Lab2 Preview

Built-in Functions

```
functionName(     )
```

name of the function

zero or more arguments

Def: A **function** is a reusable block of code

Built-in functions
`exit()` and `type()`

Different ways to run Python Code

Python Interpreter

- Enter lines one by one at the prompt
- Expression outputs are **printed** to the screen.

```
>>> x = 3+5
>>> x
8
```

Python Program

- Python evaluates each line in the file
- Expression outputs are **not printed** to the screen.
- Instead, anything that is printed to the screen requires a **print function**

example.py file

```
x = 3+5
print(x)
```

→

Terminal

```
$ python3 example.py
8
```

Print Statements

Python 2 vs. Python 3: `print` is a statement vs. a function!

Def: The **print** function explicitly write the output to the terminal.

variables

strings

ints and floats

Combos of expressions (separated by commas)

empty lines

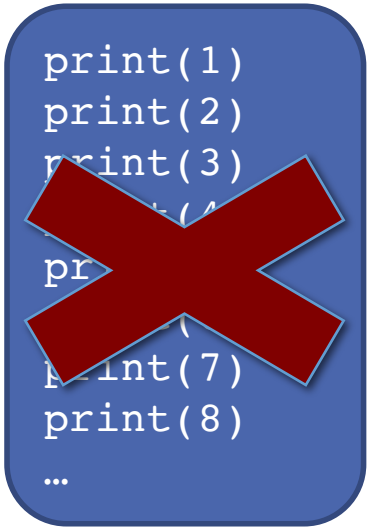
```
x = 3+5
print(x)
print('Hello World!')
print('hi' + ' ' + 'there')
print(1.0)
print(1+10)
print('Number Of Seats:',24)
print('Number Of Computers:',25)
print()
```

FOR Loops

Def: **FOR** loops are statements that specify repeated execution (also called *iteration*)

example.py

```
print(1)
print(2)
print(3)
```



"For each element in a list, do something."

example.py

```
for item in [1,2,3]:
    print(item)
```

List

Do this.

variable

example.py (unannotated)

```
for item in [1,2,3]:
    print(item)
```

Indentation matters!

FOR Loops

example.py

```
myList = [1,2,3]
for item in myList:
    print(item)
```

variable

Do this.

variable

example.py (unannotated)

```
myList = [1,2,3]
for item in myList:
    print(item)
```

Indentation matters!

example.py

```
for item in [1,2,3]:
    print(item)
```

List

Do this.

variable

example.py (unannotated)

```
for item in [1,2,3]:
    print(item)
```

Indentation matters!

FOR Loops

example.py

```
myList = [1,2,3]
for item in myList:
    print(item)
    print(item,'again')
print('Done!')
```

example.py (unannotated)

```
myList = [1,2,3]
for item in myList:
↔   print(item)
```

Indentation matters!

example.py

```
for item in [1,2,3]:
    print(item)
```

List

Do this.

variable

example.py (unannotated)

```
for item in [1,2,3]:
↔   print(item)
```

Indentation matters!