Review

int(), float(), str()

Review

- **Numeric types / constructors**
  - integers / int()
    - Whole numbers
  - floating point numbers / float( )
    - Numbers with a decimal point
- Conversion
  - float to integer
  - integer to float
- type()
  - Returns type of the value

Review

- **string** data type (object) is used to represent and manipulate text data
  - Sequence of characters enclosed in quotes
    - Alphanumeric characters (A – Z, a – z, 0 – 9)
    - Blanks
    - Punctuation
    - Various symbols
  - **string** object can be assigned to a variable
- Convert numeric to string
  - Use str() constructor

Review

- str() – the string constructor
  - Applied to a number, returns string representation of the number
  - str(3.1) return '3.1'

Lists

Another Python object

List

- A sequence of objects
  - Numbers
  - Strings
  - Combination of numbers and string
  - Other lists
- Represented as a comma-separated sequence of objects enclosed within square-brackets [ ]

List of strings

```
>>> ['Dolores', 'Mickey', 'Minnie']
['Dolores', 'Mickey', 'Minnie']
>>>
```

```
>>> # assign list to a variable
>>> names = ['Dolores', 'Mickey', 'Minnie']
>>> names
['Dolores', 'Mickey', 'Minnie']
>>> print(names)
['Dolores', 'Mickey', 'Minnie']
>>>
```

List of strings and numbers

```
>>> ['Dolores', '243 S. Wabash', 4.0]
['Dolores', '243 S. Wabash', 4.0]
>>> studentData = ['Dolores', '243 S. Wabash', 4.0]
>>> print(studentData)
['Dolores', '243 S. Wabash', 4.0]
>>>
```

What is the difference between string types and list types?

Difference between strings and lists

```
>>> alphaString = 'abcdefg'
>>> alphaList = ['a', 'b', 'c', 'd', 'e','f','g']
>>> print(alphaString, alphaList)
abcdefg ['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

```
>>> type(alphaString)
<class 'str'>
>>> type(alphaList)
<class 'list'>
>>>
```

Use type ( ) – Python
alphaString - string object
alphaList - list object

Operators

Strings and Lists

+ operator

Concatenation of strings

```
>>> 'he' + 'llo'
'hello'
>>> s = 'Hello'
>>> t = 'World'
>>> s + t
'HelloWorld'
```

How do you get the space between Hello and World?

```
>>> '3' + '5'
```

What is the result?

Concatenation of lists

```
>>> s = ['he']
>>> t = ['llo']
>>> s + t
['he', 'llo']
>>> s = [1, 2, 3]
>>> t = [4, 5, 6]
>>> s + t
[1, 2, 3, 4, 5, 6]
```

Indexing Operator []

Individual characters of a **string** and items in a **list** can be accessed using the indexing operator **[ ]**

	-5	-4	-3	-2	-1
s =	' A	p	p	l	e '
	0	1	2	3	4

```
>>> s[0]
'A'
>>> s[4]
'e'
>>> s[-1]
'e'
```

What is returned from s[1:4]? s[1:]? s[:-1]? s[::-1]?

	-3	-2	-1
s =['Apple','Orange', 'Peach']			
	0	1	2

```
>>> s[0]
'Apple'
>>> s[2]
'Peach'
>>> s[-1]
>>> 'Peach'
```

What is returned from s[1][4]?

String and List Operators

Usage	Explanation
`x in s`	x is a substring of s
`x not in s`	x is not a substring of s
`s + t`	Concatenation of s and t
`s * n, n * s`	Concatenation of n copies of s
`s[i]`	Character at index i of s

>>> help(str)

>>> help(list)

Usage	Explanation
`x in lst`	x is an item of lst
`x not in lst`	x is not an item of lst
`lst + lstB`	Concatenation of lst and lstB
`lst*n, n*lst`	Concatenation of n copies of lst
`lst[i]`	Item at index i of lst

String and List functions

Usage	Explanation
`len(s)`	(function) Length of string `s`

Usage	Explanation
`len(lst)`	Number of items in `lst`
`min(lst)`	Minimum item in `lst`
`max(lst)`	Maximum item in `lst`
`sum(lst)`	Sum of items in `lst`

Count Method

Method	
lst.count(item)	Returns the number of occurrences of item in list lst
s.count(target)	The number of occurrences of substring target in string s

Decision Structures

Change the flow of control

Decisions

if **condition**:

Condition - **comparison** expression that evaluates to a Boolean value (**True** or **False**)

Operand		Operations		
A	B	A and B	A or B	Not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

- Comparisons using **relational** operators
  - ==, !=
  - <, <=
  - >, >=
- Comparisons using **boolean** operators
  - and
  - or
  - not
- Boolean variables
- Python: Comparisons using **in** operator
  - String
    - x in s (x substring of s)
    - x not in s(x substring of s)
  - List
    - x in lst (x object in list)
    - x not in lst (x object in list)

What objects may be in a list?

One-Way Decisions

```
if <condition>:
    <indented code block>
<non-indented statement>
```

Indented code block is one or more python statements; may be another if statement

Two-Way Decisions

```
if <condition>:
    <indented code block 1>
else:
    <indented code block 2>
<non-indented statement>
```

```
if <condition 1>:
    <indented code block 1>
elif <condition 2>:
    <indented code block 2>
<non-indented statement>
```

Series of IFs

```
if <condition_1>:
    <indented code block 1>
If <condition_2>:
    <indented code block 2>
If <condition _3>:
    <indented code block 3>

<non-indented statement>
```

Nested IFs

```
if <condition_1>:
    if <condition_A>:
        <indented code block A>
    else:
        <indented code block not A>
    <indented code block 1>
<non-indented statement>
```

Variables and assignments

Mutable and immutable types

Mutable vs Immutable

List

- Contents of a list can be changed - mutable

```
>>> s = ['Apple', 'Orange', 'Peach']
>>> print (s)
['Apple', 'Orange', 'Peach']
>>> t = s
>>> print (t)
['Apple', 'Orange', 'Peach']
>>> s[1] = 'Pear'
>>> print(s)
['Apple', 'Pear', 'Peach']
>>> print (t)
['Apple', 'Pear', 'Peach']
```

String

- Contents of a string cannot be changed - immutable

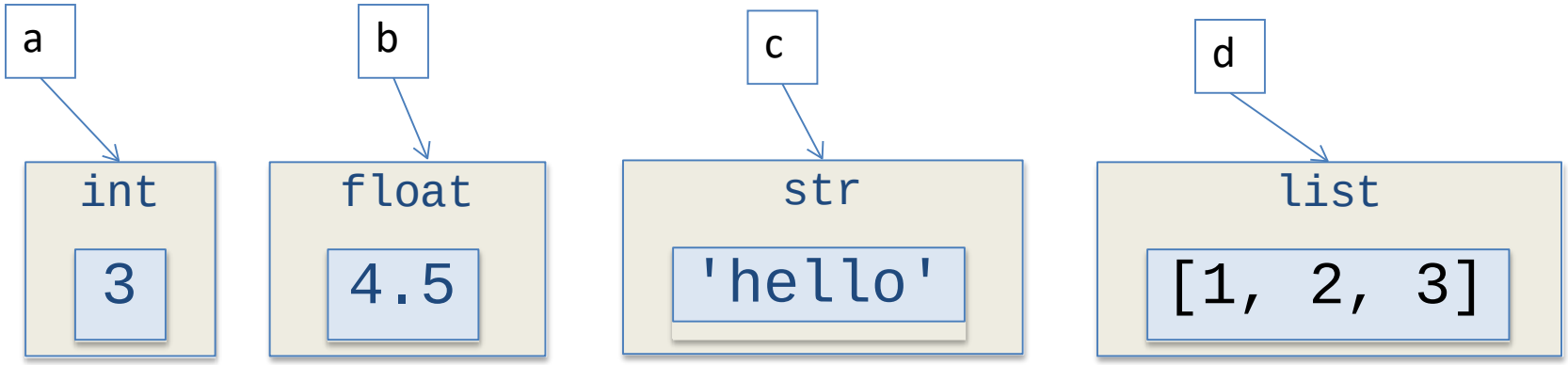
```
>>> s = 'Apple , Orange,  Peach'
>>> print (s)
Apple, Orange, Peach
>>> s[1] = 'Pear'
TypeError: 'str' object does not support
    item assignment
>> > s[1] = 'P'
TypeError: 'str' object does not support
    item assignment
```

What do we need to do to replace the first p in Apple with an uppercase P?

Assignment

```
>>> a = 3
>>> type(a)
<class 'int'>
>>> b = 4.5
>>> c = 'hello'
>>> d = [1, 2, 3]
```

It is not the identifier (variable) that has a type, but the object assigned to it.



Object a is of type int = object a belongs to class int

Example – mutable object

```
>>> b = [1, 2, 3]

>>> a = b

>>> b[2]

>>> b[2] = 4

>>> b

>>> a

>>> a = 10

>>> a

>>> b
```

Example – immutable object

```
>>> x = 23

>>> y = x

>>> y = 100

>>> x

>>> y
```

Swapping

Need to swap the values of x and y,
i.e. assign 4 to x and 3 to y

```
>>> x = 3

>>> y = 4

>>> z = x

>>> x = y

>>> y  = z
```

Practice

Examples 1 - 4

Example 1

- **Problem:** If age is greater than 62, print 'You are eligible for social security benefits
- Describe the algorithm
  1. Get a number from the user
  2. Determine if number is > 62
  3. If number < 62
     Print message
- **Key Knowledge**:
  - Do a numeric comparison
  - input() function returns a string; need to convert to integer

Note: since we will be doing several examples, we will write a user-defined function for each example and save all the functions in one Python program file. We will execute each function in IDLE by specifying the appropriate function name.

Example 2

- **Problem:** If the value of variable name is in the list [‘Mozart’, ‘Puccini’, ‘Rossini’, ‘Wagner’, ‘Verdi’, ‘Strauss’], print ‘One of the 6 most popular opera composers’
- Describe the algorithm
  1. Define a list of composer names
  2. Get a composer name (string) from the user
  3. Determine if composer name is in the list
     Print the message
- **Key Knowledge:**
  – Input from user is stored in the variable **name**

Example 3

- **Problem**: If a substring of characters appears more than 2 times in the string values associated with the variable report, then print ‘Yes’
- Define the algorithm
  1. Create and store a sequence of meaningful characters
  2. Get sequence of characters from the user i.e. substring
  3. Determine number of times substring occurs in the sequence
  4. If number of occurrences is > 2
     Print the message
- **Key knowledge**:
  – Store sequence of meaningful characters as a **string** in the variable report
  – String type has a **count** method

Example 4

- **Problem**: if at least one of the Boolean variables **north**, south, east, west is True, print 'I can escape'
- Describe the algorithm
  1. Initialize Boolean variables to True or False
  2. Determine Boolean operator that returns True if at least one value in the Boolean variables is True
  3. If True
     Print the message
- Key knowledge:
  - Boolean operator that returns True if at least one of the Boolean variables is True in 'or'. Review truth tables.
  - Test case 1: all variables are False
  - Test case 2: all variables are True
  - Test case 3: two variable are False
  - Test case 4: three variables are False

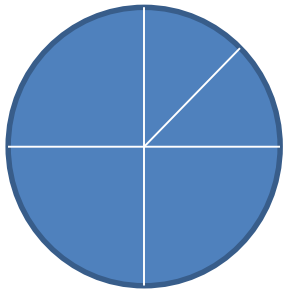
Practice

Examples 5-8

Example 5

- **Problem**: Guess the tip on a restaurant bill: the tip 10% - 20%
- Describe the algorithm
  1. Get a check amount from the user
  2. Get a guess of the tip amount from the user
  3. Compute a 10% tip
  4. Compute a 20% tip
  5. If the guess is between 10% tip calculation and 20% tip calculation
     Print ‘Okay, you guessed right’, otherwise
     Print ‘ Your guess is not good’
- Key knowledge:
  - Use relational operators to test for guess <= 20%tip, guess >=10%tip
  - Use Boolean operator ‘and’ to test for guess between 10%tip and 20%tip

Example 6



- **Problem**: Does the point(x,y) lie in the target area (disk around origin with radius r)
- Describe the algorithm
  1. Get the radius of the disk from the user
  2. Get the x-coordinate of the point
  3. Get the y-coordinate of the point
  4. Compute if the point is less then the radius of the disk
  5. If hit is successful

     Print 'You hit the target. Well done', otherwise

     Print 'You missed the target, try again later'.
- Key knowledge:
  - Review the geometry – for the area (x **2 + y ** 2) < radius ** 2

Example 7

- **Problem**: Given a number between 1 and 12, determine the month
- Describe the algorithm
  1. Get number between 1 and 12 from user
  2. Check for invalid number
  3. If valid number
     Print corresponding name of month
  4. If not a valid number
     Print Invalid month number
- Key knowledge:
  - Use a Boolean variable to determine if a number is invalid.

Example 8

- **Problem**: Given a temperature determine if it is a good day to swim, golf or play tennis
- Describe the algorithm
  1. Get temperature from user
  2. If temperature >= 80 - swim
  3. If temperature >=50 but less than 80 - golf
  4. If temperature < 50 – tennis
- Key knowledge:
  - Use a nested if