Types

- Concept of type in programming languages.
- Knowledge and skills with types expected in this course ("What will be on the exams?")
- Why types are useful?
- Plus: Some Java/Python differences wrt. types.
Definition of Type

• A type is a name (like bool, int, string, long, float NoneType, class) obvious, plus
  – a set of values (more abstract...)

• T'is interesting because: Types organize and help us use operations and functions.
  – They help us pick the functions& avoid mistakes.
Examples of types

- (NoneType, { None }). The set has 1 value!
  - **DO THIS NOW:**
    
    ```
    >>> type( None )
    ```

- (bool, { True, False } ) The set has 2 values.
  - **DO THIS NOW:**
    
    ```
    >>> type( True )
    >>> type( False )
    ```
int is a type

- The set is the counting “numbers in the range -2147483648 [i.e. \(-2^{31}\)] through 2147483647)
  - Python: The range may be larger on machines with a larger natural word size, but not smaller”.
  - [Java: This range is FIXED by Java's definition.]
  - Python: If arithmetic on ints produces a result out of the int range, it is converted to a value of type “long”
  - The set for “long” is HUGE. It determined by the size of the computer memory.

DEMO THIS NOW: bigi=??*??; print bigi
print type(bigi)
Whence -2,147,483,648?

-2,147,483,648 = -(2^{31}) = -100 ... 00 [31 0's]

2,147,483,647 = +(2^{31} - 1) = 011 ... 11 [31 1's]

- A popular natural word size is 32 (bits).
- The first #, approx 2.15 billion, is called 2 Gigabinary, abbreviated as in “a 4 Gibi-Byte or GbiB memory.” 1 Gib = 2^{30} = 1,073,741,824
- Strictly, Giga (g not b) means 1,000,000,000 as in 2.4 Gigahertz. Hard-drive sizes are advertised in Giga units so they look bigger to unwary shoppers!
- How many different values are in the int range
  \[
  \{ -2,147,483,648 = -(2^{31}), \ldots, -2, -1, 0, 1, 2, \ldots, 2,147,483,647 = +(2^{31} - 1) \}?
  \]
Lengths of intervals

• Suppose a and b are integers with $a \leq b$.
• How many different integers are in the set \{ a, a+1, a+2, \ldots, b – 2, b – 1, b \}? In other words, how many are both $\geq a$ and $\leq b$?
• Vote:

  (1) $a + b$
  (2) $a - b$
  (3) $| a - b |$
  (4) $b - a$
  (5) $b - a + 1$
  (6) $b - a - 1$
How many ints?

Apply the correct formula to

\[
\{ -2,147,483,648 = -(2^{31}), \ldots -2, -1, 0, 1, 2, \ldots \\
2,147,483,647 = +(2^{31} - 1) \}
\]

- Are 1 or 2 quantities below greater than the others? Vote: Which is largest?

(1) the number of ints in the above range
(2) \(2^{32}\)
(3) the number of 0-1 strings of length 32.
Talking carefully about * in Python

- When * operates on two "int" values, it multiplies them (as in math.) and the return value is an "int" or a "long", depending on how big it is.

- When * operates on a "string" and an "int" (in any order), it duplicates the string. (Negative ints behave like 0 in this operation).
  - Similarly for "int" and "list"

- So ---- Python's * is a **polymorphic** operator.

- When * operates on numbers with at least one "float", it multiplies and returns a "float". Roundoff may occur—the return value might not exactly equal the ordinary mathematical product.
Python and Java have
Built-in types
and
Programmer-defined types

- Built-in types have built-in names (int, bool, ...)
- Programmer-defined types are named “class” or “instance” (Java: “interface”) plus a programmer defined name, like Picture, Pixel, Color
- For now, the programmers are the Myro and the Python developers and contributors. Java and Py Python have huge libraries; in Java they ALL define classes. (Many do so in Python.)
What is the type of a Python value? Use `type()` to find out.

```python
>>> print type(192)
<type 'int'>
>>> aPture = makePicture(pickAFile())
>>> print type(aPture)
<class 'myro.graphics.Picture'>
>>> aPixl = getPixel(aPture, 1, 1)
>>> print type(aPixl)
<class 'myro.graphics.Pixel'>
>>> aCol = makeColor(1, 2, 3)
>>> print type(aCol)
<class 'myro.graphics.Color'>
```
What operations are available on a value with a given type?

>>> help(Picture)  [ much omitted ]
  | getColor(self, x, y)
  | getPixel(self, x, y)
  | setColor(self, x, y, newColor)

>>> help(Pixel)  [ ...... ]
  | getColor(self)
  | setColor(self, newColor)

>>> help(Color)  [ ...... ]
  | getRGBA(self)
  | makeDarker(self)
Viewing values of type “class”

```python
>>> print aPixl
<Pixel instance (r=9, g=5, b=30, a=255) at (1, 1)>

>>> print aPture
<Picture instance (256 x 192)>

>>> print(aPture.__dict__)
{'palette': None, 'image': <PIL.Image.Image instance at 0x8ab058c>, 'filename': '/home/seth/Courses/tcsi201/P.jpg', 'width': 256, 'mode': None, 'height': 192, 'pixels': <PixelAccess object at 0xb7d67c70>}
```
Viewing more values of type 'class'

>>> print aCol
<Color instance (r=1, g=2, b=3, a=255)>

>>> print aCol.__dict__
{'alpha': 255, 'rgb': [1, 2, 3]}
Python dictionaries

```python
>>> myD = dict()
>>> myD['apple'] = 'fruit, record, PC'
>>> myD['pear'] = 'fruit'
>>> myD['Chaiken'] = 'undefinable'
>>> print myD['pear']
>>> print myD
```

- The key ('apple', e.g.) can be most any value.
- The value ('fruit', e.g.) can be any value.
- Lookup is fast because Python uses hashing.
- Python uses its dictionaries to implement Python classes
Python Word Play, etc.

- Dictionaries help one to start programming “linguistic computations”

  - e.g. mD['cat'] = ['noun','regular']
  - mD['sleep'] = ['verb', 'intransitive']
  - mD['a'] = ['adjective','article']
  - mD['walk'] = ['verb','transitive']
  - mD['I'] = ['noun']

  I walk a cat.
  I sleep.

- Playing with this is encouraged..See ThinkCSPy.
Let's learn it now with Turtles