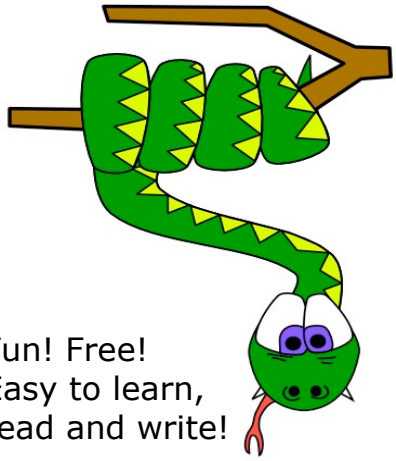
Python



Fun! Free!
Easy to learn,
read and write!

Introduction

- a programming language like Java and C++.
- a high-level language that translate a set of instructions into machine language, which is represented by 0s and 1s.

Getting Python

At home:
You want to download the latest version, which is 2.5.2.
Go to http://www.python.org/download/

Assuming you are using Windows, after you have installed it, go to Start > All > Programs > Python > IDLE.

In the lab:
Go to Start > All Programs > Python 2.5 > IDLE.

Now try this out!

In the window that appears, you should see the python prompt >>>. This is asking you to enter python commands. Type in the following to see what happens.

To display or output:

```
>>> print "Hello World"
>>> print 'Hello World'
>>> print Hello World
```
Q1: Why is there an error?

To do arithmetics (use Python as a calculator):

```
>>> print 25+38
>>> print 3*6
>>> print 3/6
>>> print 3.0/6.0
```
Q2: Why does 3/6=0?

```
>>> print 2+4*2
```
Q3: Why is the value above 10 but not 12?

```
>>> print 2**2
>>> print 2**3
```

To connect or link 2 or more words:

```
>>> print 'abc'+'def'
>>> print 'joe'+' '+'smith'
>>> print '25'+'58'
```

To know what type a value is:

```
>>> type(21)
>>> type(“Hello World”)
>>> type('Hello Word')
```

```
>>> type(2.5)
>>> type(“15”)
>>> type(“2.5”)
```

To link to another piece of data (variables):

```
>>> words = "Hello World!"
>>> print words
>>> firstname = 'joe'
>>> lastname = 'smith'
>>> print firstname + lastname
>>> print “%s is my name. “ % ("Jane”)
>>> print “%d is my favorite number. “ % (7)
>>> print "%s is number %d! " % ("Python", 1)
>>> x=2
>>> print x+5
>>> print x-0
>>> print x*7
>>> print 'x'*7
```

Q4: Explain why.

```
>>> print x/2
>>> y=3
>>> print x+y
>>> print x-y
>>> print x*y
>>> print x/y
```

To obtain user input from command-line:

```
>>> result = raw_input()
```

(Python will wait for you (the end-user) to enter something. Enter some text and hit Enter.)

```
>>> print result
>>> x = raw_input('Please enter a string: ')
>>> print x
>>> name = raw_input("Enter your name here: ")
>>> age = raw_input("Enter your age here: ")
>>> print "Your name is:", name
>>> print "And you are", age
```

To combine variables, expressions and statements:

```
>>> print “The distance from my house to my school is “, 10, “miles.”
>>> minute = 5
>>> print minute, “ minute is equals to “, minute*60, “seconds.”
```

More challenges!

Looping – The art of repeating itself

You can use loops when you want to repeat a set of instruction multiple times.

while loop pseudocode:

```
while (expression):
    # statements to execute while loop expression is True
else:
    # statements to execute when loop expression is False
```

```
>>> i = 0
>>> while(i < 10):
        i = i + 1
        print i
```

(Hit Enter twice)

```
>>> x = 10
>>> while (x > 0):
        x = x - 1
        print x
```

(Hit Enter twice)

```
>>> x = 10
>>> while (x != 0):
        print x
        x = x - 1
        print "wow, we've counted x down, and now it equals", x
```

(Hit Enter twice)

for loop pseudocode

```
for item in container:
    # action to repeat for each item in the container
else:
    # action to take once we have finished the loop.
```

```
>>> for i in range(1,11):
        print i
```
(Hit Enter twice)

```
>>> for i in range(1,1000):
        print i
```
(Hit Enter twice)

```
>>> for x in 'summer':
        print x
```
(Hit Enter twice)

```
>>> for word in ('one','word', 'after', 'another'):
        print word
```
(Hit Enter twice)

The if statement

Execute a block of statements depending on some condition.

<u>if statement pseudocode</u>

```
if(expression one):
    # Action to take if expression one evaluates True
else:
    # Action to take if all expression one evaluates False
```

```
>>> i = 8
>>> if(i % 2):
        print "Odd Number"
else:
        print "Even Number"
(Hit Enter twice)

>>> z = 4
>>> if (z > 70):
        print "Something is very wrong"
elif (z < 7):
        print "This is normal"
(Hit Enter twice)

>>> i = -8
>>> if(i > 0):
        print "Positive Integer"
elif(i < 0):
        print "Negative Integer"
```

```
else:
        print "Zero"
```
(Hit Enter twice)

Combo – while loop and if statement

```
>>> a = 10
>>> while a > 0:
            print a
            if (a > 5):
                print "Big number!"
             elif ((a % 2) != 0):
                print "This is an odd number"
                print "It isn't greater than five, either"
             else:
                print "this number isn't greater than 5"
                print "nor is it odd"
                print "feeling special?"
            a = a - 1
            print "we just made 'a' one less than what it was!"
            print "and unless a is not greater than 0, we'll do the
            loop again."
```
(Hit Enter twice)

List of Elements

To add elements to a list

```
>>> list = []              # gives an empty list you can add elements to
>>> list.append('a')
>>> print list
```

you can also initialize the elements of a list and access each element

```
>>> list2 = ['a', 'b', 'c']
>>> print list2[0]              # in computer science we count from 0
>>> print list2[2]              # list2[2] is the 3rd element not the 2nd
>>> print list2[3]
```

Q5: Explain why there is an error?

getting the length of a list or the number of elements

```
>>> print len(list2)
```

to print the elements of a list using a while loop:

```
>>> list3 = ['sarah', 'bob', 'joe']
>>> i = 0
>>> while i < len(list3)
        print list[i]
        i = i + 1
```

(Hit Enter twice)

Questions

1. Create the list [0,1,2,3,4,5,6,7,8,9] without using append.

2. Create the same list starting with an **empty list** and using a **loop**.

3. Make one change to your code from number 2 to create the list [0,2,4,6,8].

4. Write a loop that creates the list [1,3,5,7,9] using an **if statement.**

Writing a Program in a File

By writing a program in a file, we can run our code all at once.

1. To do so, open any text editor and enter the following four lines in a file named 'hello.py'. Put the lines on the left-margin-- no tabs or spaces.

```
print "hello"
print "world"
name = "jive"
print name
```

2. Save the file as hello.py

3. At the terminal window, enter:

    python hello.py

Note that you should enter the above at the operating system prompt, not the Python prompt. If you're still within the Python interpreter, exit with control-D.

Executing the above should result in the following being printed:

```
hello
world jive
```

Challenge Questions

1. Write an **if statement** that translates scores to grades. You can follow the guide below:

    A: 90 - 100
    B: 80 – 90
    C: 70 – 80
    D: 60 – 70
    E: 0 – 60

2. Write a **for loop** to produce the following output:

```
*
**
***
****
*****
```

3. Write a **while loop** to produce the following output:

```
*****
****
***
**
*
```

4. Write a **program** (in a file 'temp.py') that converts Celsius to Fahrenheit:

Sample:

> Enter a temperature in Celcius: **20**
> 20 C = 68 F

Steps:

1. Ask for input from the user
2. Compute the temperature in Fahrenheit: **F = (9/5)*C+32**
3. Print the result in the form: _ C = _ F

(Use what you know about dividing integers and floating point numbers to compute the correct temperature)

5. **Modify** your program to convert Fahrenheit to Celsius as well:

Sample:

> Enter a temperature: **68**
> Convert to (F)ahrenheit or (C)elsius: **C**
> 68 F = 20 C

Steps:

1. Ask for input from user
2. Use an **if-else statement** to do the correct conversion**:**

For example:

```
if (input == 'a')
        …
        do something
        …
else
        …
        do something else
        …
```

3. Compute the temperature in Fahrenheit or Celsius:

   F = (9/5)*C+32
   C = (5/9)*(F-32)

4. Print the Result

Media Programming Introduction

JES is a development environment for Python.

If you don't have JES on your computer, you can download it at **http://coweb.cc.gatech.edu/mediaComp-plan/94**. If you're working in a USF CS lab, JES is already installed. On Linux, you can find it in Applications > Programming > JES. On Windows, look for the frog in 'All Programs'.

JES has two main windows. The bottom panel is an interactive interpreter. This allows you to try a single command at a time.

Usually you'll use the **top window**, which is a text editor. You write the program there, then click 'Load program' to run it.

Tutorial

1. With JES you work with pictures. So the first step is to download an image from the web into your H: directory. With most browsers, you can right-click an image and choose "Save Images as" or something similar.

2. Open the JES programming environment and type in the following mini-program into the top panel editor.

   a. Ask the end-user to choose a graphic file
   b. Create a picture in memory holding the pixels of the image
   c. Show the picture

```
filename = pickAFile()
pic = makePicture(fileName)
show(pic)
```

Enter the lines exactly as shown above, then click File > Save Program. Save in imageSample.py.

3. Run your program by clicking the "Load Program" near the middle of the JES Window. You (the end-user) will be prompted to choose a .jpg or other graphic file. Choose an image from one of your folders, and the program should render the picture. That is all this first program does.

4. Now we'll modify your program and have it change the image, just as a human might do in Photoshop. For now, let's just draw some text and a line on our picture. Add the following below the previous code in your sample.py file:

a. Draw a line going from coordinate (0,0) to (200,200)
b. Draw some text at coordinate (100,100)
c. Repaint the picture with the changes

```
addLine(pic,0,0,200,200)
addText(pic,100,100,"hey dude”)
repaint(pic)
```

Do you see the line on the picture? Try different numbers and text for the parameters. What happens?

5. Note that the manipulations take place on an in-memory (RAM) version of the image. You can save the modified image to a file:

```
writePictureTo(pic,'/home/wolber/pics/out.jpg')
```

If you're using Windows, you need to put an 'r' in front of the file path. For instance:

```
writePictureTo(pic,r"H:\out.jpg")
```

Add the appropriate line to the bottom of your program, rerun your program, then go into the folder you specified and see if the new file has been created and if the image inside it is the modified picture.

6. Play around with some of the Photoshop-like functions that JES provides. To view a list of the functions, select Help ! Understanding Pictures > Picture Functions in JES. This specification describes the functions you can call.

Calling Functions: Parameters and Return Values

A function does some work for us. We can send a function some data, and it will perform some task and send us something back.

Consider the function sqrt. A program can send it a number and get something back:

```
result = sqrt(16)
```

When you call a function, you sometimes send it some data that it needs to do its job. We call such data a **parameter**. 16 is the parameter sent to sqrt in the sample above. Parameters are placed in parenthesis after the function name.

Sometimes functions return you some data when they complete. We call such data a **return value**. We say that the calling function catches the return value. In the sample above, the return value is caught by the variable result.

Some functions are built-in to the Python language. sqrt is one example.

Other functions are just sub-programs written by some programmer--named pieces of code that you can call. Anyone can write such functions using the 'def' statement. For example, I could create a function cube:

```
def cube(x):
    return x*x*x
```

and then call it:

```
result = cube(3)
```

In the JES Media Introduction tutorial, you called some functions:

```
filename = pickAFile()
pic = makePicture(fileName)
show(pic)

addLine(pic,0,0,200,200)
addText(pic,100,100,"hey dude")
repaint(pic)
```

The makePicture function has one parameter, fileName. You tell makePicture a fileName, and it loads a picture for you (in memory). That picture is returned to your program and 'caught' in the variable pic.

In-Class Worksheet

1. What are the parameters of the other JES function calls above? Which need to catch a return value?

2. The JES Help provides a list of function definitions, also known as an application programmer interface (API). Open JES and find the definition for the function addLine. What are the names for the five parameters it expects sent to it? How would you call it to draw a vertical line all the way down the middle of a picture?

Using Python to Send Email

#Sample program 1

```
import smtplib

name = "Sami Rollins"
sender = "srollins@cs.usfca.edu"
recipient = "srollins@gmail.com"
session = smtplib.SMTP("nexus")
msg = "To: " + name + " <" + recipient + ">\n" + "This is a test
message."

try:
   smtpresult = session.sendmail(sender, recipient, msg)
except:
   print "failure"
```

#Sample program 2

```
import webbrowser

webbrowser.open("http://www.google.com")

print "done..."
```

Exercises

1. Write a program to prompt the user for a word and print "You entered: " followed by the word that the user entered.

2. Write a program to prompt the user for a URL (using raw_input) and launch a browser window displaying that URL.

3. Write a program to prompt the user for a name, sender email address, and receiver email address and send an email message to the specified address.