How are you writing your name?

- Easy way and hard way

```python
def drawname(x, y):
    myturt.clear()
    ...
    window = turtle.Screen()
    myturt = turtle.Turtle()
    myturt.onclick(drawname)
    ...
```

scripting - Python $\rightarrow$ C# $\rightarrow$ C (hard way)

parameter values come from mouse click

dot op

call 2nd

no arguments

OSU Oregon State University
Void Routines/Functions

print_menu()

• Define the function before it is called/invoked

def print_menu():
    print("Hello, welcome to Wells Fargo!")
    print("What would you like to do today?")
    print("Please enter 1 for savings, 2 for checking")
    print_menu()
Value Returning Functions

\[ y = f(x) = 3 \times x \]

- Define the function before it is called/invoked
- Function contains a return statement

```python
def f(x):
    return 3 * x
```

\[ y = f(3) \]

```
print(y)
```

- Could the argument be a variable?
Decomposition/Modularity

- **Control**
  - **Module1**
  - **Module2**
    - **Module3**
  - Main
Scope of Local and Global Variables

- Variables: A, B, C (Global to all modules)
- Control
  Variables: X, Y, Z (Local to Control)
- Module 1
  Variables: D, E, F (Local to Module 1)
- Module 2
  Variables: G, H, I (Local to Module 2)
- Module 3
  Variables: X, J, K (Local to Module 3)

Note: Variables A, B, C not in any function except main.
Terminology

• Function vs. Method vs. Procedure vs. Module
  — Refer to the same thing!

• Actual Parameters vs. Arguments
  — Refer to the same thing!

• Formal Parameters vs. Parameters
  — Refer to the same thing!

• Pass by value vs. pass by reference
  — VERY, VERY different!!!
Decomposition

• More general or specific?
• What happens if our $f(x)$ changed to $x^2$?
• Do we need to pass a value to $x$?

def f(x):
    x = 10

pass by value

def ma/n ():
    x = 3
    f(x)
Demo...

```python
# This function evaluates x^2 at a specific x, and returns the y
def f(x):
    y = x * x  # We can change the function easily here
    x = 3      # We cannot change the x value to something new outside f
    return y

# Let's have a control function from now on...
def main():
    x = 2  # If we make x a primitive, then we can't change it in f
    print(f(x))  # We can print or assign a value returning function
    print(x)

main()  # Function call to start the program...
```
Pass-by-Value and Pass-by-Reference

• Immutable vs. Mutable
  – Primitives – immutable
    a = 3
  – Lists – mutable
    a = [3]

\[
\begin{array}{c}
{4.2, \text{"jen", \text{"j"}}} \\
{10, \text{float, \text{true, \text{string, \text{char}}}}}
\end{array}
\]

\[
\begin{array}{c}
a = [3, 2] \\
a[0] = 3 \\
a[1] = 2
\end{array}
\]
This function evaluates $x^2$ at a specific $x$, and returns the $y$

```python
def f(x):
    y = x[0] * x[0]  # We can change the function easily here
    x[0]=3  # We can change a list item in a function
    return y
```

Let's have a control function from now on...

```python
def main():
    x = [2]  # If we make $x$ a list, then we can change it in $f$
    print(f(x))  # We can print or assign a value returning function
    print(x)

main()  # Function call to start the program...
```