Outline

1. Simulation
   - estimating wait times
   - top down design

2. Functions in Python
   - definition and arguments
   - body and return statement
top down design
functions in Python

1. Simulation
   - estimating wait times
   - top down design

2. Functions in Python
   - definition and arguments
   - body and return statement
What If Scenarios
should we buy another printer?

Estimate wait times for printer jobs to finish.

Problem: printer shared by several users.

- the printer queues jobs along FIFO protocol
- arrival times are uniformly distributed
- length of the jobs is normally distributed

Given $n$ jobs arriving uniformly in time interval $[0, T]$, with average length $\mu$ and standard deviation $\sigma$;

what is the average wait time?
And what is the standard deviation of the wait times?
Simulation
- estimating wait times
- top down design

Functions in Python
- definition and arguments
- body and return statement
top down design
divide work in separate tasks and conquer

Observe the logical division of actions: submitting jobs and printing jobs are separate tasks.

Key point: separate making of jobs from processing jobs.

Five tasks:

1. ask the user for the parameters of the simulation
2. make jobs, generate arrival and processing times
3. simulate printing and compute wait times
4. compute average wait time and standard deviation
5. print the results

There will be one main program and five functions, one function for each task.
tree structure
a hierarchy of tasks

Plus one utility function to apply format ‘.2f’
to lists of floats.
data flow

Input/Output descriptions for the five tasks:

1. ask the user for the parameters of the simulation
   output: \((n, T, \mu, \sigma)\)

2. make jobs, generate arrival and processing times
   input: \((n, T, \mu, \sigma)\)
   output: lists \(A\) and \(P\) with times

3. simulate printing and compute wait times
   input: lists \(A\) and \(P\) with times
   output: list of waiting times \(W\)

4. compute average wait time and standard deviation
   input: list of waiting times \(W\)
   output: average \(a\) and deviation \(d\)

5. print the results
   input: average \(a\) and deviation \(d\)
busy state \( b \) of the printer

\[
W = [0]; b = P[0]; i = 1
\]

\[
i < \text{len}(A)\?
\]

- False: return \( W \)
- True:

\[
t = A[i] - A[i-1]
\]

\[
b = 0 \quad t \geq b? \quad False \quad b = b - t
\]

- True:

\[
W\.append(b)
\]

\[
b = b + P[i]
i = i + 1
\]
1. Simulation
   ● estimating wait times
   ● top down design

2. Functions in Python
   ● definition and arguments
   ● body and return statement
definition and arguments

The general syntax of a function definition is:

```python
def <function name> ( <arguments> ):
    <function body>
```

All statements in the body must be indented!

For the arguments, we distinguish between:

1. required arguments that always must be given
2. optional arguments have default values

The normal probability density function `npdf`:

```python
def npdf(arg, mean=0, sigma=1):
```

has one required argument: `arg` and two optional arguments: `mean` (with default value 0), standard deviation `sigma` (with default value 1).
top down design
functions in Python

1. Simulation
   - estimating wait times
   - top down design

2. Functions in Python
   - definition and arguments
   - body and return statement
The body of the function starts typically with a documentation string, to document the function: what it does, its arguments.

```python
def npdf(arg, mean=0, sigma=1):
    "normal probability density function"
```

In a Python shell:

```python
>>> import npdf
>>> help(npdf)
NAME
    npdf - Same name for function and module.
FILE
    /Users/jan/Courses/MCS260/Summer23/Lec13/npdf.py
FUNCTIONS
    npdf(arg, mean=0, sigma=1)
        normal probability density function
```
the body of a function — call with keyword arguments

Defining \( f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \), mean is \( \mu \), \( \sigma \) is the standard deviation.

def npdf(arg, mean=0, sigma=1):
    "normal probability density function"
    from math import pi, sqrt, exp
    result = exp(-(arg - mean)**2/(2*sigma**2))
    result = result/(sigma*sqrt(2*pi))
    return result

Saved in the file npdf.py, use as

```python
>>> import npdf
>>> y1 = npdf.npdf(2,2.3,0.1)
>>> y2 = npdf.npdf(mean=2.3, sigma=0.1, arg=2)
>>> y1 == y2, and y2 is computed via keyword arguments.
The names arg, mean and sigma are formal parameters.
```
def f(x):
    return x**2

a = 2
b = f(a)

- The argument \(x\) of \(f\) is a **formal parameter**.
- At the call \(b = f(a)\),
  the formal parameter with name \(x\) refers to the object which is referred to by the **actual parameter** with name \(a\).
- The **return** statement assigns the object that holds the value \(x**2\) to the variable \(b\).
- Note that \(a\) and \(b\) are **outside the scope** of \(f\): the definition of \(f\) cannot use \(a\) nor \(b\).
  Moreover: \(x\) does not exist outside the definition of \(f\).
execution of function call — before the call

def f(x):
    return x**2

a = 2
b = f(a)

after a = 2:

```
name  refers to  object
  a  2
```
def f(x):
    return x**2

a = 2
b = f(a)

after a = 2:

calling f(a):

name

object

name

refers to

2

x
**execution of function call — after the call**

```python
def f(x):
    return x**2

a = 2
b = f(a)
```

**after** \(a = 2\):

- **name** \(a\) refers to object \(2\)

**calling** \(f(a)\):

- **name** \(x\) refers to object \(4\)

**after** \(b = f(a)\):

- **name** \(b\)
Exercises

1. A word is a palindrome if it reads the same backwards as forwards, e.g.: racecar. Draw the flowchart for an algorithm to decide if a string is a palindrome.

2. Write a Python function for exercise 1, which takes as input a string s and returns True if s is a palindrome, False otherwise.

3. Add print statements in the Simulate function of simuwait.py to set up a table that records all values of t and b.

4. Adjust the program simuwait.py for multiple printers.

5. Let c contain the coefficients of a polynomial of degree d which we want to evaluate at x. Write a function to compute

   \[ y = c[d] \times x^d + \ldots + c[1] \times x + c[0]. \]

6. Let c contain the coefficients of a polynomial of degree d which we want to evaluate at x. Write a function to compute

   \[ y = ( \ldots ((c[d] \times x + c[d-1]) \times x + c[d-2]) \times x + \ldots ) \times x + c[0]. \]