Parameters & Arguments

Tues, Oct 13th

Announcements

- Ruha Benjamin talk “Race to the Future: Reimagining the Default Settings of Technology & Society”
  - 4 pm today
- Creative Coder grades delayed
- LA application deadline Monday
- Video tutorial up, more code in class repo. Access issues?
- Reminder—no quiz this week!
- **Benchmark - Nov 13th**

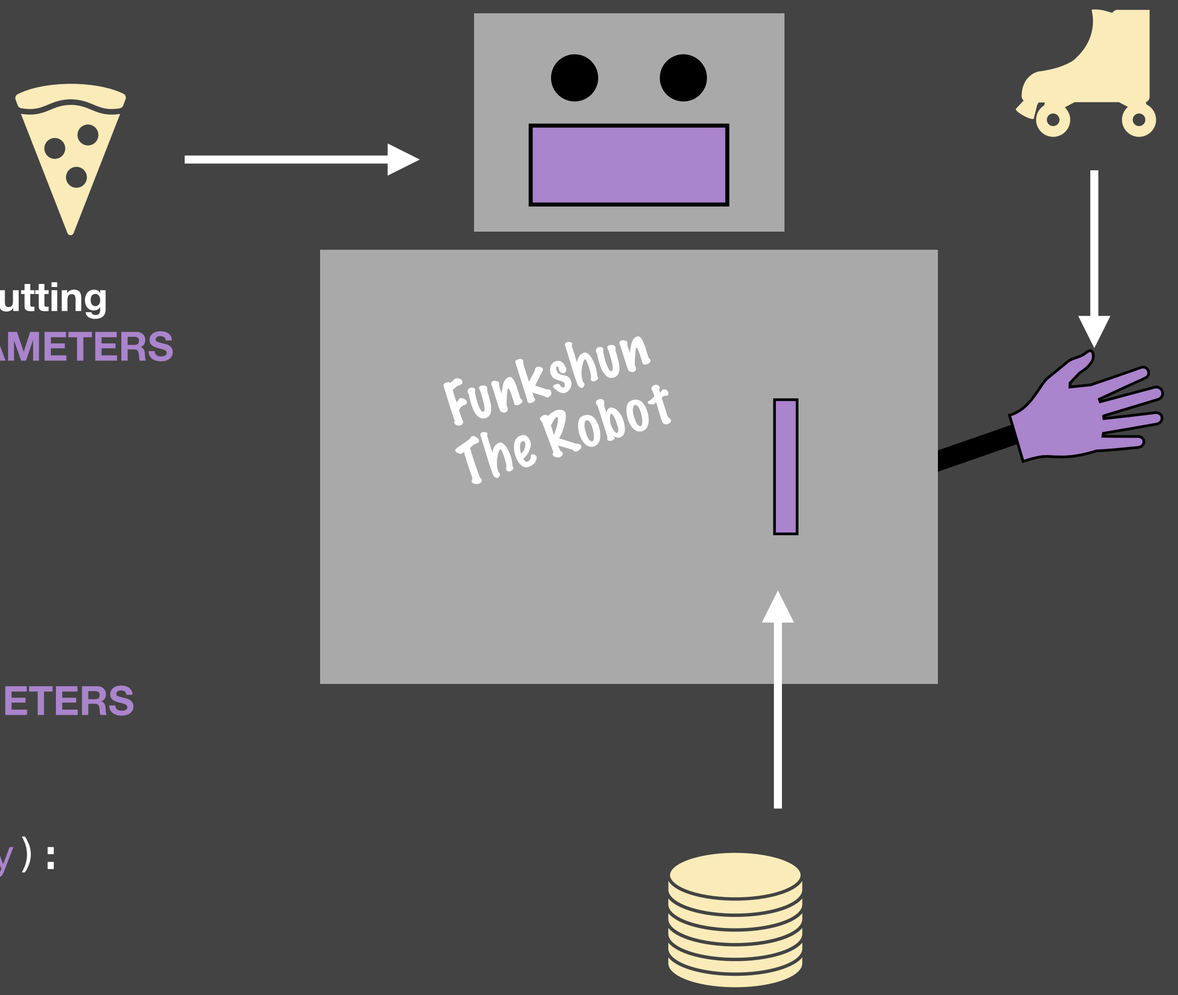
Today's agenda

- Functions review
- Data in and out of functions
  - Return keyword
  - Parameter types!
    - Required
    - Optional
    - How they affect arguments
- For Thursday: Read about objects, play Find the Function!

Function review

Tell me what you think are functions when you watch some code run

Structure of a Function



All the different ways of putting stuff into Funkshun are PARAMETERS

The items we put into Funkshun are ARGUMENTS

This function has 3 PARAMETERS

For our function to work, we give it 3 ARGUMENTS

```
def funkshun(size, x, y):
    <DO STUFF HERE>
```

```
funkshun(10, -100, 100):
```

Getting data in and out of functions

- Functions are like robot…there's only certain ways to get things in and out of the robot
- Getting things into the fun
  - Set up **parameters**, the put)
- Once things are in, you ca
- Getting things out of the function:
  - **return** keyword

```
def funkshun(p1,p2):
    SUM = p1 + p2
    return SUM
```

Function anatomy

define keyword

Parameters - names for variables used inside the function

Data goes into the function via parameters

```
def funkshun(param1, param2):
    '''DOCSTRING - what the function does'''
    SUM = param1 + param2
    return SUM
```

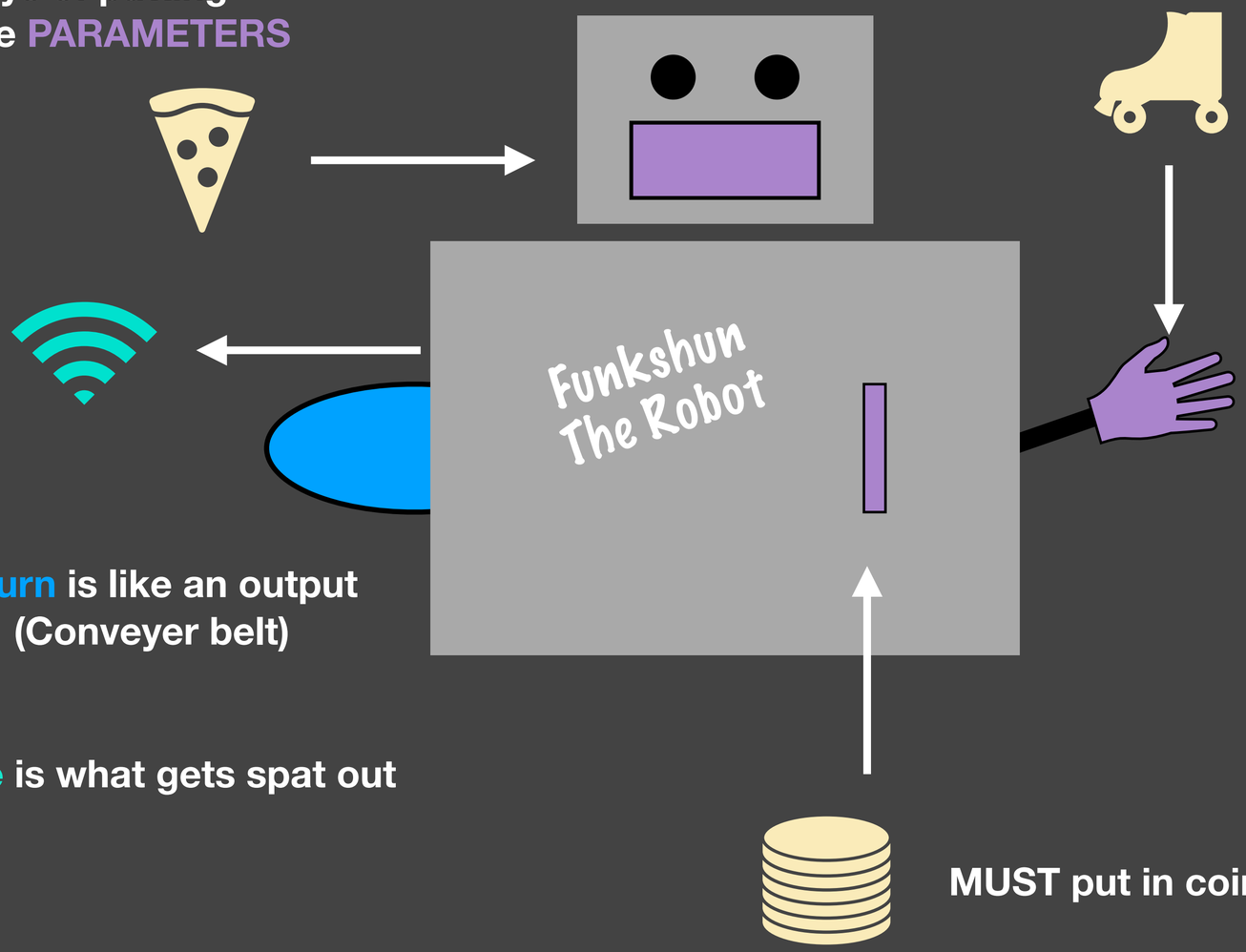
Parameters get used inside the function

Optional keyword - returns value(s) to the workspace outside the function

Data goes out via RETURN

Structure of a Function

All the different ways of putting stuff into Funkshun are PARAMETERS



The items we put into Funkshun are ARGUMENTS

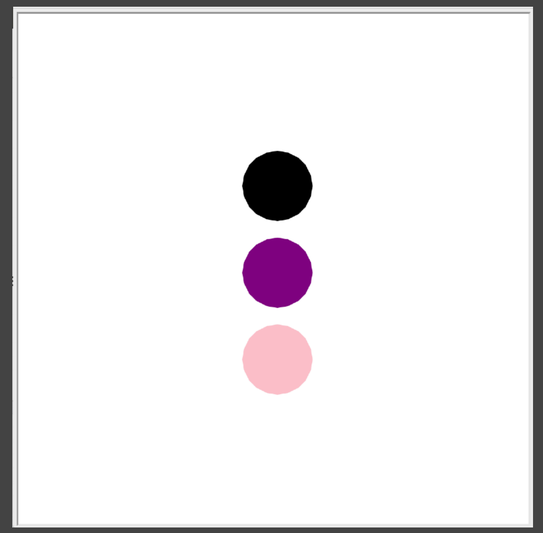
return is like an output (Conveyer belt)

value is what gets spat out

MUST put in coins to get the bot to run

Let's try it

```
1. def whichClicked(x, y):
2.     '''Determines which turtle in a list was clicked on'''
3.     for idx in range(len(turtList)):
4.         turtX = turtList[idx].xcor() # get x position of each bubble
5.         turtY = turtList[idx].ycor() # get y position of each bubble
6.         if turtX - 5 < x < turtX + 5 and turtY - 5 < y < turtY + 5:
7.             # see if click is within some range. Default is + or - 5 pixels
8.             # make the output of the function whichever index gives a True, above.
9.             return idx
```



- `whichClicked(0, -100) # line 57`
- `selected = whichClicked(0, -100)`

How do you call this function?

How would you get the index out of the function?

How would you save the output to a variable?

return keyword

- Place before the variable or value that you want to send out of the function
  - `return value`
- Typically the last line in a function but works in if statements too!
  - `if True:`
  - `    return value1`
  - `else:`
  - `    return value2`
- Functions with return keywords can have that value assigned to variables
  - `var = func(input)`

Types of parameters

```
1. def whichClicked(x, y):
2.     '''Determines which turtle in a list was clicked on'''
3.     for idx in range(len(turtList)):
4.         turtX = turtList[idx].xcor() # get x position of each bubble
5.         turtY = turtList[idx].ycor() # get y position of each bubble
6.         if turtX - 5 < x < turtX + 5 and turtY - 5 < y < turtY + 5:
7.             # see if click is within some range. Default is + or - 5 pixels
8.             # make the output of the function whichever index gives a True, above.
9.             return idx
```

What if we want to make another parameter for our buffer size (the area around the click that still counts as a click)

Types of parameters

```
1. def whichClicked(x, y, buffer):
2.     '''Determines which turtle in a list was clicked on'''
3.     for idx in range(len(turtList)):
4.         turtX = turtList[idx].xcor() # get x position of each bubble
5.         turtY = turtList[idx].ycor() # get y position of each bubble
6.         if turtX - buffer < x < turtX + buffer and turtY - buffer < y < turtY + buffer:
7.             # see if click is within some range. Default is + or - 5 pixels
8.             # make the output of the function whichever index gives a True, above.
9.             return idx
```

What if we wanted to set a default value for our new parameter buffer?

Types of parameters

```
1. def whichClicked(x, y, buffer=5):
2.     '''Determines which turtle in a list was clicked on'''
3.     for idx in range(len(turtList)):
4.         turtX = turtList[idx].xcor() # get x position of each bubble
5.         turtY = turtList[idx].ycor() # get y position of each bubble
6.         if turtX - buffer < x < turtX + buffer and turtY - buffer < y < turtY + buffer:
7.             # see if click is within some range. Default is + or - 5 pixels
8.             # make the output of the function whichever index gives a True, above.
9.             return idx
```

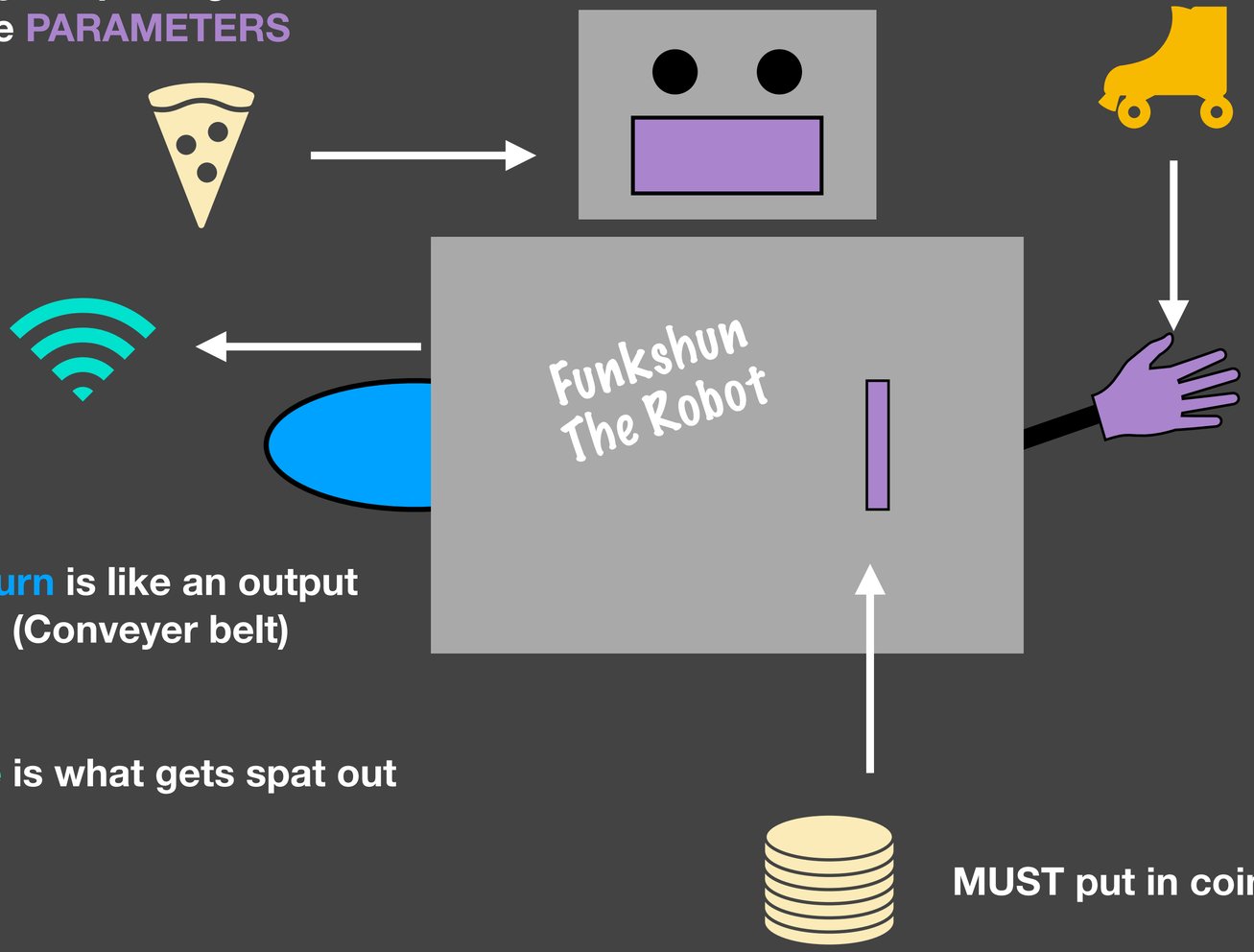
What happens when this function is called now?

```
whichClicked(0,0)
```
or
```
whichClicked(0,0,30)
```

Optional parameters means optional arguments!

Structure of a Function

All the different ways of putting stuff into Funkshun are PARAMETERS



Skate is optional

The items we put into Funkshun are ARGUMENTS

return is like an output (Conveyer belt)

value is what gets spat out

MUST put in coins to get the bot to run

Parameter Types: Required & Optional

- You can give a parameter a default value by defining the variable in the definition:
  - `def newFunc(reqParam, optParam=value):`
- Then, when you call the function, you can either roll with the default value, or update it!
  - `newFunc(reqArg, optArg)`
- Try changing the buffer value when you call `whenClicked`

Types of arguments

- **Positional argument**
  - Using the ORDER of parameters to assign the argument values
    - `def funcname(param1, param2, param3):`
    - `funcname(arg1, arg2, arg3)`
  - Have to have the arguments entered in the **correct order**.
  - How we've been using functions so far:
    - `turtle.color('color1','color2')`

Types of arguments

- **Keyword argument**
  - Using the NAME of parameters to assign the argument values
    - `def funcname(param1, param2, param3):`
    - `funcname(param2=arg1, param3=arg2, param1=arg3)`
  - Order doesn't matter!
  - How we've been using functions so far:
    - `turtle.Turtle(shape='circle',visible=False)`

Your Turn!

- Try calling whenClicked using **keyword arguments**
- Try changing around the order of arguments (inputs) and see what happens.

Review!

- Functions are like containers, and data goes in and out in specific ways.
- GETTING DATA OUT:
  - You can send data out of a function using the _____ keyword.
  - With the **`return`** keyword, you can then save the output of a function to a _____.
- GETTING DATA IN:
  - Set up the “doors” to the function using _____, which gives a name to some data the function will work with.
  - To give a function variable a default value, you can make an _____ **parameter**.
    - `def funcname(optParam=defaultValue):`

Review!

- Functions are like containers, and data goes in and out in specific ways.
- GETTING DATA IN:
  - Once you have “doors” set up, the data that goes through it is called an _____, which gives the value to some data the function will work with.
  - There are two types of arguments: _____ and _____
  - _____ uses the ORDER of the inputs to assign them to the right parameters.
  - _____ uses the NAMES of the parameters to assign values.
    - `funcname(param=value)`

How many?

- `def counter(start, limit):`
  - How many arguments (inputs) need to be entered to call this function?
  - `counter(0,10)`
- `def color(color1="black", color2="black"):`
  - How many optional parameters does this function have?
  - How many arguments (inputs) need to be entered to call this function?
- `def byeTurtle(x, y):`
  - How many optional parameters does this function have?
  - How many arguments (inputs) need to be entered to call this function?

What's legal?

- `color(color2="red", color1="pink"):`
  - What kind of arguments (inputs) are these?
- `counter(limit=10, start=0):`
  - How many parameters did this function have?
  - Which parameter came first in the function definition?
    - `Can't tell!`
- `def byeTurtle(x,y):`
  - …
- `byeTurtle(y=0, x=0):`
  - Is this allowed?
  - What kind of arguments (inputs) are these?

Apply Yoself!

- The **`whenClicked`** function requires x and y to be passed in, which comes from the **onclick** callback function

- Where do you think **`whenClicked`** gets called?
  A. On it's own
  B. In a callback function
  C. Inside a function definition for a callback

- Did you notice that we never defined or passed in `turtList`? How do you think this function gets access to that data?

```
1. def whichClicked(x, y, buffer=5):
2.     '''Determines which turtle in a list was clicked on'''
3.     for idx in range(len(turtList)):
4.         turtX = turtList[idx].xcor() # get x position of each bubble
5.         turtY = turtList[idx].ycor() # get y position of each bubble
6.         if turtX - buffer < x < turtX + buffer and turtY - buffer < y < turtY + buffer:
7.             # see if click is within some range. Default is + or - 5 pixels
8.             # make the output of the function whichever index gives a True, above.
9.             return idx
```

Check out selectTurtle.py and see if you can figure it out!!

For Thursday

- Read: Beginner's Guide to Object-Oriented-Programming (ignore the code)
- Find functions in old code, apps, programs, etc.
  Bring an example of one you think you found on Thursday!
- Thursday: scope, intro to objects
- selectTurtle.py on Canvas, Class Code Repo