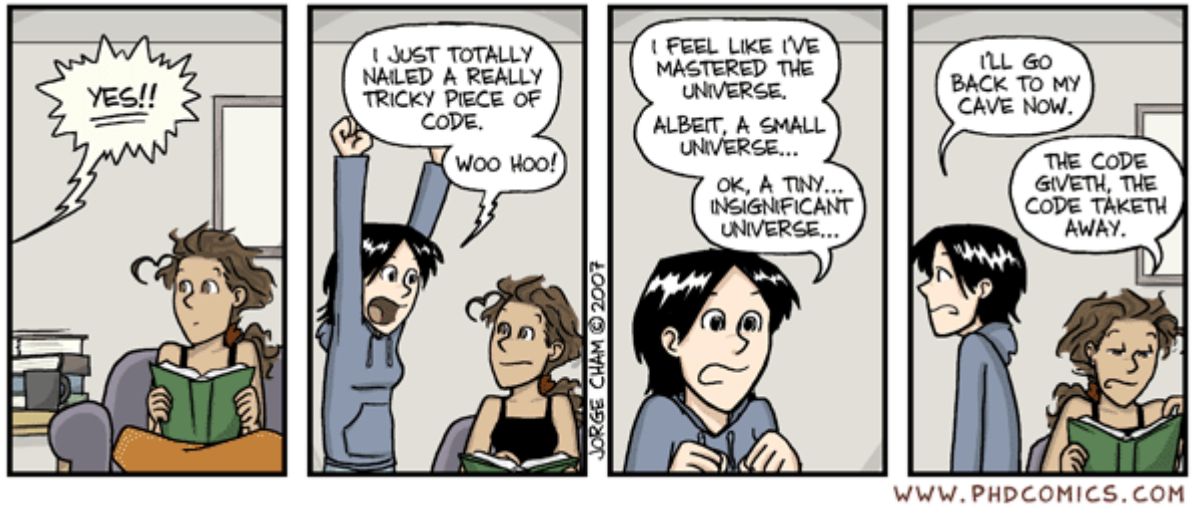
CS 115, Autumn 2021

Lecture 22: parameters



Thanks to Marty Stepp and Stuart Reges for parts of these slides

Organizing our code

- Our programs have become really long! This is a problem because:
  - They are hard to read
  - It is hard to find bugs in them
  - Often the same code is repeated
    - We can eliminate part of this with loops
      - Can we eliminate all of it?
- Python contains a lot of built-in commands. Can we make our code into new commands?

Creating commands

- Python commands like `print()` are just functions
- Recall: a ***function*** is a way to group statements together and give them a name.
- It would have been nice to put each animal in its own function in the Stamps project.
  - Why couldn't we do this?

Example: `bird`

- Code to draw a bird from lab 5:

```
panel.fill_oval(x + 5, y, BIRD_WIDTH - 10, BIRD_HEIGHT, "#990099") # body
panel.fill_oval(x + 40, y + 25, 20, 40, "#770077")                 # right wing
panel.fill_oval(x, y + 25, 20, 40, "#770077")                      # left wing
panel.fill_oval(x + 20, y + 10, 5, 5, "black")                     # left eye
panel.fill_oval(x + 35, y + 10, 5, 5, "black")                     # right eye
panel.fill_rect(x + 28, y + 18, 4, 8, "#eecc00")                   # beak
```

- Why can't we put this in a function using what we know already?

Scope

- A variable's scope is the part of the program where Python knows it exists and allows you to use it
  - Remember: variables created inside a function only exits in that function.
    - If we try to access a variable outside of our scope Python either creates a new variable with the same name or gives us an error

Example: `bird`

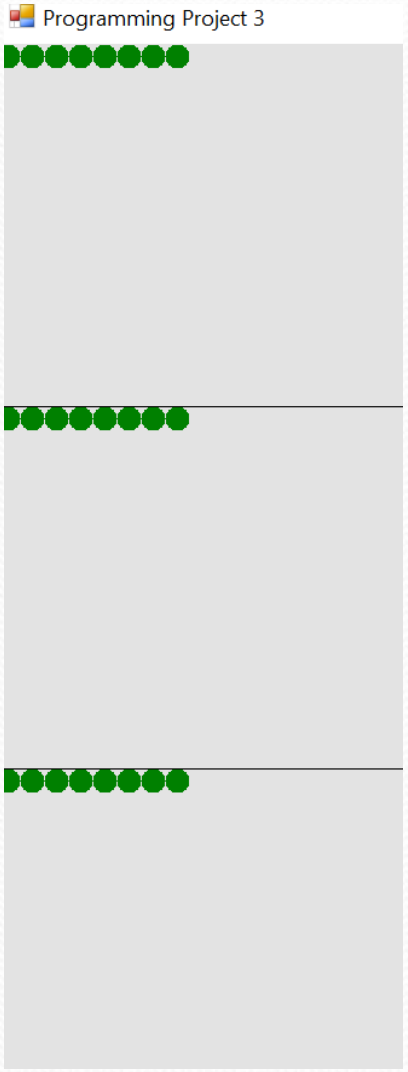
- Problem: `x` and `y` are declared in a different function

```
panel.fill_oval(x + 5, y, BIRD_WIDTH - 10, BIRD_HEIGHT, "#990099") # body
panel.fill_oval(x + 40, y + 25, 20, 40, "#770077")                 # right wing
panel.fill_oval(x, y + 25, 20, 40, "#770077")                      # left wing
panel.fill_oval(x + 20, y + 10, 5, 5, "black")                     # left eye
panel.fill_oval(x + 35, y + 10, 5, 5, "black")                     # right eye
panel.fill_rect(x + 28, y + 18, 4, 8, "#eecc00")                   # beak
```

- Why not make them `global` like `BIRD_WIDTH` and `BIRD_HEIGHT`?
  - `global` variables are dangerous – anyone can change their values
  - `global` variables can lead to name conflicts
  - In this class every non-constant variable **must** be in a function

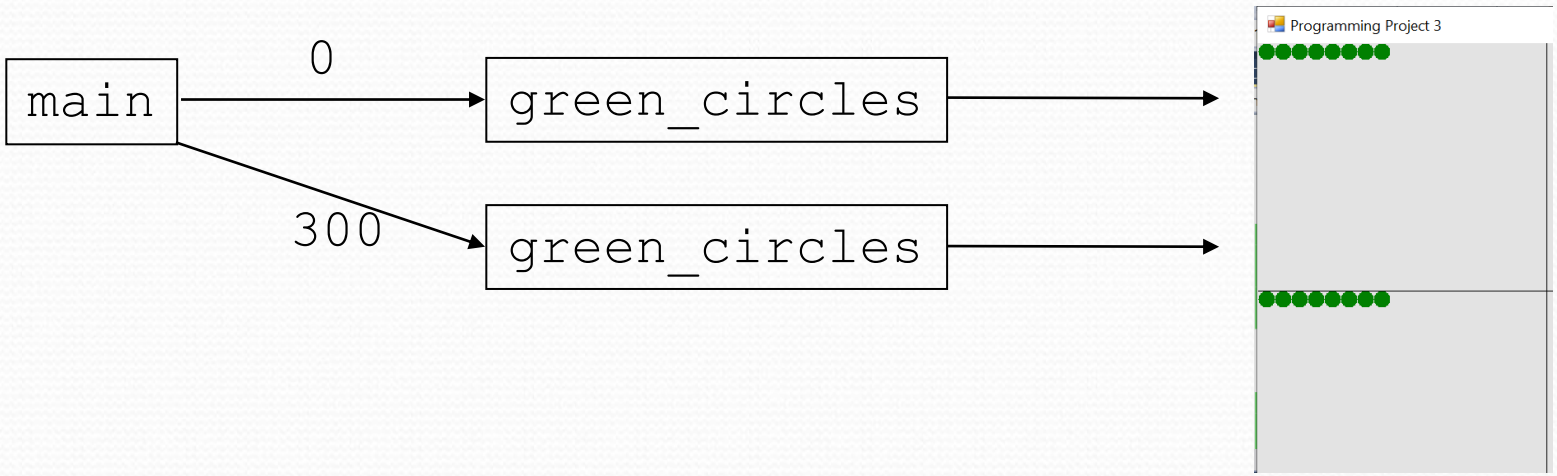
Solution: parameters

- Parameters allow us to send information to a function
  - We can make a function that will do many similar tasks
    - For example, draw a pattern of circles at different locations
  - Sometimes also called arguments



Parameterization

- **parameter**: A value passed to a function by its caller.
  - Instead of `green_circles_top_left`, `green_circles_middle_left`, write `green_circles` to draw at any location.
    - When *declaring* the function, we will state that it requires a parameter for the y coordinate.
    - When *calling* the function, we will specify what y to draw at.



Declaring a parameter

Stating that a function requires a parameter in order to run

```
def <name> (<name>):
    <statement>(s)
```

- Example:

```
def say_password(code):
    print("The password is: " + str(code))
```

  - When `say_password` is called, the caller must specify the integer code to print.

Passing a parameter

Calling a function and specifying values for its parameters

<name> (***<expression>***)

- Example:

```
def main()
    say_password(42)
    say_password(12345)
```

Output:

```
The password is 42
The password is 12345
```

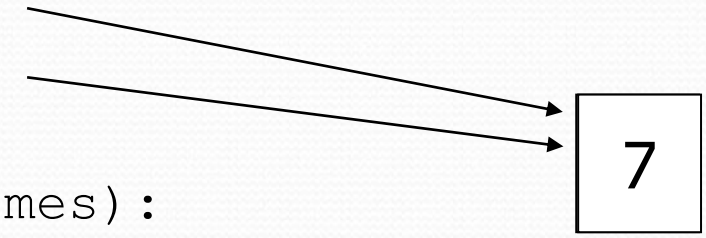
How parameters are passed

- When the function is called:
  - The value is stored into the parameter variable.
  - The function's code executes using that value.

```
def main()
    chant(3)
    chant(7)

def chant(times):
    print("Just", times, "salads...")

main()
```



Common errors

- If a function accepts a parameter, it is illegal to call it without passing any value for that parameter.

```
chant()          // ERROR: parameter value required
```

- Exercise: Change the lab 5 code to place the bird drawing in its own function.

Multiple parameters

- A subroutine can accept multiple parameters. (separate by ,)
  - When calling it, you must pass values for each parameter.

- Declaration:

```
def <name>(<name>, …, <name>):
    <statement>(s)
```

- Call:

```
<name>(<exp>, <exp>, …, <exp>)
```

Multiple parameters example

```
def main():
    print_numbers(4, 9)
    print_numbers(17, 6)
    print_numbers(8, 0)
    print_numbers(0, 8)

def print_numbers(number, count):
    for i in range(1, 6):
        print(str(number) + str(count) + " ")
    print ()
```

Output:

```
49 49 49 49 49
176 176 176 176 176
80 80 80 80 80
08 08 08 08 08
```