Recall: Python primitive elements are objects

```python
>>> 99  # for example, a number

>>> type(99)  # objects have a type that
<class 'int'>  # determines range of values
               # and operations

>>> id(example)  # objects are stored in memory
4297627072  # variables associate a label

>>> example = 99  # with an object
```
Recall, 2.

```python
>>> example + 1  # expressions combine objects
100  # expressions are evaluated and
     # return a value
```

The value returned by an expression is automatically printed when Python code is executed in the Shell.

And is *not* automatically printed otherwise!
(For example, if executing code from .py file.)

Recall, 3.

```python
>>> example = 99
>>> example + 1
??
>>> example2 = example + 1
>>> example
??
>>> example2
??
>>> example = example + 1
>>> example
```
>>> example = 99
>>> example + 1
100
>>> example2 = example + 1
>>> example
99
>>> example2
100
>>> example = example + 1
>>> example
100

>>> x = 0
>>> x
??
>>> x = x + 1
>>> x
??
>>> x = x + 2
>>> x
??
CIS 122 Introduction to Programming and Problem Solving  Spring 2017

>>> x = 0

>>> x
0

>>> id(x)
4297623904

>>> x = x + 1

>>> x
1

>>> id(x)
4297623936

>>> x += 2  # x = x + 2

>>> x
3

>>> id(x)
4297624000

CIS 122 Introduction to Programming and Problem Solving  Spring 2017

Why variables?

Readability/Clarity

Reuse
Solving a problem using Python:

- You have $50 and are buying some new movies that cost $15 each. You will also purchase some old movies, which are 1/3 the cost of a new movie. Write Python code to show how much money you have left after buying 1 new movie and 2 old movies.

\[
\text{answer} = 50 - (15 \times 2) - (5 \times 2)
\]

- You have $50 and are buying some new movies that cost $15 each. You will also purchase some old movies, which are 1/3 the cost of a new movie. Write Python code to show how much money you have left after buying 2 new movies and 2 old movies.

\[
\text{answer} = 50 - (15 \times 2) - (5 \times 2)
\]
CIS 122 Introduction to Programming and Problem Solving  Spring 2017

• You have $50 and are buying some new movies that cost $15 each. You will also purchase some old movies, which are 1/3 the cost of a new movie. Write an expression that shows how much money you have left after buying 2 new movies and 2 old movies.

answer = 50 – (15 * 2) – (5 * 2)

... changed mind - 2 new movies and 3 old movies
... new movies actually cost $10
... old movies are actually ½ the cost of new movies
... [later] what problem is being solved ??

CIS 122 Introduction to Programming and Problem Solving  Spring 2017

• You have $50 and are buying some new movies that cost $15 each. You will also purchase some old movies, which are 1/3 the cost of a new movie. Write an expression that shows how much money you have left after buying 2 new movies and 2 old movies.

start_cash = 50
cost_new_movie = 15
cost_old_movie = cost_new_movie / 3
buy_new = 2
buy_old = 2
spend_new = cost_new_movie * buy_new
spend_old = cost_old_movie * buy_old
end_cash = start_cash – spend_new – spend_old
You have $50 and are buying some new movies that cost $10 each. You will also purchase some old movies, which are 1/2 the cost of a new movie. Write an expression that shows how much money you have left after buying 2 new movies and 3 old movies.

\[
\begin{align*}
\text{start\_cash} &= 50 \\
\text{cost\_new\_movie} &= 10 \\
\text{cost\_old\_movie} &= \text{cost\_new\_movie} / 2 \\
\text{buy\_new} &= 2 \\
\text{buy\_old} &= 3 \\
\text{spend\_new} &= \text{cost\_new\_movie} \times \text{buy\_new} \\
\text{spend\_old} &= \text{cost\_old\_movie} \times \text{buy\_old} \\
\text{end\_cash} &= \text{start\_cash} - \text{spend\_new} - \text{spend\_old}
\end{align*}
\]

Python Primitive Elements

Variables

```python
>>> example = 88
>>> example

a) 88  b) 99
c) syntax error  d) none of these
```
Python Primitive Elements
Variables

```python
>>> example = 99
>>> nextvar = example / 9
>>> nextvar
```

a) 11  b) 11.0  c) syntax error  d) none of these

```python
>>> nextvar = example // 9
>>> nextvar
```

a) 11  b) 11.0  c) syntax error  d) none of these
Python Primitive Elements
Variables

```python
>>> nextvar = 99.0 // 9
>>> nextvar

a) 11  b) 11.0  c) syntax error  d) none of these
```

```python
>>> nextvar = 99.0 // 9
>>> nextvrr

a) 11  b) 11.0  c) syntax error  d) none of these
```

```python
>>> a = 42
>>> a = 10
>>> a = 10
>>> b = a
>>> b = 20
>>> b = 20
>>> a = a * 2
>>> c = a * b
>>> a = b
>>> a
>>> c = c + b
>>> b = 15
>>> b
>>> a += b
>>> a
>>> b
>>> a
>>> b
>>> c
>>> b
```
When something goes wrong:

```python
>>> temperature = 20
>>> temptrature
>>> temperature +* 20
>>> temperature ++ 20
>>> temperature^2
```

```
# name error (runtime)
# syntax
# ok (logic?)
# logic
```

"Good [programming] comes from experience, and a lot of that comes from bad [programming]."
– Will Rogers (sort of)
Python toolkit so far

numbers (ints and floats)
numeric operations e.g., + - ...

expressions
variables and assignment

Syntax error
NameError

numeric operations:
+ - * // % **
pow
abs, round

int, float
help, print
type, id

built-in functions
A function names an operation:

```python
>>> abs
<built-in function abs>
```

Calling a function:

```python
>>> abs()
```

```python
>>> abs(7)
7
```

Type Error: abs() takes exactly one argument (0 given)
Python Functions

The general form of a function call:

<function name>(<arguments>)

>>> abs(-7)
7

>>> y = 2.001
>>> round(y)
2
>>> pow(3, 2)
9
Python Functions

```python
>>> y = 2.001
>>> round(y)
2
>>> pow(3, y)  # be careful
??
```
For example,

```python
>>> y = 3.14159
>>> z = abs(round(y))
??
```

Python built in functions

Functions are an executable data type

```python
>>> <fn name>(<arguments, if any>)

  function calls are expressions
  they return a value
```
Python Functions

>>> round(3.5)
4
>>> round(4.5)
4

Another data type - Strings

>>> 'Hello, World'
'Hello, World'

>>> len('Hello, World')
12
Another data type - Strings

```python
>>> 'Hello, World'
'Hello, World'

>>> "Hello, World"
"Hello, World"

>>> "Hello, Duck's World"
"Hello, Duck's World"
```

CIS 122 Spring 2017

Another data type - Strings

```python
'''
CIS 122 Spring 2017
Week 2 Examples
'''
```

CIS 122 Introduction to Programming Spring 2017
Another data type - Strings

```python
>>> h = 'Johnson\tDeady\tFriendly\nHall'
>>> print(h)
Johnson    Deady    Friendly
Hall

>>> len(h)
27
```
Another data type - Strings

```python
>>> formula = '2\u03C0r'
>>> print(formula)
2\(\pi\)r

>>> len(formula)
3
```

Another data type - Strings

```python
>>> '97403'
'97403'

>>> type('97403')
<class 'str'>
```
Another data type - Strings

```python
>>> 97403
97403

>>> Lincoln
NameError: name 'Lincoln' is not defined
```

• Objects
  – Value
  – Type (range of values and operations)
  – Id (unique identifier; memory location)

```python
>>> x = 210
>>> y = '210'
>>> x
>>> y
>>> len(x)
>>> len(y)
>>> x / 3
>>> y / 3
>>> x + 3
>>> y + '3'
>>> x + '3'
```
CIS 122 Introduction to Programming Spring 2017

>>> x + '3'
Traceback (most recent call last):  File "<pyshell#25>" , line 1, in <module>
x + '3'
TypeError: unsupported operand type(s) for +: 'int' and 'str'

>>> y / 3
Traceback (most recent call last):  File "<pyshell#27>" , line 1, in <module>
y / 3
TypeError: unsupported operand type(s) for /: 'str' and 'int'

>>> len(x)
Traceback (most recent call last):  File "<pyshell#28>" , line 1, in <module>
len(x)
TypeError: object of type 'int' has no len()

CIS 122 Introduction to Programming Spring 2017

Python Strings

✓ string literals – len, +, *

• strings are sequences – sequential operators
  – index, slice, for
String operators
  + and * are overloaded operators

>>> 'try' + 'this'
'trythis'

>>> 'try this' * 4 + 'ok'
'try thistry thistry thistry thisok'

Strings are sequences of characters

“hello, world”

0 1 2 3 4 5 6 7 8 9 10 11

>>> greeting = “hello, world”
>>> len(greeting)
12
Strings are sequences of characters

“hello, world”

0 1 2 3 4 5 6 7 8 9 10 11

>>> greeting[0]
>>> greeting[11]
>>> greeting[-1]

>>> greeting[len(greeting)]  # be careful
IndexError: string index out of range

>>> greeting [1:5]        'ello'
>>> greeting[:5]         'hello'
>>> greeting[2:]         'llo, world'
>>> greeting[2:9]        'llo, wo'
>>> greeting[0:len(greeting)] 'hello, world'
Strings are sequences of characters

```python
>>> a = "ARMADILLO"
>>> c = "CHAMELEON"
>>> string0 = a[2:5]*2 + c[3:6] + c[7:len(c)]
>>> string0
??
```
Python sequential operator `for`

```python
>>> sentence = 'Hello, CIS122!
>>> for ch in sentence:
    print(ch)

Hello,
C...
```

Python `Print` and `Input`

Another function that works with string args:

```python
>>> print('hello, world')
hello, world
```
Python `print` and Input

Another function that works on strings:

```python
>>> print('hello, world')
hello, world

>>> print('hello,
  world')
hello,
  world

>>> greeting = 'hello,\world'
>>> len('greeting')
8
>>> greeting
'hello,\world'
>>> len(greeting)
12
>>> print(greeting)
hello,  world
Python Print and `input`

```python
>>> name = input()
CIS 122

>>> name
'CIS122'

>>> print(name)
??
```
Python Print and Input

```python
>>> days_in_month = input('How many days? ')
How many days? 30

>>> days_in_month
'30'

>>> type(days_in_month)
<class 'str'>
```

```python
>>> day_before = days_in_month - 1
TypeError: unsupported operand type(s) for -: 'str' and 'int'

>>> total = .01 * (2 ** day_before)
```
Python Print and **Input**

```python
>>> days_in_month = input('How many days? ')  
How many days? 30

>>> days_in_month = int(days_in_month)

>>> days_in_month  
30

>>> type(days_in_month)  
<class 'int'>
```

Python Print and **Input**

What will be the result when this code is executed? (be careful)

```python
>>> days_worked = input('How many days? ')  
How many days? 20

>>> pay_check = days_worked * 120
```
Python Print and **Input**

```python
>>> days_worked = int(input('How many days? '))
How many days? 20

>>> days_worked
20

>>> days_worked * 120
2400
```

```python
>>> type(days_worked)
??

>>> days_worked = str(days_worked)
>>> days_worked = float(days_worked)
```