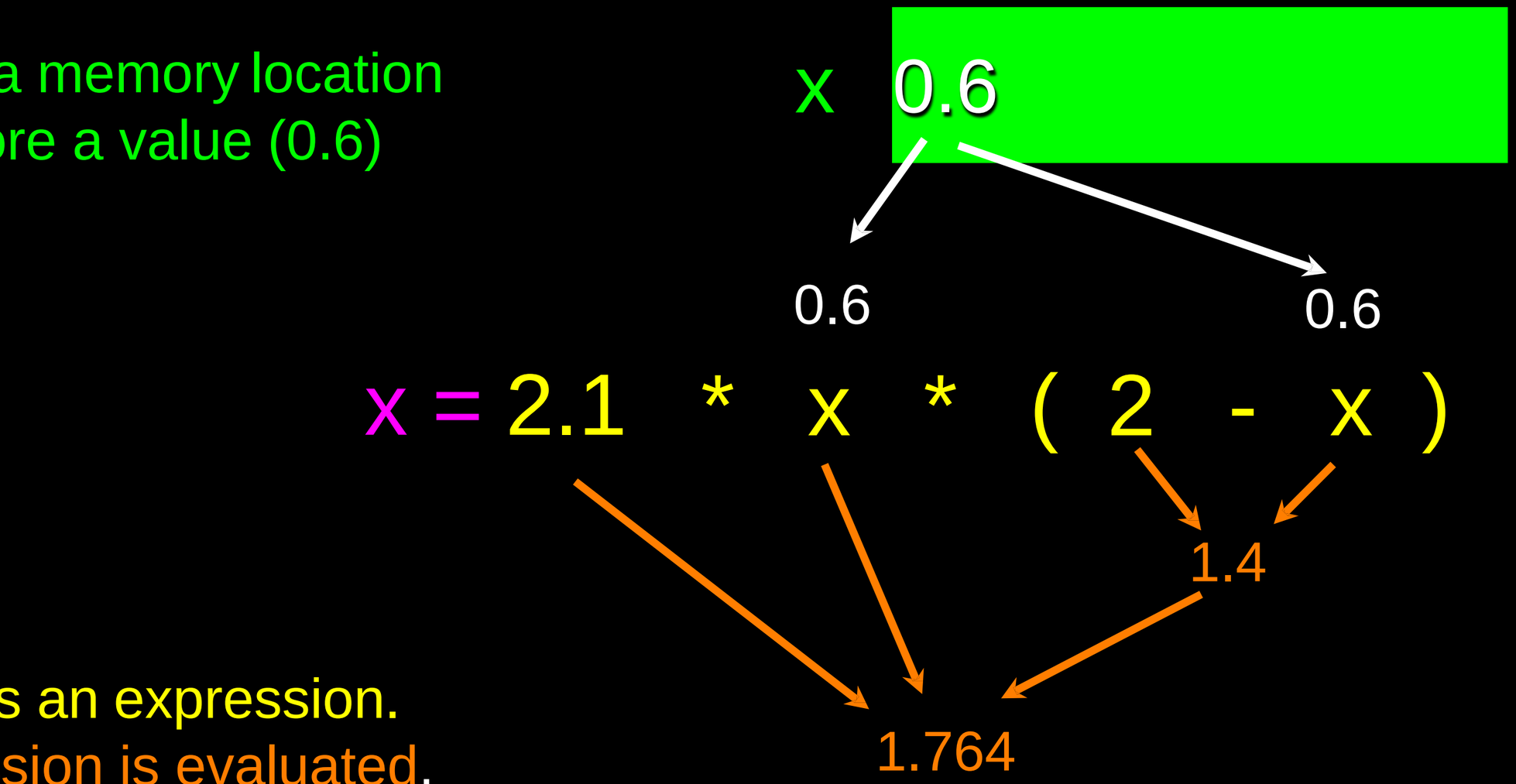
Assignment Statements

-- Assign a value to a variable using the assignment statement (=)

-- An assignment statement consists of an expression on the right hand side anda variable to store the result

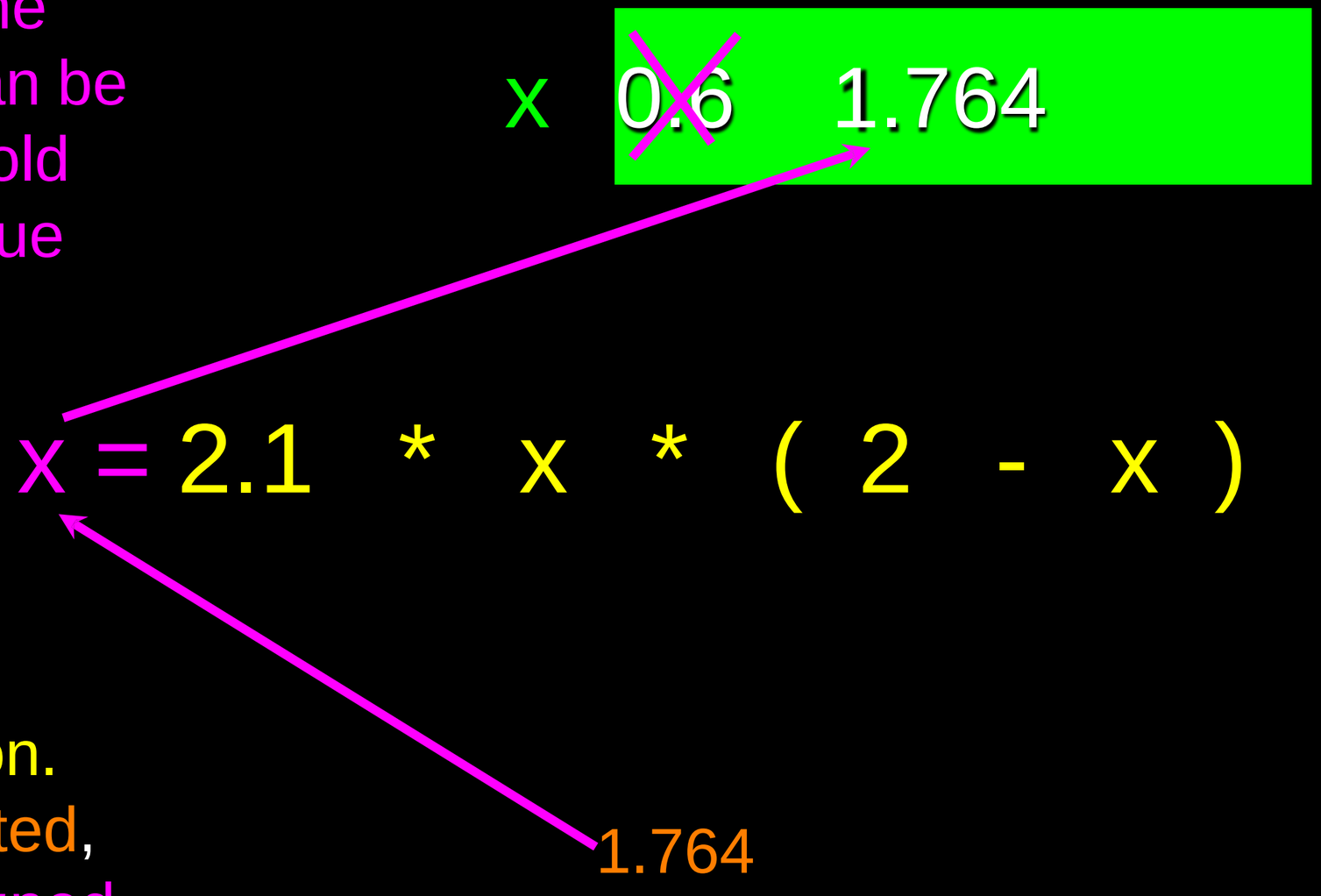
```
x = 2.1 * x * ( 0.5 - x )
```

A variable is a memory location used to store a value (0.6)



Right side is an expression. Once expression is evaluated, the result is placed in (assigned to) x.

A variable is a memory location used to store a value.The value stored in a variable can be updated by replacing the old value (0.6) with a new value (1.764).



Right side is an expression. Once expression is evaluated, the result is placed in (assigned to) the variable on the left side (i.e. x).

Numeric Expressions

-- Use “computer-speak” to express classic math operations

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
%	Remainder

Numeric Expressions

```
>>> xx = 2
>>> xx = xx + 2
>>> print xx
4
>>> yy = 440 * 12
>>> print yy
5280
>>> zz = yy / 1000
>>> print zz
5
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print kk
3
>>> print 4 ** 3
64
```

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
%	Remainder

Order of Evaluation

-- Sequence of operators - python must know how to operate

-- This is called “operator precedence”

-- Which operator “takes precedence” over the others?

x = 1 + 2 * 3 - 4 / 5 ** 6

Operator Precedence Rules

-- Highest precedence rule to lowest precedence rule

-- Parenthesis are always respected

-- Exponentiation (raise to a power)

-- Multiplication, Division, and Remainder

-- Addition and Subtraction

-- Left to Right

Parenthesis
Power
Multiplication
Addition
Left to Right

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print x
11
>>>
```

Parenthesis
Power
Multiplication
Addition
Left to Right

1 + 2 ** 3 / 4 * 5

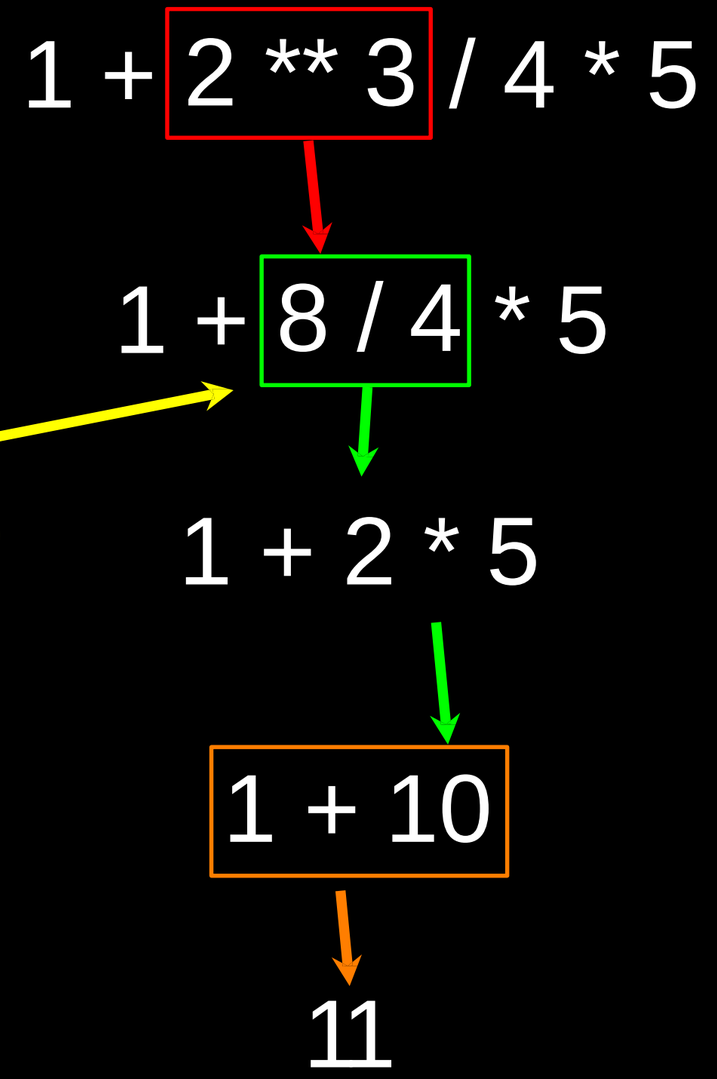
1 + 8 / 4 * 5

1 + 2 * 5

1 + 10

11

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print x
11
>>>
```



Note 8/4 goes before 4*5 because of the left-right rule.

Parenthesis
Power
Multiplication
Addition
Left to Right

Python Integer Division is Weird!

-- Integer division truncates

-- Floating point division produces floating point numbers

This changes in Python 3.0

```
>>> print 10 / 2
5
>>> print 9 / 2
4
>>> print 99 / 100
0
>>> print 10.0 / 2.0
5.0
>>> print 99.0 / 100.0
0.99
```

Mixing Integer and Floating

-- If there is an integer operand and the other operand is a floating point the result is a floating point

-- The integer is converted to a floating point before the operation

```
>>> print 99 / 100
0
>>> print 99 / 100.0
0.99
>>> print 99.0 / 100
0.99
>>> print 1 + 2 * 3 / 4.0 - 5
-2.5
>>>
```

What does “Type” Mean?

-- In Python variables, literals, and constants have a “type”

-- Python knows the difference between an integer number and a string

-- For example “+” means “addition” if something is a number and “concatenate” if something is a string

```
>>> ddd = 1 + 4
>>> print ddd
5
>>> eee = 'hello ' + 'there'
>>> print eee
hello there
```

concatenate = put together

Type Matters

-- Some operations are prohibited

-- You cannot “add 1” to a string

-- Determine type by using the type() function

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> type(eee)
<type 'str'>
>>> type('hello')
<type 'str'>
>>> type(1)
<type 'int'>
>>>
```

Several Types of Numbers

-- Numbers have two main types

-- integers are whole numbers: -14, -2, 0, 1, 100, 401233

-- floating point numbers have decimal parts:-2.5 , 0.0, 98.6, 14.0

-- There are other number types - they are variations on float and integer

```
>>> xx = 1
>>> type (xx)
<type 'int'>
>>> temp = 98.6
>>> type(temp)
<type 'float'>
>>> type(1)
<type 'int'>
>>> type(1.0)
<type 'float'>
>>>
```

Type Conversions

-- In an integer and floating point expression, the integer is implicitly converted to a float

-- You can control this with the built in functions int() and float()

```
>>> print float(99) / 100
0.99
>>> i = 42
>>> type(i)
<type 'int'>
>>> f = float(i)
>>> print f
42.0
>>> type(f)
<type 'float'>
>>> print 1 + 2 * float(3) / 4 - 5
-2.5
>>>
```

String Conversions

-- Use int() and float() to convert between strings and integers

-- You will get an error if the string does not contain numeric characters

```
>>> sval = '123'
>>> type(sval)
<type 'str'>
>>> print sval + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int'
>>> ival = int(sval)
>>> type(ival)
<type 'int'>
>>> print ival + 1
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
```

User Input

-- Instruct Python to pause and read data from the user using the raw_input function

-- The raw_input function returns a string

```
nam = raw_input( ‘Who are you?’
print 'Welcome', nam
```

```
Who are you? ECSU REU
Welcome ECSU REU
```

Converting User Input

-- To read a number from the user, it must be convert from a string to a number using a type conversion function

```
inp = raw_input( ‘Europe floor?’ )
usf = int(inp) + 1
print 'US floor', usf
```

```
Europe floor? 0
US floor 1
```

Comments in Python

-- Anything after a # is ignored by Python

-- Why comment?

-- Describe what is going to happen in a sequence of code

-- Document who wrote the code or other information

-- Turn off a line of code (debugging purposes) - perhaps temporarily

```
# Get the name of the file and open it
name = raw_input('Enter file:')
handle = open(name, 'r')
text = handle.read()
words = text.split()

# Count word frequency
counts = dict()
for word in words:
   counts[word] = counts.get(word,0) + 1

# Find the most common word
bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

# All done
print bigword, bigcount
```

String Operations

-- Some operators apply to strings

-- + implies “concatenation”

-- * implies “multiple concatenation”

-- Python knows when it is dealing with a string or a number and behaves appropriately

```
>>> print 'abc' + '123'
Abc123
>>> print 'Hi' * 5
HiHiHiHiHi
>>>
```

Mnemonic Variable Names

-- How to choose variable names? -- there is a bit of “best practice”

-- Variables should help determine what values are stored in them (“mnemonic” = “memory aid”)

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print x1q3p9afd
```

```
a = 35.0
b = 12.50
c = a * b
print c
```

What is this code doing?

```
hours = 35.0
rate = 12.50
pay = hours * rate
print pay
```

Happy Coding!!!

Develop a program, which asks the user to enter their name and their age! Also, print a message addressing the user telling him or her the year that person will turn 75 years old.