Functions

What is a function?
It’s a method that isn’t part of a class.

Unlike in Java, in Python, it isn’t necessary for everything to reside in a class.

Like methods, functions gives us a mechanism for decomposing our code into small units, each of which accomplishes a logical task.
Let’s look at an example: consider the problem of determining the distance between two random points in the plane.

Generate first point
Generate second point
Call function to calculate the distance
Output result

Why use functions?

- Encapsulation (as mentioned on first slide)
- Code reuse
- Ease of maintenance
- Can reduce code size
- Isolate functionality for testing
Functions in Python

def euclidean_distance(pt1, pt2):
    dx = pt1[0] - pt2[0]
    dist = math.sqrt(dx**2 + dy**2)

    return dist

What’s missing (relative to Java)? Access modifier, return type, {}, an overwhelming sense of dread

Composition

The result of one function can be input to another

    angle = math.atan(math.tan(math.pi))

    dist = math.sqrt(sum_of_squares(dx, dy))

result of tan is input to atan

result of sum_of_squares is input to sqrt (sum_of_squares is not a built-in function)
Python Functions – Cool Feature 1

Multiple return values

A function to calculate the area and circumference of a circle

```python
def circle_stats(radius):
    area = math.pi * radius * radius
    c = 2 * math.pi * radius
    return area, c
```

```python
area, circumference = circle_stats(5)
```

Python Functions – Cool Feature 1

Multiple return values

A function to calculate polar coordinates given x, y coordinates in first quadrant

```python
def polar(a, b):
    theta = math.atan(float(b)/a)
    d = distance_formula((a, b), (0, 0))
    return theta, d
```

```python
angle, distance = polar(x, y)
```
Python Functions – Cool Feature 2

Optional parameters

A function to calculate the area and circumference of a circle

```python
def circle_stats(radius=1):
    area = math.pi * radius * radius
    c = 2 * math.pi * radius
    return area, c
```

```python
area, circumference = circle_stats(5)  # Passing a value
area, circumference = circle_stats()  # Using default
```

Python Functions – Cool Feature 2

Optional parameters

A function to calculate the area and circumference of a circle

```python
def foo(a, b=2, c=3, d=4):
    print(a)
    print(b)
    print(c)
    print(d)
```

```python
foo(5)  # Valid
foo(5, b=6)  # Valid
foo(5, c=7)  # Valid
foo(5, b=6, c=7, d=8)  # Valid
foo()  # Invalid
```
Python Functions – Cool Feature 2

Optional parameters

A function to calculate the area and circumference of a circle

```python
def foo(a, b=2, c=3, d=4):
    print(a)
    print(b)
    print(c)
    print(d)
```

What about this?

```python
foo(5, c=7, b=6)
```

Valid (but demented)

Exercise

Write a function that takes at least 1 and no more than 4 radii as parameters and returns the areas of circles with those radii.

```python
def areas(r1, r2=0, r3=0, r4=0):
a1 = math.pi * r1**2
a2 = math.pi * r2**2
a3 = math.pi * r3**2
a4 = math.pi * r4**2

return a1, a2, a3, a4
```
Exercise

Write a function that takes a Python list as a parameter. Each element in the list will be changed with probability p. The user may supply the value for p or use the default value of 0.5. When a value is changed, it is selected uniformly at random in the range 0..n. The user may supply n or use the default value of 9.

```python
def list_alter(vals, p=0.5, n=9):
    for i in range(len(vals)):
        if random.random < p:
            vals[i] = random.randint(0, n)
```

Question

```python
nums = [10, 20, 40]

Consider this code fragment:
    list_alter(nums, p=1.0)
    print(nums)
```

What is printed?
Parameter Passing

• For primitive types: a copy is passed to the function
  What are the implications of this?

• For complex types: a reference is passed to the function
  What are the implications of this?

Example 1

```python
def alter_x(x):
    x += 1

val = 5
alter_x(val)
print(val)
```

What is printed?
Example 2

```python
def alter_x(x):
    x += 1
    return x

val = 5
val = alter_x(val)
print(val)
```

What is printed?

Example 3

```python
def alter_list(nums):
    for i in range(len(nums)):
        nums[i] += 1

vals = [1, 2, 3]
alter_list(vals)
print(vals)
```

What is printed?
Example 4

```python
def alter_list(nums):
    for i in range(len(nums)):
        nums[i] += 1
    return nums

vals = [1, 2, 3]
vals = alter_list(vals)
print(vals)
```

What is printed?

Example 5

```python
def alter_list(nums):
    for i in range(len(nums)):
        nums[i] += 1
    return nums

vals = [1, 2, 3]
other = alter_list(vals)
print(other)
```

What is printed?
Example 6

def alter_list(nums):
    for i in range(len(nums)):
        nums[i] += 1
    return nums

vals = [1, 2, 3]
other = alter_list(vals)
vals[0] = 9
print(other)

What is printed?