

```
def isEven(num):
```



```
def isEven(num):
   if(num%2==0):
   else:
```



```
def isEven(num):
    if(num%2==0):
        return True
    else:
```



```
def isEven(num):
    if(num%2==0):
        return True
    else:
        return False
```



Write a function called isEven that takes an argument num and returns True if num is even and False otherwise.

```
def isEven(num):
    if(num%2==0):
        return True
    else:
        return False
```

Can this code be refactored?



# **Aside - Refactoring**

The process of taking code **that works** and changing how it's implemented to be more *efficient* (faster, less lines of code, etc.).

```
def avg(num1, num2):
    sum = num1+num2
    avg = sum / 2
    return avg

Refactored

def avg(num1, num2):
    return (num1+num2)/2

    return Academy
```

Write a function called isEven that takes an argument num and returns True if num is even and False otherwise.

```
def isEven(num):
    if(num%2==0):
        return True
    else:
        return False
```

Can this code be refactored?



## Code Appetizer - Refactored

#### **Original**

```
def isEven(num):
    if(num%2==0):
        return True
    else:
        return False
```

#### Refactored

```
def isEven(num):
    if(num%2==0):
        return True
    return False
```



#### **Functions - Return Statements**

never gets executed

When a return statement is reached inside of a function, the program immediately exits the function and does not run any more lines of code inside of that function.



## Code Appetizer - Refactored

#### **Original**

```
def isEven(num):
    if(num%2==0):
        return True
    else:
        return False
```

#### **More Refactored**

```
def isEven(num):
    return num%2==0
```

#### Refactored

```
def isEven(num):
    if(num%2==0):
        return True
    return False
```



## Code Appetizer - Refactored

#### **Original**

```
def isEven(num):
    if(num%2==0):
        return True
    else:
        return False
```

#### Refactored

```
def isEven(num):
    if(num%2==0):
        return True
    return False
```

#### **More Refactored**

```
def isEven(num):
    return num%2==0
```

This statement will give us back a boolean value: either True or False. And we can directly return that!



# **Agenda**

- 6:30 6:45 Code Appetizer
- 6:45 6:50 Rapid Review
- 6:50 7:15 Lists Lecture
- 7:15 7:50 Lists Exercise
- 7:50 8:00 Break
- 8:00 8:15 File I/O Lecture
- 8:15 8:50 File I/O Exercise
- 8:50 9:20 Bart Simulator
- 9:25 9:30 Exit Ticket



#### Name Game...

#### **Problem**

Names are hard for me...



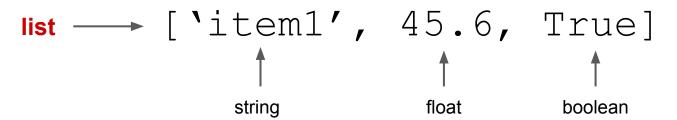
# Lists



#### What is a list?

A *list* is an ordered sequence of items.

The items can be different types.





#### What is a list?

#### A list can be assigned to a variable

```
favorites = ['blue', 8, 'soccer']
```

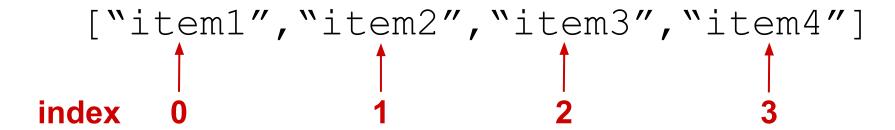
## or passed to a function as an argument.

```
reverse_list([1,2,3])
reverse_list(favorites)
```



#### What is a list?

Each item in a list is associated with an index. The indices start at 0 and go up from there.





# Defining a List - Formula

```
['red', 'yellow', 'blue']
```

- Lists must start and end with square brackets [ ].
- List items must be separated by commas , .
- A list with nothing inside is called an "empty list" and is defined by [].

  Hackbright Academy

## Accessing a List Item - Formula

Name of the list Square brackets Index of the element to access



# Accessing a List Item - Formula

- List items are accessed using their index.
   Indices start at 0, so the first item in the list is at index 0.
- List items can be accessed from the end by using a negative sign.

  \*\*Academy\*\*

### When to use a list?

When the order of elements matters

```
face_cards = ['J', 'Q', 'K', 'A']
```

 When there are many of a certain type of item.

```
groceries = ['apples', 'lemons', 'kale']
students = ['Carrie', 'Aom', 'Dely']
Hackbright
Academy
```

```
my list = [42, 'blanket', False, 3.14]
my list[2]
my list[0]
my list[-1]
my list[1:3]
my list[2:]
```



```
my list = [42, 'blanket', False, 3.14]
my list[2]
                     False
my list[0]
my list[-1]
my list[1:3]
my list[2:]
```



```
my list = [42, 'blanket', False, 3.14]
my list[2]
                      False
my list[0]
               \Rightarrow 42
my list[-1]
my list[1:3]
my list[2:]
```



```
my list = [42, 'blanket', False, 3.14]
my list[2]
                       False
                \Rightarrow 42
my list[0]
              \Rightarrow 3.14
my list[-1]
my list[1:3]
my list[2:]
```



```
my list = [42, 'blanket', False, 3.14]
my list[2]
                        False
my list[0]
                  \Rightarrow
                        42
                  \Rightarrow 3.14
my list[-1]
               \Rightarrow
my list[1:3]
                        ['blanket', False]
my list[2:]
                                             Hackbright
```

Academy

```
my list = [42, 'blanket', False, 3.14]
my list[2]
                           False
my list[0]
                    \Rightarrow
                           42
                           3.14
my list[-1]
                    \Rightarrow
                    \Rightarrow
my list[1:3]
                           ['blanket', False]
                    \Rightarrow
my list[2:]
                           [False, 3.14]
                                                   'Hackbright
                                                   Academy
```

# **Basic List Operations**

Python Expression	Result	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
1 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]:  print x,	1 2 3	Iteration Hackbri Acaden

#### **Exercise Time!**

Do the Lists Exercises 1 from the website!

