Introduction To Python Programming

Introduction to Python programming Course Outline

- Intro to Computer Science
- Environment Setup (Anaconda)
- Command Line
- Conda & pip package managers
- Jupyter Notebook
- Input & Output
- Variables
- Data types
- Numbers & Math
- Boolean & Comparison and Logic
- If Conditions
- For & While Loops
- Strings
- Lists
- Tuples
- Sets
- Dictionaries
- Advanced If and Loops
- List Comprehensions
- Dictionary Comprehensions
- Exceptions
- **File Handling**
- Functions
- Built-in functions & Operators (zip, enumerate, range, …)
- Map, Filter, Reduce
- Lambda Expressions

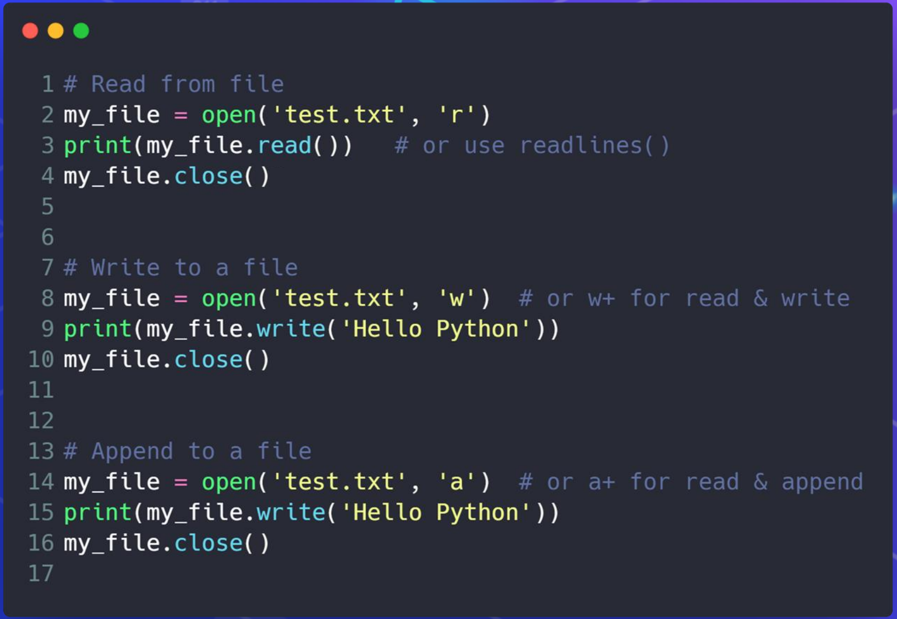
File Handling

Python supports handling of various file types, one example is text files

Python can open text files in three modes:

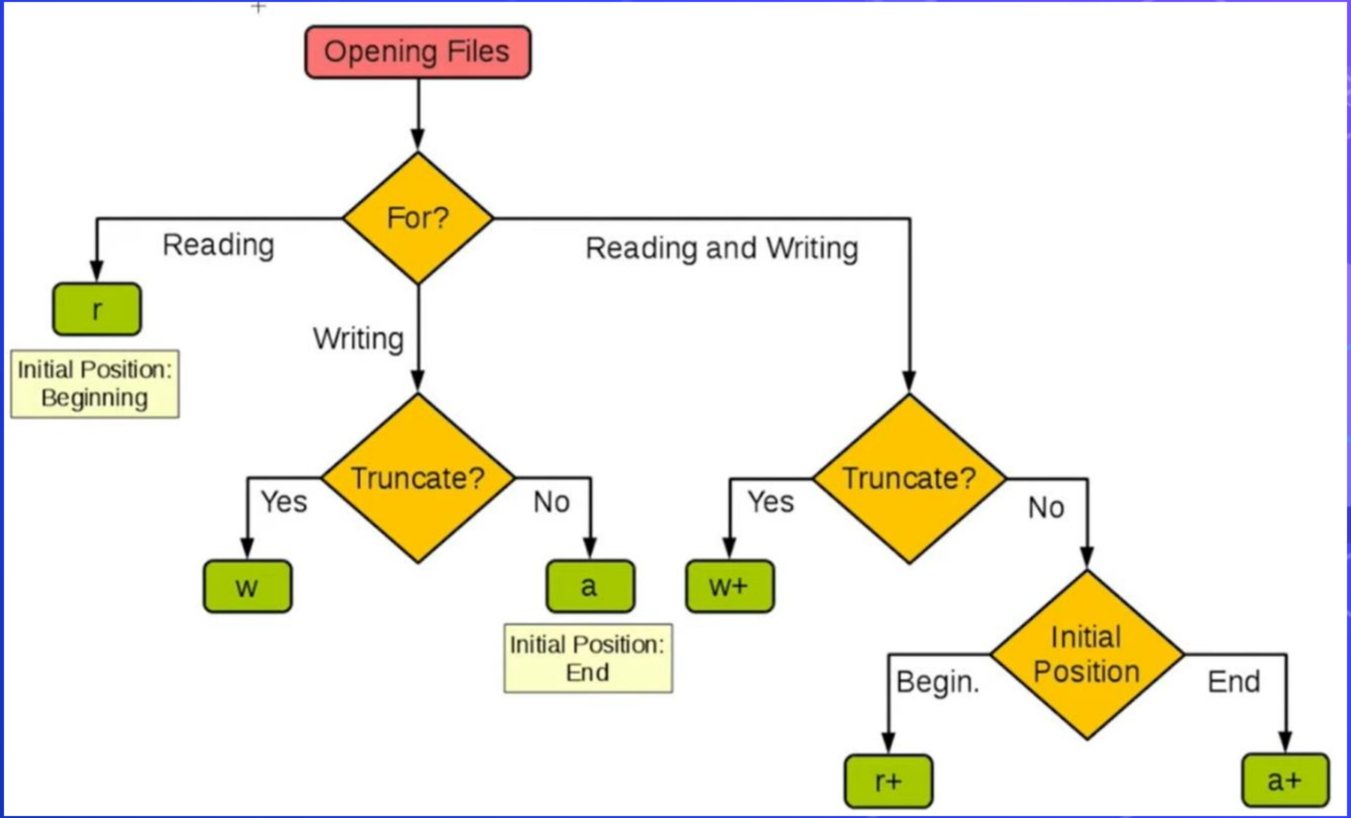
- Read mode (r)
- Write mode (w)
- Append mode (a)

Handling files with Python is very important since most of our data is stored in files of different types



```python
# Read from file
my_file = open('test.txt', 'r')
print(my_file.read())   # or use readlines()
my_file.close()


# Write to a file
my_file = open('test.txt', 'w')  # or w+ for read & write
print(my_file.write('Hello Python'))
my_file.close()


# Append to a file
my_file = open('test.txt', 'a')  # or a+ for read & append
print(my_file.write('Hello Python'))
my_file.close()

```

File Handling



Practice #1

Read the file :

1. Count the number of comments in this session by count the number of lines
2. Count the number of comments for each name

Task #1

Read the file :

1. Print each line in the file in a separate line
2. Append in the end of the file the phrase “My name is your_name”

Introduction to Python programming Course Outline

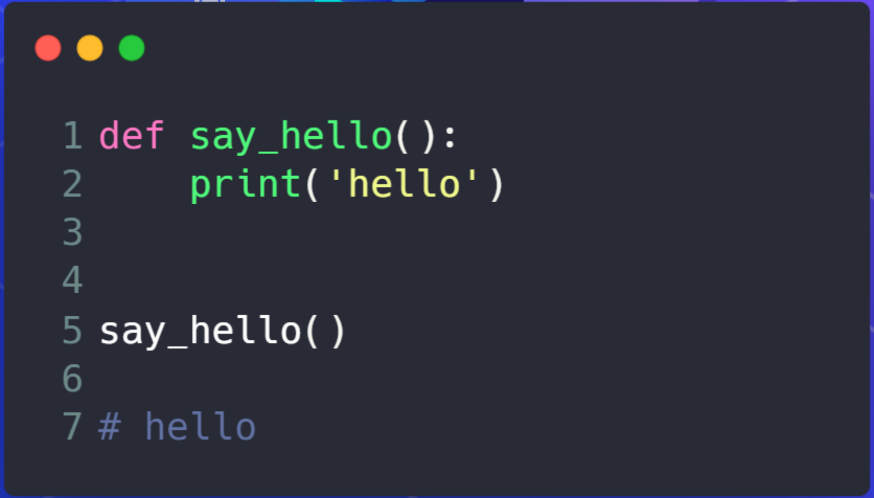
- Intro to Computer Science
- Environment Setup (Anaconda)
- Command Line
- Conda & pip package managers
- Jupyter Notebook
- Input & Output
- Variables
- Data types
- Numbers & Math
- Boolean & Comparison and Logic
- If Conditions
- For & While Loops
- Strings
- Lists
- Tuples
- Sets
- Dictionaries
- Advanced If and Loops
- List Comprehensions
- Dictionary Comprehensions
- Exceptions
- File Handling
- **Functions**
- Built-in functions & Operators (zip, enumerate, range, …)
- Map, Filter, Reduce
- Lambda Expressions

Functions

Functions are used to store a block of code to run later when needed

They become very handy when code needs to be used frequently, and it helps in encapsulation

In Python, we define a function and give it a name. When we need to use it, we 'call' it using it's name

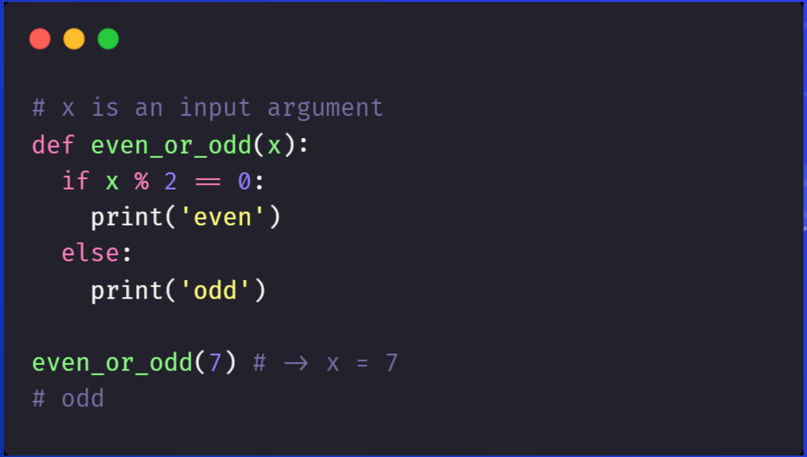


```python
def say_hello():
    print('hello')


say_hello()

# hello
```

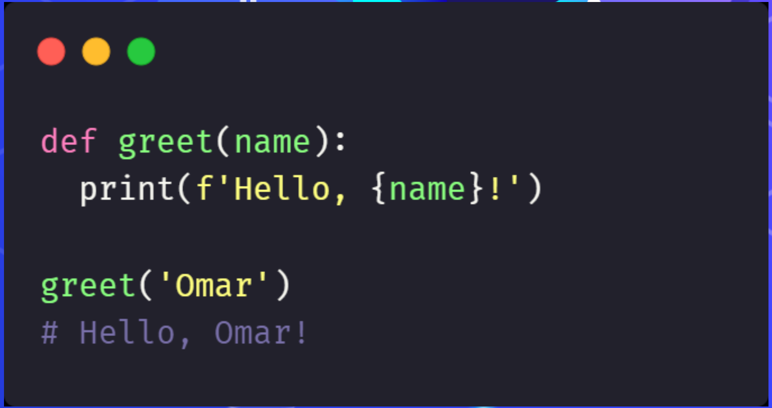
Function Arguments

Functions can have parameters that would be passed when the function is called

Those are called input arguments



```python
# x is an input argument
def even_or_odd(x):
  if x % 2 == 0:
    print('even')
  else:
    print('odd')

even_or_odd(7) # -> x = 7
# odd
```



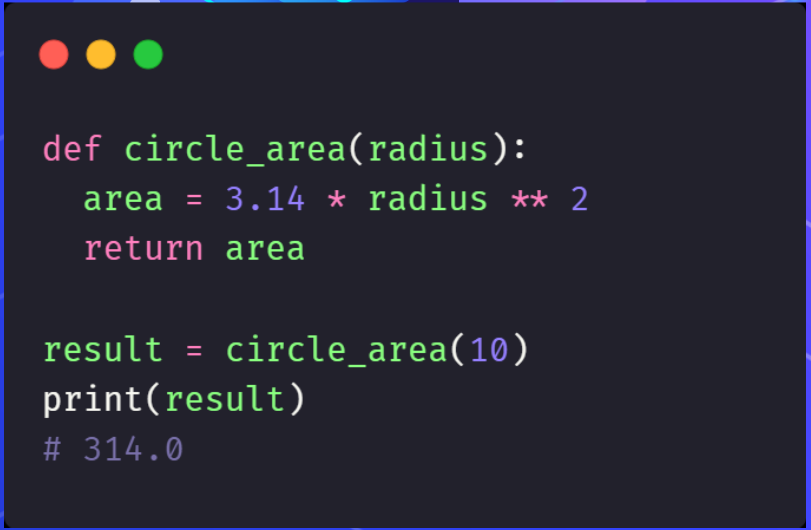
```python
def greet(name):
  print(f'Hello, {name}!')

greet('Omar')
# Hello, Omar!
```

Function Return

In many cases, functions can be used to perform a certain operation to calculate a value

We usually need this value for further use

We can use functions’ ‘return’ to return a value back to our program

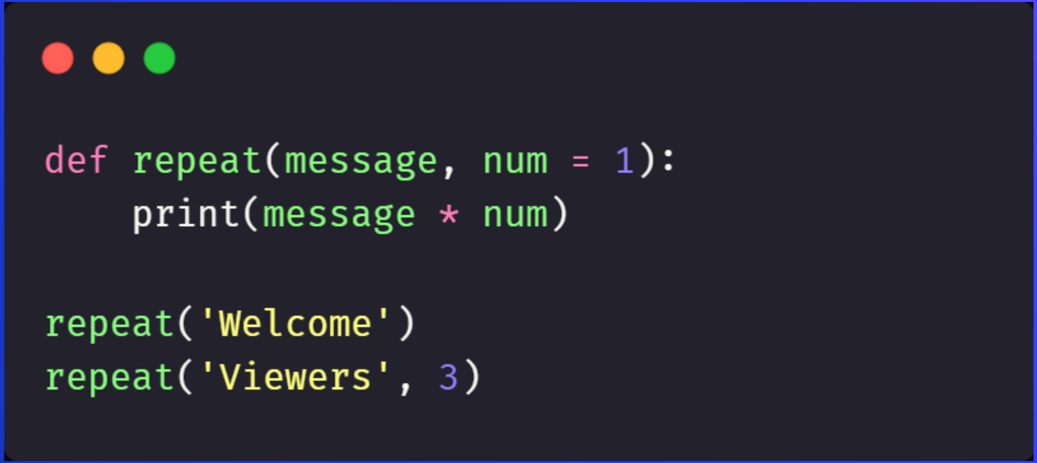


```python
def circle_area(radius):
  area = 3.14 * radius ** 2
  return area

result = circle_area(10)
print(result)
# 314.0
```

Quiz Time!

What will be the output of the following statement:

Q1
.



```
def repeat(message, num = 1):
    print(message * num)

repeat('Welcome')
repeat('Viewers', 3)
```

A. Welcome

Viewers

B. Welcome

ViewersViewersViewers

C. Welcome

Viewers,Viewers,Viewers

D. Welcome

Practice #2

- Create a function that prints True when the summation num1 and num2 is larger than 50 , otherwise prints False

Practice #3

- Write a python calculator function that simulates the four basic calculations .