# Outline

1. Anonymous Functions with lambda
   - guessing a secret number
   - mapping functions to lists
2. Storing Data in Functions
   - a remember table
   - algorithms and data structures

MCS 260 Lecture 15
Introduction to Computer Science
Jan Verschelde, 30 June 2023

# lambda and list comprehensions

# guessing a secret number

A little game: try to guess a number.

Instead of storing the secret explicitly, we use *an oracle*.

For a given input, the oracle will return `True` if the input matches the secret and returns `False` otherwise.

Our number guessing game with an oracle:

```
oracle = generate_secret()
repeat
    guess = input('Give a number : ')
until oracle(guess)
```

The function `oracle()` is a function computed by the function `generate_secret()`, using `lambda`.

# oracles and trapdoor functions

## password security

Guarding of passwords:

- the password is encrypted,
- only the encrypted password is saved on file.

Password verification consists in

1. calling the encryption algorithm on user input,
2. checking if the result of the encryption equals the encrypted password stored on file.

The encryption algorithm acts as an oracle.

The oracle is typically a *trapdoor* function:

1. efficient to compute output for any input,
2. very hard to compute the inverse of an output.

# lambda and list comprehensions

1. Anonymous Functions with lambda
   - guessing a secret number
   - mapping functions to lists
2. Storing Data in Functions
   - a remember table
   - algorithms and data structures

# List Comprehensions

A *list comprehension* is a syntax to go through all elements of a range or list, in order to make a new list.

Let `f` be some function and `L` be a list, then

```
R = [f(x) for x in L]
```

`R` has the values of the function mapped to all elements in `L`.

To select all even numbers from the list `L`:

```
F = [x for x in L if x % 2 == 0]
```

In this list comprehension, the `if` test is a *conditional expression*, used to filter elements from a list.

The `zip()` combines two lists into a list of tuples,
useful to reduce lists to a single element when applied repeatedly.

# Monte Carlo without Loops

## a functional implementation

Recall the Monte Carlo method to estimate $\pi$:

1. generate $n$ points $P$ in $[0,1] \times [0,1]$
2. $m := \{ (x,y) \in P : x^2 + y^2 \leq 1 \}$
3. the estimate is then $4 \times \#m/n$

Main ingredients in *a functional implementation*:

1. `X = [random.uniform(0, 1) for _ in range(n)]`
2. `Y = [random.uniform(0, 1) for _ in range(n)]`
3. `P = list(zip(X, Y))` returns a list of tuples
4. `R = [(x, y) for (x, y) in P if x**2 + y**2 <= 1]`
5. `E = 4.0*len(R)/n`

*without explicit loops!*

# lambda and list comprehensions

1. Anonymous Functions with lambda
   - guessing a secret number
   - mapping functions to lists

2. Storing Data in Functions
   - a remember table
   - algorithms and data structures

# Functions to Store Data

Consider the definition of the function:

```
def fun(item, data=[]):
    data.append(item)
    print(data)
```

- `data` is the default argument, initialized to a list,
- `data.append(item)` appends `item` to `data`.

The first time the function is called:

1. A list is made in memory and assigned to `data`.
2. The value of `item` is appended to the list `data`.

The next times the function is called:

1. The same list in memory is used as `data`.
2. The value of `item` is appended to the list `data`.

# lambda and list comprehensions

1. Anonymous Functions with lambda
   - guessing a secret number
   - mapping functions to lists
2. Storing Data in Functions
   - a remember table
   - algorithms and data structures

# Algorithms and Data Structures

## a summary of Python in the small

Niklaus Wirth: programs = algorithms + data structures

Three basic control structures in any algorithm:

1. sequence of statements
2. conditional statement: if else
3. iteration: while and for loop

For every control structure,
we have a matching data structure:

| | control structures | data structures |
|---|---|---|
| 1 | sequence | tuple |
| 2 | if else | dictionary |
| 3 | while / for | list |

# Exercises

1. Generate the list [(1,1),(1,2),(1,3),(1,4), .. ,(1,$n$)], for any given $n$.
   Use this list then to create all fractions $1.0/k$, for $k$ from 1 to $n$.
   Finally, use `round()` to round all fractions to two decimal places.
2. Approximate the exponential function as $\sum_{k=0}^{n} \frac{x^k}{k!}$.

   Write list comprehensions to evaluate this approximation
   for given $x$ and $n$.
3. Use list comprehensions to generate points $(x, y)$
   uniformly distributed on the unit circle: $x^2 + y^2 = 1$.
   For some angle $t \in [0, 2\pi]$: $x = \cos(t)$, $y = \sin(t)$.
4. Add the option `feedback` to the guessing of a secret:
   if `feedback=True`, then `too small` or `too large` is printed,
   if the guess is too small or too large.