COP 3035

# Intro Programming in Python

Summer 2024

# Lecture 13 – part 1

Homework 4 – 06/28/24
Lab 7 – 07/01/24

# Lecture 13 – part 2

# Review

# Review

Dictionary comprehensions

Ternary operator

# Dictionary Comprehensions

- **Like list comprehensions**, they offer a shorter syntax for creating dictionaries when compared to using loops.

**Syntax form:**

```
{key: value  for variable in iterable}

{key: value  for variable in iterable if condition}
```

**Examples:**

```
squares = {}
for x in range(6):
    squares[x] = x**2
Print(squares)

{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

```
squares = {x: x*x for x in range(6)}
print(squares)


{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

# Ternary Operator

- The ternary operator in Python is a concise way to execute simple **if-else** statements in a **single line**.
- It is also known as the conditional expression.
- The basic syntax of the ternary operator is:

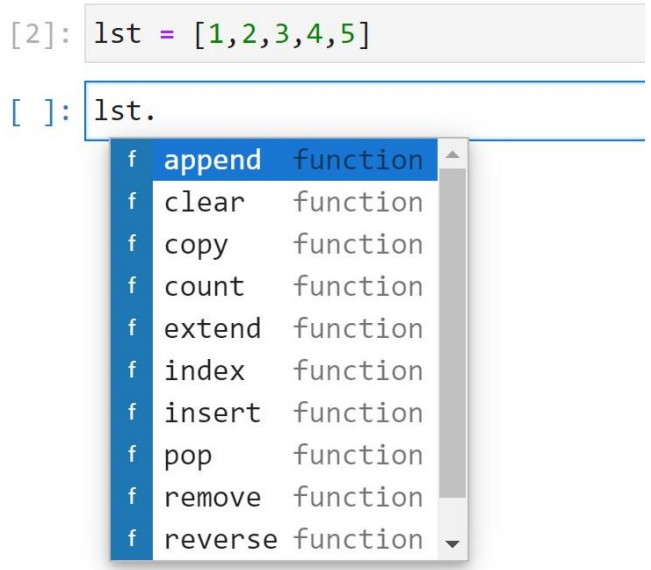
```
(a if condition else b)
```

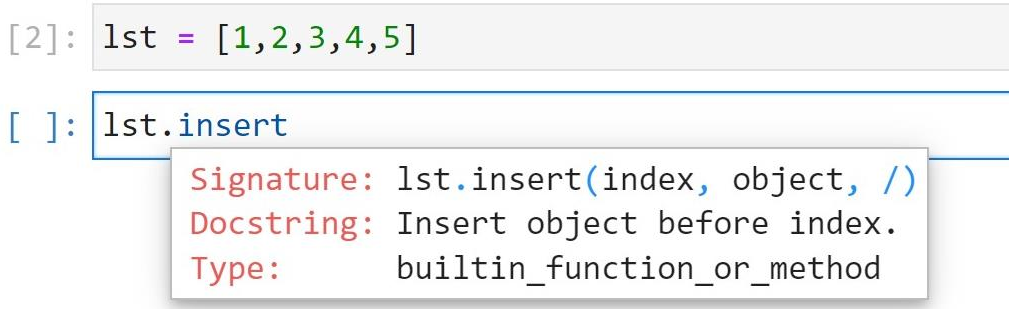
```
x = 10
result = "Greater than 5" if x > 5 else "Less than or equal to 5"
print(result)

Greater than 5
```

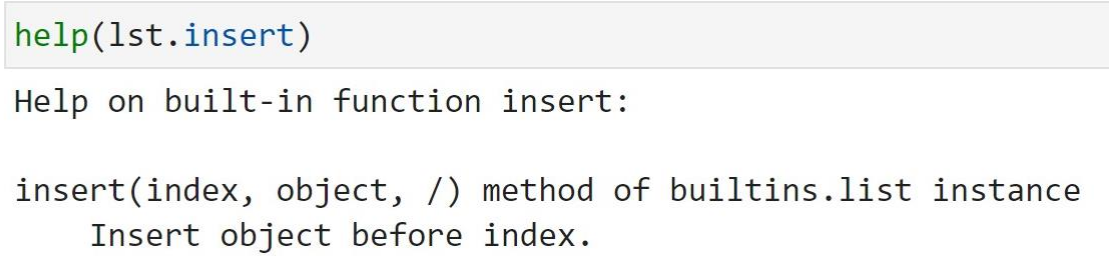
# Built-in Methods

https://docs.python.org/

- Use Shift+Tab in the Jupyter Notebook to get more help about the method.
- You can also use the **help()** function:



# What is a function ?

- A function is a valuable tool that groups a set of statements together, allowing them to be executed multiple times.
- This prevents us from having to write the same code repeatedly.

**Function Syntax**

```
def name_of_function(arg1,arg2):
    '''
    This is where the function's Document String (docstring) goes.
    When you call help() on your function it will be printed out.
    '''
    # Do stuff here
    # Return desired result
```

# Lecture 13 – part 3

# More on Functions

# Example

1. Return the square number from a list of numbers.

# Example

2. A function that returns the letter grade. The inputs are the grade number and the grade conversion table.

# Challenge

- Create as many functions as you can from the HW 3 appendix and class discussion group.

# Lecture 13 – part 4

# Lambda Expressions

# Lambda Expressions

- Lambda expressions allow us to create "anonymous" functions.
- We can quickly make ad-hoc functions without needing to properly define a function using def.
- Lambda's body is a single expression, not a block of statements.

```
def square(num):
    result = num**2
    return result
```

```
def square(num): return num**2
```

```
lambda num: num ** 2
```

```
square = lambda num: num **2
```