# CSE 2040 Programming IV Lecture #25

# What will we learn about functions

- Function Decorator
- Anonymous Functions or Lambdas
- Lambdas Functions with filter()
- Lambdas Functions map()
- Lambdas Functions with reduce()

# Function Decorators

- Decorators are the most common use of higher-order functions in Python. It allows programmers to modify the behavior of function or class.
- Function Decorators:
  - Decorator is a function that accepts a function as parameter and returns a function. A decorator takes the result of a function, modifies the result and returns it.
  - Thus decorators are useful to perform some additional processing required by a function.

```
'''A decorator to increase the value of a function by 2 '''
def decor(fun):
    def inner():
        value = fun()
        return value+2
    return inner

#take a function to which decorator should be applied
def num():
    return 10

#call decorator function and pass num
result_fun = decor(num)  #result_fun represents 'inner' function
print(result_fun())      #call the result_fun and display the result
```

```
'''A python program to apply a decorator to a function using @ symbol'''

def decor(fun):
    def inner():
        value = fun()
        return value+2
    return inner

@decor
def num():
    return 10

#call the num() function and display the result
print(num())
```

```
result_fun = decor(num)
print(result_fun())
```

```
'''A python program to create two decorators.'''
def decor(fun):
    def inner():
        value = fun()
        return value+2
    return inner

def decor1(fun):
    def inner():
        value = fun()
        return value*2
    return inner

def num():
    return 10

result_fun = decor(decor1(num))
print(result_fun())
```

```
'''To apply two decorators to the same function using @ symbol'''
def decor(fun):
    def inner():
        value = fun()
        return value+2
    return inner

def decor1(fun):
    def inner():
        value = fun()
        return value*2
    return inner

@decor
@decor1
def num():
    return 10

print(num())
```

# Anonymous Functions or Lambdas

- Functions are defined using def keyword.
- A function without a name is called “anonymous function”.
  - Anonymous functions are not defined using “def”
  - They are defined using the keyword lambda and hence they are also called “Lambda functions”.

# Lambdas Functions

```
def square(x):
    return x*x
```

Normal Function

---

```
lambda x:x*x
```

Lambda Function
where **lambda** keyword, x is an argument, Colon **:** represents the beginning of the function that contains an expression x*x.

Syntax: **lambda argument_list : expression**

# Lambdas Functions

```
def square(x):
    return x*x
```

y = square(5)

---

lambda x:x*x

Lambda functions return a function and hence they should be assigned to a function as:

f = lambda x: x*x

```
f = lambda x: x*x
value = f(5)
print('Square of 5 = ',value)
```

here, 'f' is the function name to which the lambda expression is assigned. Now, if we call the function f() as:

value = f(5)

# Lambdas Functions Example

```
f = lambda x, y: x+y
result = f(1.55, 10)
print('sum = ',result)
```

```
max = lambda x,y: x if x>y else y
a, b = [int(n) for n in input("Enter two numbers: ").split(',')]
print('Bigger number = ', max(a,b))
```

# Lambdas Functions with filter()

- filter() function is useful to filter out the elements of a sequences of a sequence depending on the result of a function.
  - Syntax : **filter(function, sequence)**
- Here, the 'function' represents a function name that may return either True or False;
- 'sequence' represents a list, string or tuple

filter_test.py on LMS!

# Lambdas Functions map()

- map() function is similar to filter() function but it acts on each element of the sequence and perhaps changes the elements.
- Syntax: **map(function, sequence)**
- Here, the 'function' performs a specified operation on all the elements of the sequence and the modified elements are returned which can be stored in another sequence.

# Lambdas Functions with reduce()

- reduce() function reduces a sequence of elements to a single value by processing the elements according to a function supplied.
- reduce()  function belongs to functools module
- Syntax: **reduce(function, sequence)**
- E.g: **lst=[1,2,3,4,5]**

  **reduce(lambda x, y: x*y, lst)**

Reduce the list into a final value: 120

# References


- Dr. R. Nageswara Rao, Core Python Programming, Second Edition, 2018
  - Page – 266-270