

# Computer Science Programming Fundamentals

## Functions

*$f_x$*

# 💡 Functions Intro

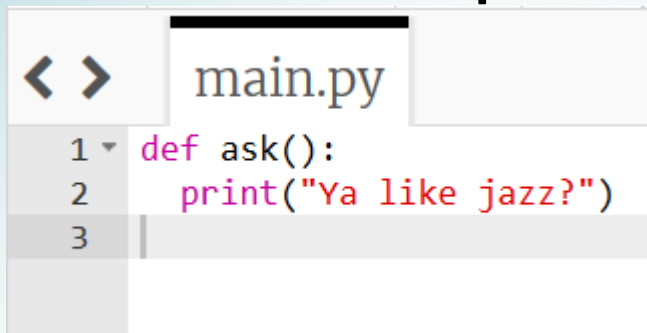
As we have seen, loops can help us reuse our code. But what if we want to use the same code in a different part of our program? Do we have to write it again there?

How can we test just one part of our code to make sure it's working correctly?

Let's learn how to reuse the code we need, whenever and wherever we need it. We'll also see a great way to make our code easier to read and understand.

# Functions pt 1

- Run the example:



```
< > main.py
1 def ask():
2     print("Ya like jazz?")
3
```

**Q:** Nothing happened. What's wrong?

**A:** If you build a car but you don't drive it, it won't help you. We've built our function, now we need to use it!

- Run:

```
def ask():
    print("Ya like jazz?")
```

```
ask()
```

The last line **calls** the function, telling Python to use it! You can call a function as many times as you like.

# Functions pt 2

`print()`, `input()`, and `range()` are all functions we've already seen.

Many times, we put something between the parentheses when we use those functions.

Things we put between the parentheses are called **parameters**. One function can take many parameters!

When we make our own functions, we can let them take parameters.

- Run:

```
def count(num):  
    for x in range(1, num + 1):  
        print(x)  
    if x == num:  
        print("Done!")  
  
count(10)
```

## Functions pt 3

In the code on the previous page, we defined a function called count that needs a parameter called `num`.

Then, we made a for loop and an if statement inside the function.

On the last line, outside the function, we called `count()` and we gave it the number 10 to work with.

**Q:** What does the function do with `num`?

**A:** `num` is used to tell Python how many times the for loop will run. Then, we print "Done!", but only if we are on the last time the loop runs.

# Functions pt 4

Some functions give us something when they are done running. Isn't that nice? This is called **returning**.

- Run:

```
def find_wallet():  
    print("Ooh, I found someone's  
wallet!")  
    return "the wallet"
```

```
print(find_wallet())
```

**Q: Why does this code print out "the wallet"?**

**A:** Because that's what the function returns. It returns "the wallet"! That is, a string that says "the wallet."



# Functions pt 5

What if this function is bad and wants to take some money from the wallet?

- Run:

```
money = 5
```

```
def find_wallet():
```

```
    money = money - 2
```

```
    print("Ooh, I found someone's wallet!")
```

```
    return "the wallet"
```

```
print(find_wallet())
```

**Q: Why the error?**

**A:** Python doesn't realize that when we use the variable `money` here, we are talking about the variable we defined outside the function, not the one we defined inside it.

Because it's confused, it's telling us that we can't use a variable before we finish assigning it.

# Functions pt 6

Variables made inside a function, loop, or if statement are **local variables**. We can only use them inside their home.

Variables made outside a function are **global variables**. We can use them anywhere in the file.

It takes a little bit of work to get around this error, but at least it's for a good cause, right?

- Run:

(Code on next page)



# Functions pt 7

```
money = 5
```

```
def find_wallet():  
    print("Ooh, I found someone's  
wallet!")  
    return "the wallet"
```

```
def steal(starting_amount,  
stolen_amount):  
    print("I just stole", stolen_amount,  
"dollars!")  
    return starting_amount -  
stolen_amount
```

```
money = steal(money, 2)
```

```
print(find_wallet(), "was returned  
with", money, "dollars in it.")
```

There are other ways to get around the problem, but some of them require a more advanced knowledge of Python. Notice the two parameters in the `steal()` function!

# Functions pt 7



## Activity:

- You got in trouble in class again! As a punishment, your teacher is going to make you write some things on the board over and over again until you learn your lesson.
- While the teacher isn't looking, make a function that writes whatever you need to write, however many times you need to write it.
- Starting code:

```
sentence1 = "I will not  
steal wallets."
```

```
sentence2 = "I will not chew  
gum in class"
```

```
sentence3 = "I will not  
stick said gum under the  
door handle"
```

# Functions pt 8

An ideal answer:

```
def write_bot(msg, amnt):  
    for x in range(1, amnt + 1):  
        print(msg)
```

```
write_bot(sentence1, amounts[0])  
write_bot(sentence2, amounts[1])  
write_bot(sentence3, amounts[2])
```

# Functions pt 9

## Explanation:

We define the sentences and amounts first.

Then, there is a function defined that takes two parameters. The function has a for loop that prints the given sentence the given amount of times.

Finally, we call the function three times, with a different sentence and amount each time. The amounts are given in a list, so students will have to remember how to access list items.