*Tags: Functions, F Strings, Type Casting, Len Function, Input Function,Arguments

Introducing *#Functions*

Reuasble actions that have a name
`func_name()`

- e.g. Print(), Type(), help()

#Arguments - fancy word for inputs, separate by commas
`burger(bun, tomato, bacon, cheese)`

The *#Len* Function

Length: len() function will return the length of whatever item we pass to it.

- numbers do not have a length, only sequences and strings do.

```
word = "chicken"
len(word)
7
```

The *#Input* Function

Input() prompts a user to enter some input, converts it into a string, and then returns it. We can use it to gather user input in our programs.

```
age = input("how old are you?")
print("you are" + age + "years old")
```

#Type Casting

The type() function accepts an input object and will return the type of that object

```
type("hi")
<class `str`>

type(55)
<class `int`
```

Casting types: used to change type

- int()
- float()
- str()

```
age = "19"
age = float(age)
19.0
```

Age Calculator Exercise

```
age = input("how old are you (in years) ?")
days = float(age) * 365
print(days)
```

#F Strings

f-strings are an easy way to generate strings that contain interpolated expressions. Any code between curly braces {} will be evaluated, and then the result will be turned into a string and inserted into the overall string. *code inside curly brackets is executed

```
f"there are {24*60*60} seconds in a day"
"there are 86400 seconds in a day"
```

- Often used with variables
- must use curly brackets {}, NOT parens ()

```
age = input("How old are you (in years) ?")
days = float(age) * 365
print(f"{age} years is {days} days old")
```

Shopping Cart Exercise

```
print("WELCOME TO THE USELESS STORE")

print("*" * 30)


item = input("What item are you purchasing: ")

price = float(input(f"What is the price of {item}: "))

quantity = float(input(f"How many {item} are you buying: "))


print(f"added {quantity} {item}(s) to shopping cart!")

print(f"Subtotal: ${quantity * price}")
```