Programming Styles

Thomas Schwarz, SJ

Programming Styles

- Styles of Programming
  - Imperative Programming:
    - Describe in detail how computation proceeds
    - Basically, change states of variables
    - This is what we practiced up till now

Programming Styles

- Functional Programming
  - Define functions
    - Specify program behavior by executing nested functions
    - Pure functional programming: No variables that capture a state
  - Advantage: Easier to prove programming correctness

Programming Styles

- Declarative Programming
    - Specify what a program should do
        - System figures out how to do it.
        - Example 1: Prolog (Classic AI programming language)
            - Specify rules in Prolog:
                - `animal(X) :- cat(X)` means every cat is an animal
                - `?- cat(tom).` means that tom is a cat
            - You can ask about the world defined by these rules
                - `?- animal(X).` asks for what things are animals
            - Prolog consists of rules and base facts, then on its own finds out other facts.

Programming Styles

- Declarative Programming:
  - Example 2: SQL — Database Language
    - Database consists of relations stored in various tables
    - Example:

Marquette_ID	First_Name	Family_Name	Address
123123007	David	Roy	1984 31st Street, Milwaukee, WI 54321
97007007	Thomas	Schwarz	4821 Wisconsin Ave, Milwaukee, WI 54213
14309873	Joseph	Cuelho	9821 12th Avenue, Milwaukee, WI 54321
90874132	Donald	Drumpf	321 Pennsylvania Ave, Madison, WI 32451

Programming Styles

- Declarative Programming:
  - Example SQL:
    - SQL statement describes all combinations of record pieces

```
SELECT first_name, family_name FROM
addresses, classes

WHERE classes.name = “COSC1010” and
classes.role = “instructor” and
classes.id = addresses.id
```

Programming Styles

- Declarative Programming:
    - Example SQL:
        - SQL statement describes all combinations of record pieces
        - How the database engine performs the query is **not** specified
        - In fact, for complicated queries, the database will try out several ways before selecting the actual algorithms

Programming Styles

- Object-Oriented Programming
  - Program defined various objects
    - Objects have data and methods
      - E.g. Marquette Persons have IDs, names, addresses, …
      - Classes have lists of participants
  - We will learn Object-Oriented (OO) programming in this class

Functional Programming In Python

- Recall anonymous functions
  - We can define a function with the lambda expression
  - Example:

```
lambda x, y: (x+y)/(x**2+y**2+1)
```

    - creates a function without name

```
def fun(x,y):
    return (x+y)/(x**2+y**2+1)
```

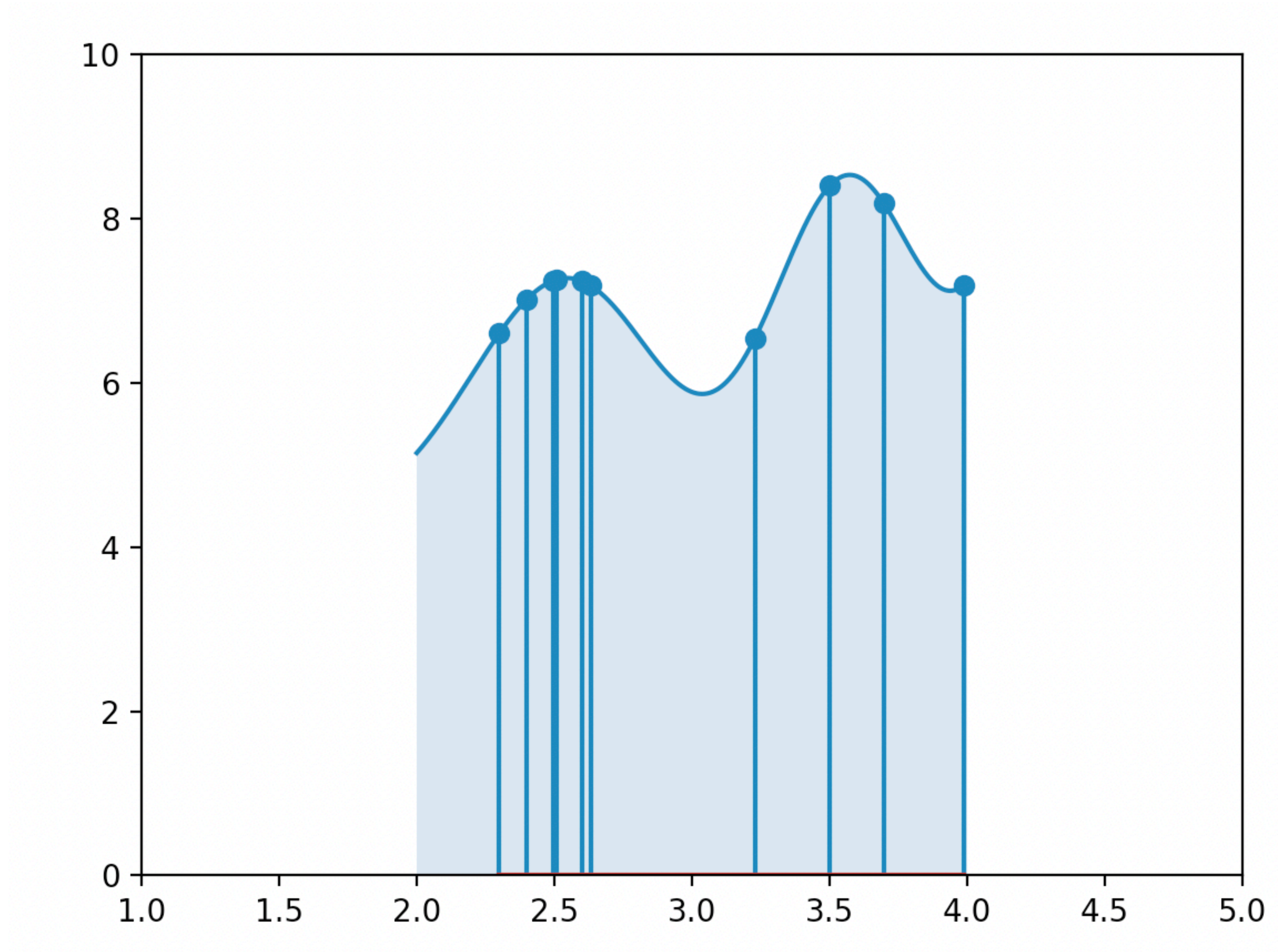
    - creates the same function with a name

Functional Programming in Python

- Python can use functions as arguments in functions
- Python can use functions as return values

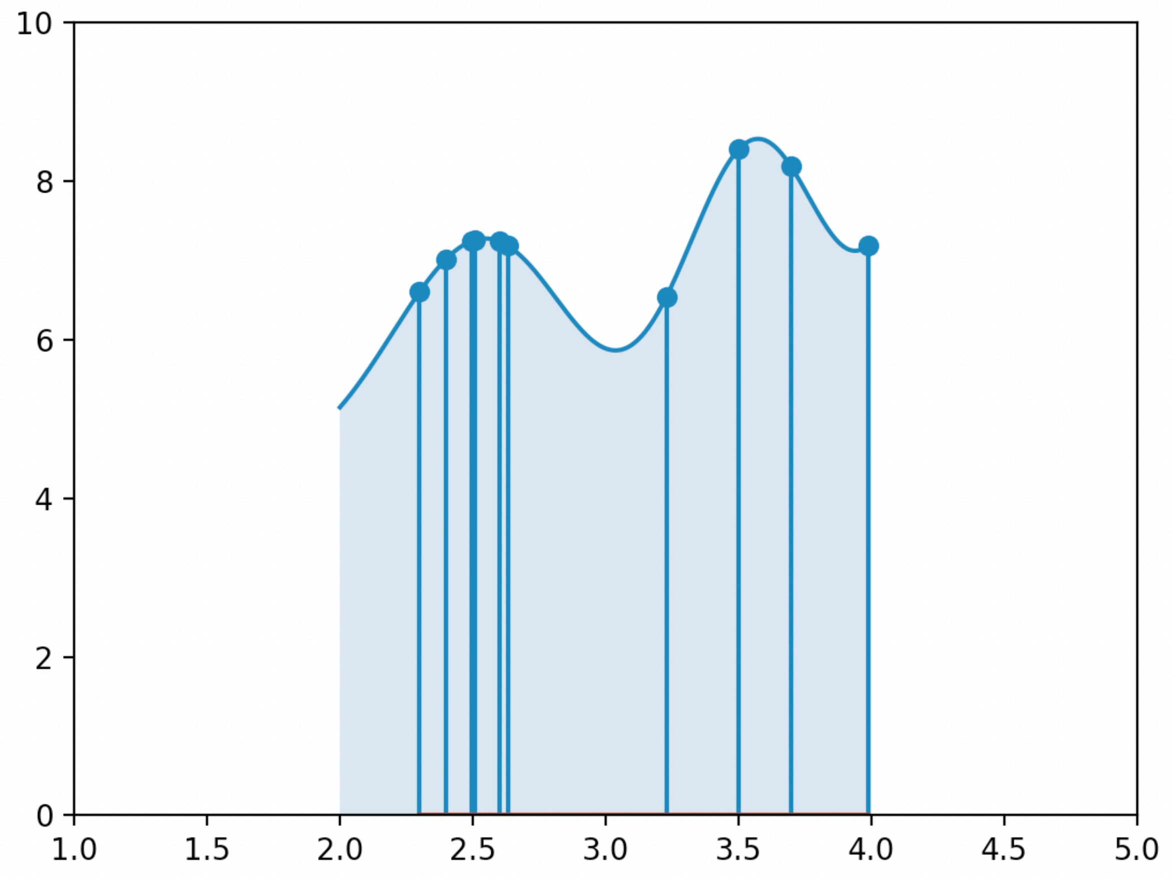
Functional Programming in Python

- Monte Carlo Integration of a function: $\int_a^b f(x)dx$
    -



Functional Programming in Python

- Idea: Select *n* random points between a and b
- Calculate the function value on these points
- Calculate their average
- Multiply with (b-a)
- This is an estimate
    - for the integral



Functional Programming in Python

- Implementation

```
import random as rd

def integral(fun, a, b, nr_of_points):
    sum_of_f = 0
    for _ in range(nr_of_points):
        sum_of_f += fun(rd.uniform(a,b))
    mean_of_fs = sum_of_f/nr_of_points
    return mean_of_fs * (b-a)
```

Functional Programming in Python

- Implementation

```
import random as rd

def integral(fun, a, b, nr_of_points):
    sum_of_f = 0
    for _ in range(nr_of_points):
        sum_of_f += fun(rd.uniform(a,b))
    mean_of_fs = sum_of_f/nr_of_points
    return mean_of_fs * (b-a)
```

Contains now

$f(r_1) + f(r_2) + \ldots + f(r_n)$

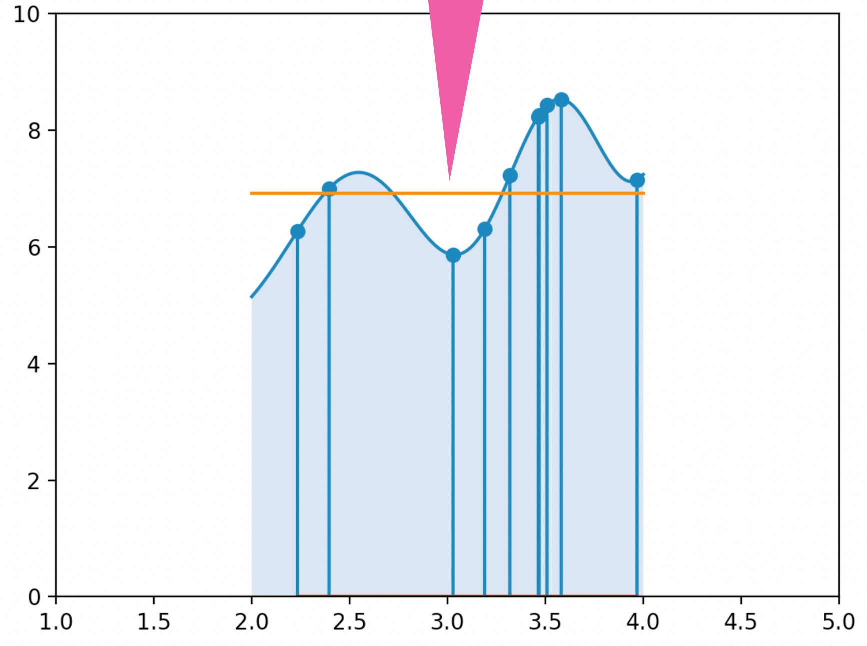
Functional Programming in Python

- Implementation

```
import random as rd

def integral(fun, a, b, nr_of_points):
    sum_of_f = 0
    for _ in range(nr_of_points):
        sum_of_f += fun(rd.uniform(a,b))
    mean_of_fs = sum_of_f/nr_of_points
    return mean_of_fs * (b-a)
```

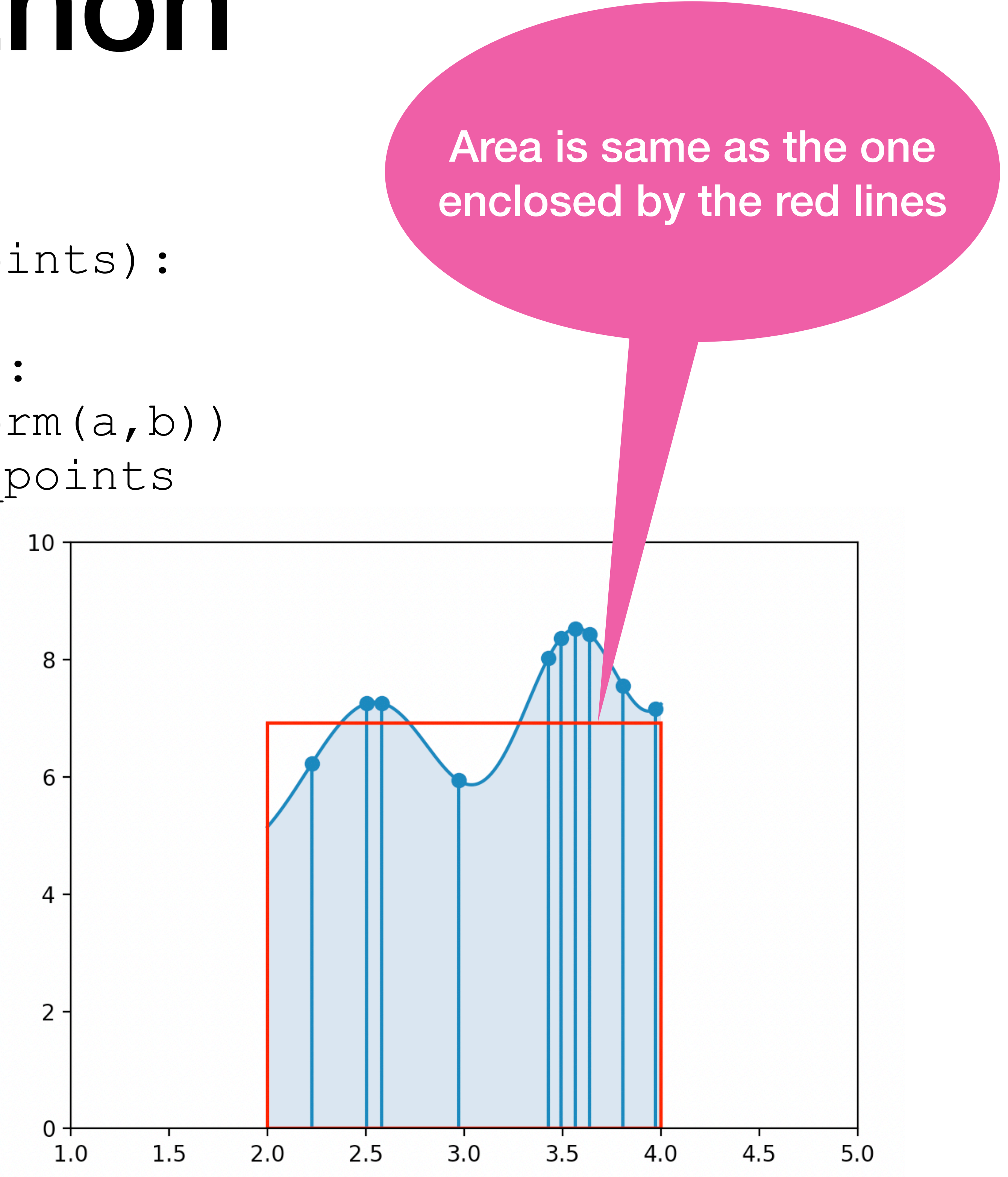
Contains now average function value (estimated)



Functional Programming in Python

```
import random as rd

def integral(fun, a, b, nr_of_points):
    sum_of_f = 0
    for _ in range(nr_of_points):
        sum_of_f += fun(rd.uniform(a,b))
    mean_of_fs = sum_of_f/nr_of_points
    return mean_of_fs * (b-a)
```



Functional Programming in Python

- We can try this out using a lambda expression

```
for _ in range(20):
    print(integral(lambda x: 5+x**2/5+math.cos(x**2),2,4,1000))
```

```
13.848738547694959
13.84527895244639
14.017133065965641
13.832408510459572
13.873003112163618
13.75656692127387
13.87544139865498
13.807364056962463
13.819856780953849
13.785217661244614
13.782351635543083
13.818585721154232
13.842831044731115
13.853038466753764
13.832754911335963
```

Functional Programming in Python

- Returning a function
    - Calculate a random polynomial of degree 3
        - Idea:
            - generate four “random” coefficients
            - create the polynomial
            - return it

Functional Programming in Python

- Implementation

```
import random as rd

def random_poly( ):
    a = rd.randint(-3,3)
    b = rd.randint(-5,5)
    c = rd.randint(-5,5)
    d = rd.randint(-2,2)
    return lambda x: a*x**3+b*x**2+c*x+d
```

Functional Programming in Python

- Implementation without using lambda

```
def random_poly_d( ):
    a = rd.randint(-3,3)
    b = rd.randint(-5,5)
    c = rd.randint(-5,5)
    d = rd.randint(-2,2)
    def inner(x):
        return a*x**3+b*x**2+c*x+d
    return inner
```

Create a new function

Functional Programming in Python

- Implementation without using lambda

```
def random_poly_d( ):
    a = rd.randint(-3,3)
    b = rd.randint(-5,5)
    c = rd.randint(-5,5)
    d = rd.randint(-2,2)
    def inner(x):
        return a*x**3+b*x**2+c*x+d
    return inner
```

And return it