CSci 127: Introduction to Computer Science

hunter.cuny.edu/csci

Frequently Asked Questions

- What does `len()` mean?
  `len(s)` *gives the length (# of items or chars.). Ex:* `len("hi!!")` *is 4.*

- Why do you sometimes use parenthesis and other times brackets?
  Parenthesis are for functions: ex: `print("CUNY")`

  Brackets are used for access items in a list or string: ex: `message[3]`

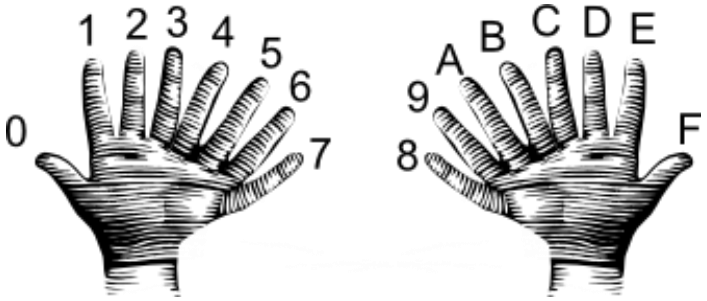
- *The colon, :, gives a slice, substring or sublist, ex:* `myString[3:5]`

Recap: Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.
- Can specify by numbers (RGB):
  - Fractions of each:
    e.g. `(1.0, 0, 0)` is 100% red, no green, and no blue.
  - 8-bit colors: numbers from 0 to 255:
    e.g. `(0, 255, 0)` is no red, 100% green, and no blue.
  - Hexcodes (base-16 numbers)...

Recap: Hexadecimal



00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

In Pairs or Triples...

Let's start with loops & slices:

```
word = "Hunter"
for i in range(2,10,3):
    for c in word:
        print(i,c, end = "")
    print()

pali = "a man a plan a canal Panama"
print(pali[0], pali[-1])
print(pali[2:5], pali[-4:-1])

qPop = [152999,284041,469042,1079129,1297634,
    1550849,1809578,1986473,1891325,1951598,
    2229379,2230722]
print("Queens population in 1900:", qPop[0])
print("Since 2000:", qPop[-3:len(qPop)])
```

Python Tutor

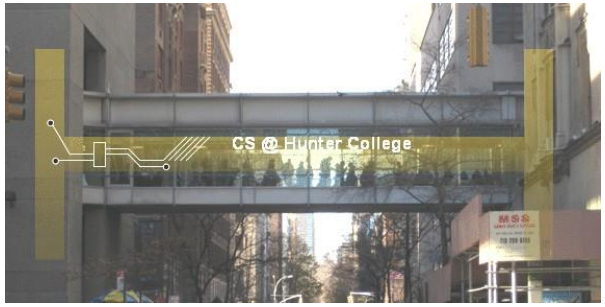
```
word = "Hunter"
for i in range(2,10,3):
    for c in word:
        print(i,c, end = "")
    print()

pali = "a man a plan a canal Panama"
print(pali[0], pali[-1])
print(pali[2:5], pali[-4:-1])

qPop = [152999,284041,469042,1079129,1297634,
    1550849,1809578,1986473,1891325,1951598,
    2229379,2230722]
print("Queens population in 1900:", qPop[0])
print("Since 2000:", qPop[-3:len(qPop)])
```

(Demo with `pythonTutor`)

Design Question: Cropping Images

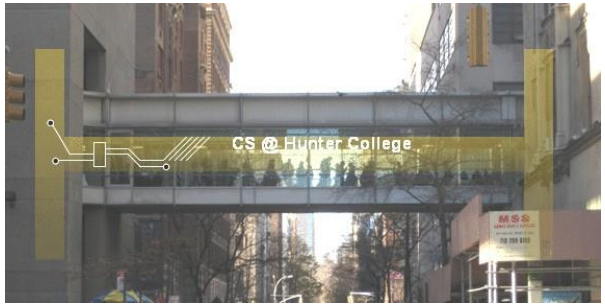


Design Question: Design an algorithm that will crop an image.

- First: specify what the inputs & outputs for the algorithm .
- Next: write pseudocode.
- translate to Python

**This image has 287 rows, 573 columns

Design: Cropping Images



- **First: specify inputs/outputs.**
  Input file name,
  output file name,
  upper, lower, left, right ("bounding box")

Design: Cropping Images

- First: specify inputs/outputs. *Input file name, output file name, upper, lower, left, right ("bounding box")*
- Next: write pseudocode.
  1. Import `numpy` and `pyplot`.
  2. Ask user for file names and dimensions for cropping.
  3. Save input file to an array.
  4. Copy the cropped portion to a new array.
  5. Save the new array to the output file.
- **translate to Python.**

Design: Cropping Images

```
#Name:  CSci 127 Teaching Staff
#Date:  Fall 2017
#This program loads an image, displays it, and then creates, displays,
#    and saves a new image that has only the red channel displayed.

#Import the packages for images and arrays:
import matplotlib.pyplot as plt
import numpy as np

inImg = input('Enter input image: ')
img = plt.imread(inImg)  #Read in image from csBridge.png
plt.imshow(img)          #Load image into pyplot
plt.show()               #Show the image (waits until closed to continue)

outImg = input('Enter out image: ')
t = int(input('Enter top:'))
b = int(input('Enter bottom:'))
l = int(input('Enter left: '))
r = int(input('Enter right: '))

img2 = img[t:b,l:r]      #Slice the original array by dimensions entered

plt.imshow(img2)         #Load our new image into pyplot
plt.show()               #Show the image (waits until closed to continue)

plt.imsave(outImg, img2) #Save the image we created to the out file.
```

Relational Operators

Used to compare numbers to determine relative order

Operators:

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

Relational Expressions

Boolean expressions – `true` or `false`

Examples:

`12 > 5` is `true`

`7 <= 5` is `false`

if `x` is 10, then

`x == 10` is `true`,

`x != 8` is `true`, and

`x == 8` is `false`

Relational Expressions

Can be assigned to a variable:

```
result = x <= y;
```

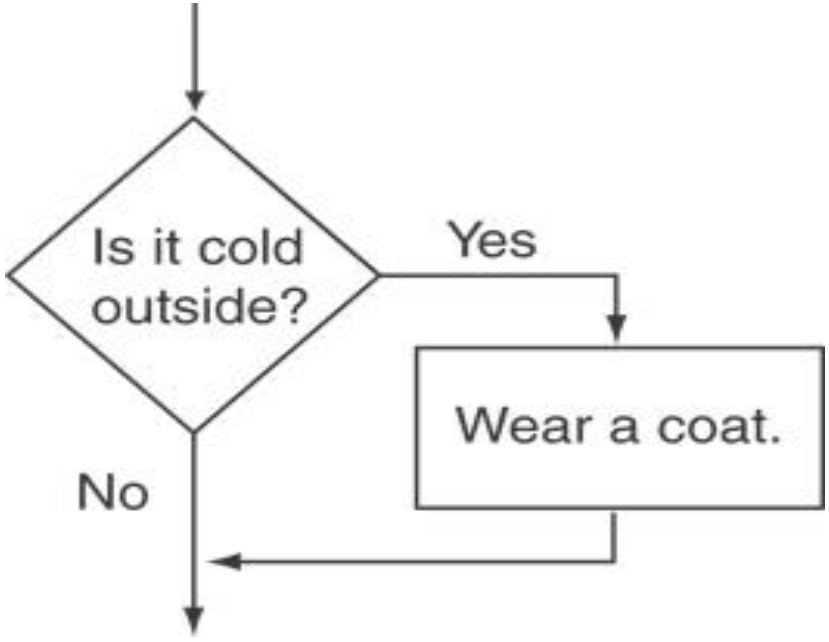
Do not confuse `=` and `==`

The `if` Statement

Allows statements to be conditionally executed or skipped over
Models the way we mentally evaluate situations:

"If it is raining, take an umbrella."
"If it is cold outside, wear a coat."

Flowchart for Evaluating a Decision



The `if/else` statement

- Provides two possible paths of execution
- Performs one statement or block if the *`expression`* is true, otherwise performs another statement or block.

The **`if/else`** statement

General Format:

```
if expression:
    statement1
else:
    statement2
```

`if/else`-What Happens

To evaluate:

```
if expression:
    statement1
else:
    statement2
```

If the *`expression`* is `true`, then *`statement1`* is executed and *`statement2`* is skipped.

If the *`expression`* is *`false`*, then *`statement1`* is skipped and *`statement2`* is executed.

The `if/elif` Statement

- Tests a series of conditions until one is found to be true
- Often simpler than using nested `if/else` statements
- Can be used to model thought processes such as:

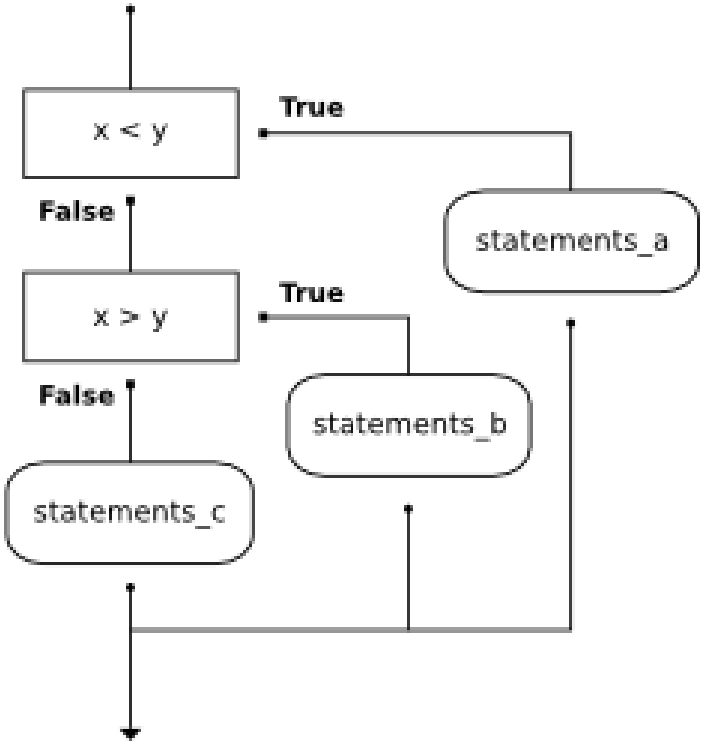
  "If it is raining, take an umbrella,
  else, if(elif) it is windy, take a hat,
  else, take sunglasses”

if/elif else Format

```
if expression:
      statement1
elif expression:
      statement2
      .
      . // other else ifs
      .
elif expression:
      statementn
else:
      statement
```

Decisions

```
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x and y must be equal")
```



(This was just a first glance, will do much more on decisions over the next several weeks.)

In Pairs or Triples...

Predict what these will do (novel concepts):

```python
yearBorn = int(input('Enter year born: '))
if yearBorn < 1946:
    print("Greatest Generation")
elif yearBorn <= 1964:
    print("Baby Boomer")
elif yearBorn <= 1984:
    print("Generation X")
elif yearBorn <= 2004:
    print("Millennial")
else:
    print("TBD")

x = int(input('Enter number: '))
if x % 2 == 0:
    print('Even number')
else:
    print('Odd number')
```

```python
import turtle

tess = turtle.Turtle()
myWin = turtle.Screen()     #The graphics window
commands = input("Please enter a command string: ")

for ch in commands:
    #perform action indicated by the character
    if ch == 'F':            #move forward
        tess.forward(50)
    elif ch == 'L':          #turn left
        tess.left(90)
    elif ch == 'R':          #turn right
        tess.right(90)
    elif ch == '^':          #lift pen
        tess.penup()
    elif ch == 'v':          #lower pen
        tess.pendown()
    elif ch == 'B':          #go backwards
        tess.backward(50)
    elif ch == 'r':          #turn red
        tess.color("red")
    elif ch == 'g':          #turn green
        tess.color("green")
    elif ch == 'b':          #turn blue
        tess.color("blue")
    else:                    #for any other character
        print("Error: do not know the command:", c)
```

Python Tutor

```python
yearBorn = int(input('Enter year born: '))
if yearBorn < 1946:
    print("Greatest Generation")
elif yearBorn <= 1964:
    print("Baby Boomer")
elif yearBorn <= 1984:
    print("Generation X")
elif yearBorn <= 2004:
    print("Millennial")
else:
    print("TBD")

x = int(input('Enter number: '))
if x % 2 == 0:
    print('Even number')
else:
    print('Odd number')
```

(Demo with `pythonTutor`)

IDLE

```python
import turtle

tess = turtle.Turtle()
myWin = turtle.Screen()     #The graphics window
commands = input("Please enter a command string: ")

for ch in commands:
    #perform action indicated by the character
    if ch == 'F':           #move forward
        tess.forward(50)
    elif ch == 'L':         #turn left
        tess.left(90)
    elif ch == 'R':         #turn right
        tess.right(90)
    elif ch == '^':         #lift pen
        tess.penup()
    elif ch == 'v':         #lower pen
        tess.pendown()
    elif ch == 'B':         #go backwards
        tess.backward(50)
    elif ch == 'r':         #turn red
        tess.color("red")
    elif ch == 'g':         #turn green
        tess.color("green")
    elif ch == 'b':         #turn blue
        tess.color("blue")
    else:                   #for any other character
        print("Error: do not know the command:", c)
```

(Demo with `IDLE`)