PHOT 110: Introduction to programming

LECTURE 05

Michaël Barbier, Spring semester (2023-2024)

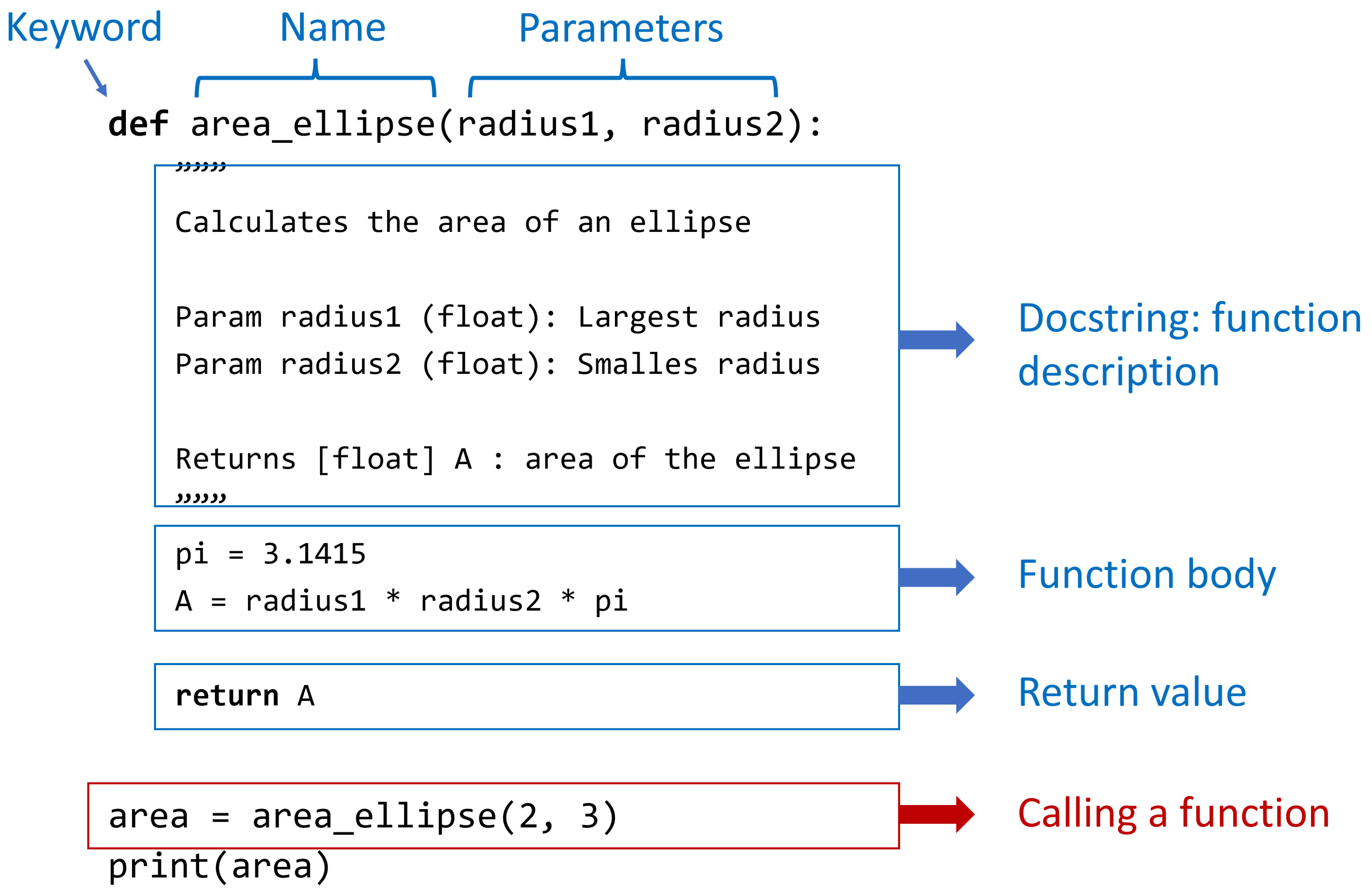
PROGRAM STRUCTURES SO FAR …

0. **Calculator**: Only expressions, re-use previous result, no re-running
1. Scripts: **variables & expressions**: re-run same script, track and remove human mistakes
2. Branching (**`if`**, **`else`**): explicit logical expressions, execute code according to decisions
3. **`while`** and **`for`** Loops: Repeat code, according to condition
4. **Functions**: further re-use, abstraction of code, call whenever required, allows parameters

FUNCTIONS

- are separate reusable pieces of code
- can be invoked or called from the script
- accept 0 or more parameters
- return an object(s) (possible `None`)

FUNCTION DEFINITION SYNTAX



```
def area_ellipse(radius1, radius2):
    ”””
    Calculates the area of an ellipse

    Param radius1 (float): Largest radius
    Param radius2 (float): Smalles radius

    Returns [float] A : area of the ellipse
    ”””
    pi = 3.1415
    A = radius1 * radius2 * pi
    return A

area = area_ellipse(2, 3)
print(area)
```

FUNCTION DECLARATION

```python
def area_ellipse(radius1, radius2):
    """
    area_ellipse calculates the area of an ellipse

    Param radius1 (float): Largest radius
    Param radius2 (float): Smalles radius

    Returns [float] A : area of the ellipse
    """
    pi = 3.1415
    A = radius1 * radius2 * pi

    return A

```

```
The area of the ellipse is 18.849
```

RETURN VALUES

- Use the **`return`** keyword
- the value can be an expression
- Without **`return`**, function returns **None**
- Multiple outputs possible

```python
def add(a, b):
    """ Sum a and b """
    return a + b

sum = add(2, 5)
print(sum)
```

```
7
```

RETURN VALUES

- Use the **return** keyword

```python
def add(a, b):
    """ Sum a and b """
    sum = a + b
    return sum

sum = add(2, 5)
print(sum)
```

```
7
```

RETURN VALUES

- Use the **`return`** keyword
- the value can be an expression

```python
def add(a, b):
    """ Sum a and b """
    return a + b

sum = add(2, 5)
print(sum)
```

```
7
```

RETURN VALUES

- Use the **`return`** keyword
- the value can be an expression
- Without **`return`**, function returns **None**

```python
def add(a, b):
    """ Sum a and b """
    a + b

sum = add(2, 5)
print(sum)
```

```
None
```

RETURN VALUES

- Use the **return** keyword
- the value can be an expression
- Without **return**, function returns **None**
- Multiple outputs possible

```python
def add(a, b):
    """ Sum a and b and give extra outputs """
    return a + b, "a second output", a - b

out = add(2, 5)
print(out)
```

```
(7, 'a second output', -3)
```

UNPACKING MULTIPLE OUTPUTS

```python
import math

def polar_to_carth(radius, angle):
  """ polar_to_carth converts polar to carthesian """
  x = radius*math.cos(angle)
  y = radius*math.sin(angle)

  return x, y

x, y = polar_to_carth(3, math.pi/3)
print(f"Polar (3, pi/3) = {(x, y)}")
```

```
Polar (3, pi/3) = (1.5000000000000004, 2.598076211353316)
```

MULTIPLE RETURN STATEMENTS POSSIBLE

```python
def minimum(a, b):
  """ Returns the minimum of two numbers """
  if a <= b:
    return a
  else:
    return b

print(minimum(3.4, 6.5))
```

```
3.4
```

VARIABLE SCOPE

Script variables and function parameters can have the same name

```python
def increment(a):
    """ Increment number by 1 """
    a = a + 1
    return a

# Define the variable in the script
a = 5
print("Variable a (before) = " + str(a))
b = increment(a)
print("Variable a (after) = " + str(a))
print("Variable b = " + str(b))
```

```
Variable a (before) = 5
Variable a (after) = 5
Variable b = 6
```

VARIABLE SCOPE

- Functions can use script variables from **outside** the function
- But **cannot** change those variables

```
def increment(a):
    """ Increment number by 1 """
    a = a + x
    return a

# Define the variable in the script
a = 5
x = 4
print(increment(a))
```

```
9
```

VARIABLE SCOPE

- Functions can use script variables from **outside** the function
- But **cannot** change those variables

```python
def increment(a):
    """ Increment number by 1 """
    a = a + x
    x = x + 3
    return a

# Define the variable in the script
a = 5
x = 4
print(increment(a))
```

```
UnboundLocalError: cannot access local variable 'x' where it
is not associated with a value
```

FUNCTIONS CAN CALL FUNCTIONS

```python
def minimum(a, b):
  if a <= b:
    return a
  else:
    return b

def add(a, b, fct):
  """ Sum a, b and minimum of both """
  return a + b + fct(a, b)

a = 10; b = 2
print(add(a, b, minimum))
```

```
14
```