Lecture # 7&8

CSC336 Web Technologies

Credit Hours: 3(2, 1)

Course Instructor: **SAIF ULLAH IJAZ**

Lecturer CS Dept, CUI Vehari

MSc University of Leicester, UK

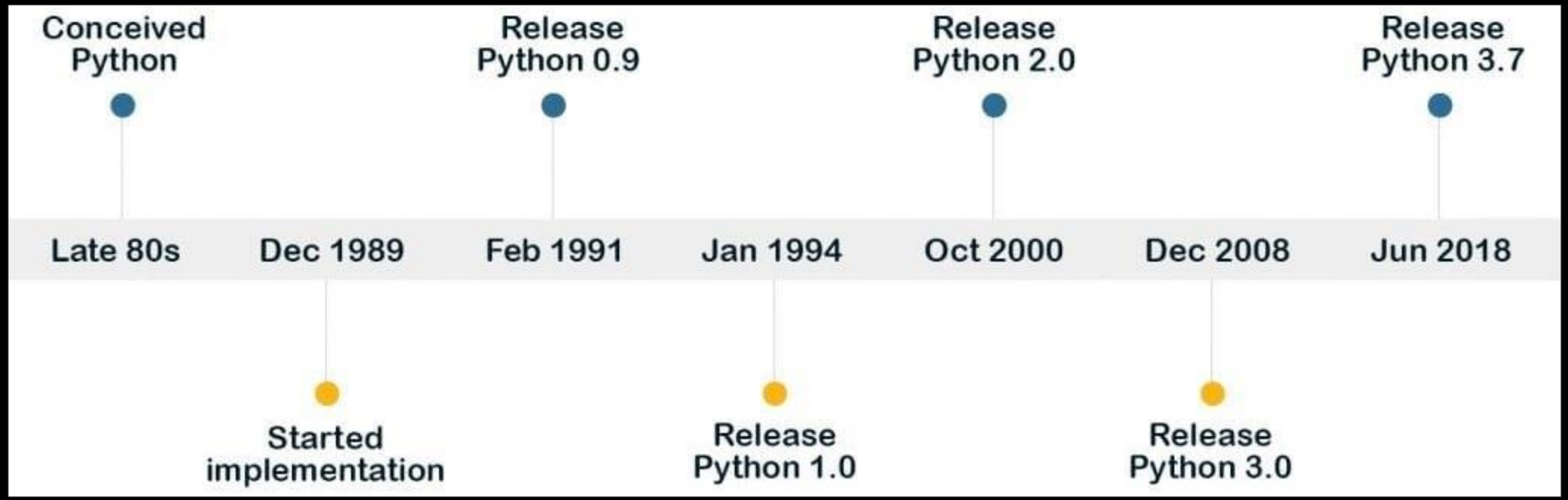
BSc COMSATS University Islamabad

Python

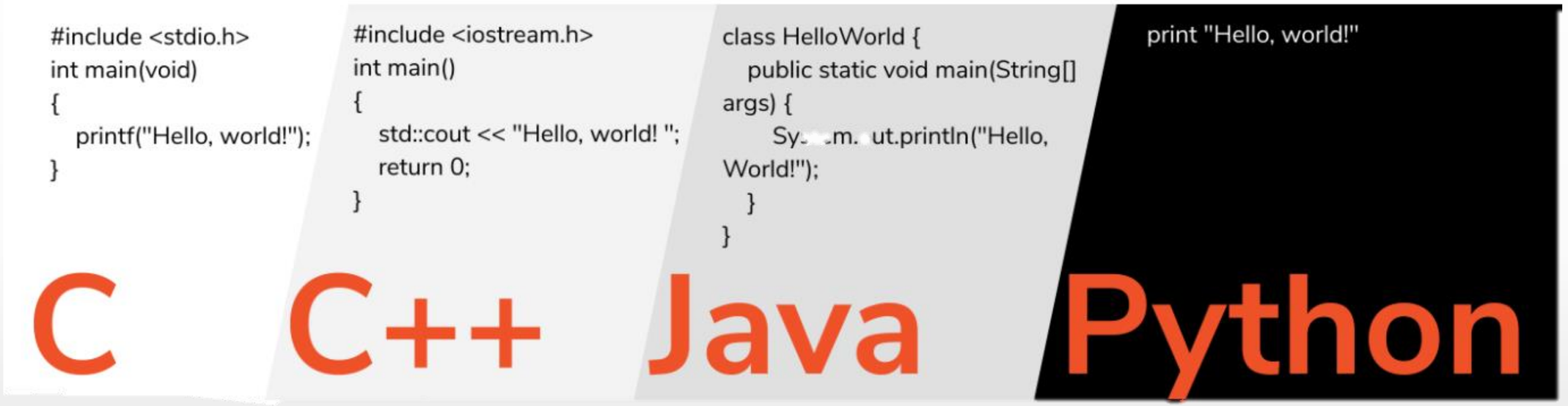
History

- Started by Guido Van Rossum as a hobby
- Now widely spread
- Open Source! Free!
- Versatile

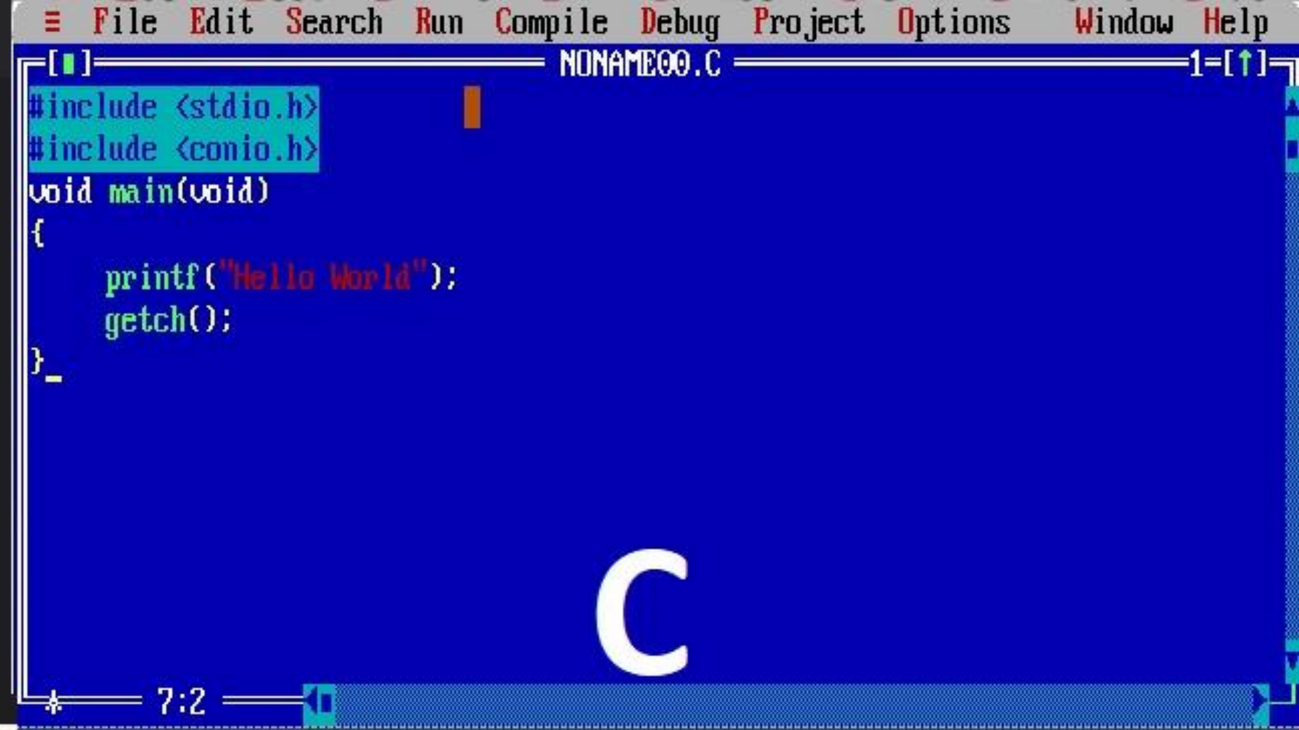
Guido Van Rossum by Doc Searls on Flickr CC-BY-SA



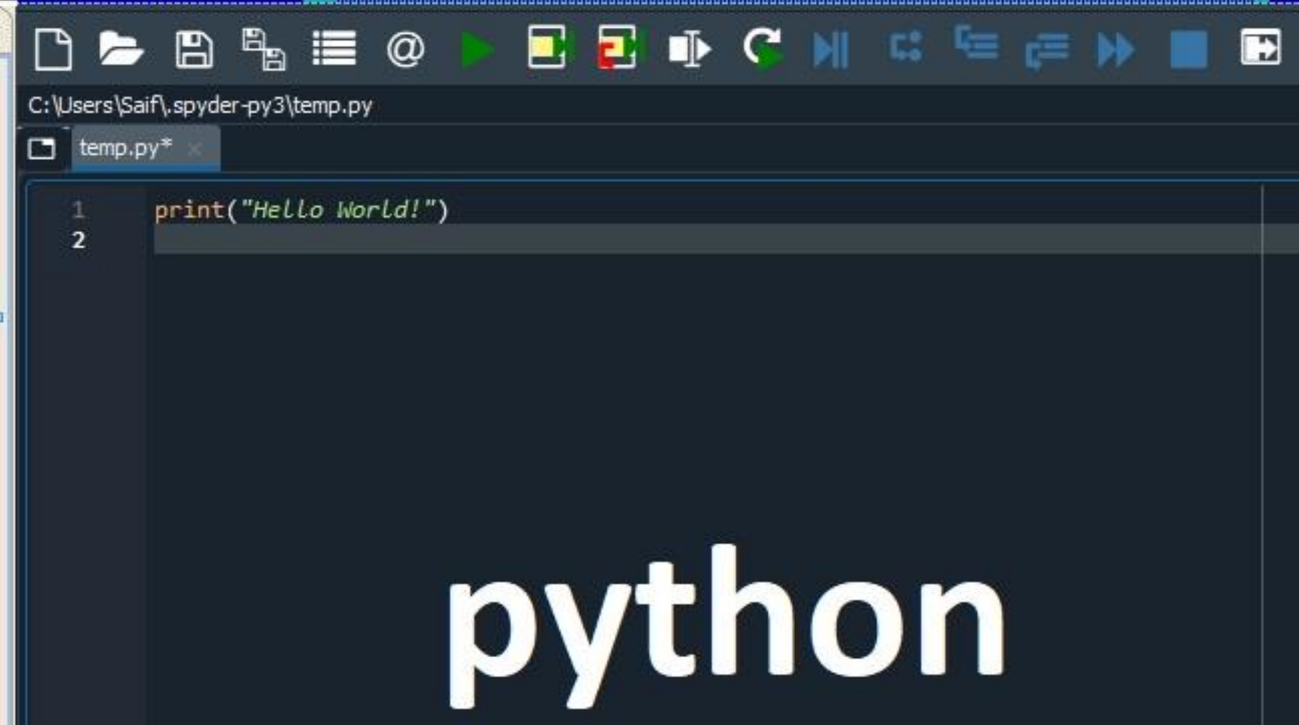
Print statement comparisons




EXPLORER
TRIPLEX
triplex.cpp
triplex.exe
triplex.obj
triplex.cpp
triplex.cpp > main()
#include <iostream>
int main()
{
std::cout << "Hello World!";
return 0;
}
C++


File Edit Search Run Compile Debug Project Options Window Help
NONAME00.C
#include <stdio.h>
#include <conio.h>
void main(void)
{
printf("Hello World");
getch();
}
7:2
C


*HelloWorld.java
public class HelloWorld {
/**
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub
System.out.println("Hello world!");
}
}
Java


C:\Users\Saif\.spyder-py3\temp.py
temp.py*
print("Hello World!")
python

Hello, World

```
print("Hello, world!")
```

Variables

```
a = 28
b = 1.5
c = "Hello!"
d = True
e = None
```

Types

```
a =  28            int
b =  1.5           float
c =  "Hello!"      str
d =  True          bool
e =  None          NoneType
```

Input

```
name = input("Name: ")
print(f"Hello, {name}")
```

name.py

```
name = input("Name: ")
print("Hello, " + name)

print(f"Hello, {name}")
```

Conditions

```
if x > 0:
    print("x is positive")
elif x < 0:
    print("x is negative")
else:
    print("x is 0")
```

conditions.py

```
n = input("Number: ")

if n > 0:
    print("n is positive.")
else:
    print ("n is not positive.")
```

conditions.py

```
n = int(input("Number: "))

if n > 0:
    print("n is positive.")
else:
    print ("n is not positive.")
```

conditions.py

```
n = int(input("Number: "))

if n > 0:
    print("n is positive.")
elif n < 0:
    print("n is negative.")
else:
    print("n is zero.")
```

sequences.py

```
name = "Saif"
print(name[0])

print(name[1])
```

sequences.py

```
names = ["Saif", "Rehan", "Haris"]
print(names)


print(names[0])
```

sequences.py

```
coordinateX = 10.0
coordinateY = 20.0

coordinate = (10.0, 20.0)
```

Data Structures

list - sequence of mutable values

tuple - sequence of immutable values

set - collection of unique values

dict - collection of key-value pairs

...

lists.py

```
# Define a list of names
names = ["Saif", "Rehan", "Haris", "Gohar"]

print(names)

print(names[0])
```

lists.py

```
# Define a list of names
names = ["Saif", "Rehan", "Haris", "Gohar"]

names.append("Daniyal")

names.sort()
print(names)
```

sets.py

```
# Create an empty set
s = set()

# Add elements to set
s.add(1)
s.add(2)
s.add(3)
s.add(4)
print(s)
```

sets.py

```
# Create an empty set
s = set()

# Add elements to set
s.add(1)
s.add(2)
s.add(3)
s.add(4)
s.add(3)   < ---
print(s)
```

sets.py

```
# Create an empty set
s = set()

# Add elements to set
s.add(1)
s.add(2)
s.add(3)
s.add(4)
s.add(3)
s.remove(2)  < ---
print(s)
```

sets.py

```
# Create an empty set
s = set()

# Add elements to set
s.add(1)
s.add(2)
s.add(3)
s.add(4)
s.add(3)
s.remove(2)
print(s)
print(f"The set has {len(s)} elements.")
```

loops.py

```
for i in [0, 1, 2, 3, 4, 5]:
    print(i)

for i in range(6):
    print(i)
```

loops.py

```
names = ["Saif", "Rehan", "Haris"]

for name in names:
    print(name)

name = "Saif"
for ch in name:
    print(ch)
```

dictionaries.py

```
cities = {"Saif": "Vehari", "Rehan":"Burewala"}

print(cities["Saif"])


cities["Haris"] = "Vehari"

print(cities["Haris"])
```

Lecture # 8

functions.py

```
def square(x):
    return x * x

for i in range(10):
    print(f"The square of {i} is {square(i)}.")
```

squares.py

```
for i in range(10):
    print(f"The square of {i} is {square(i)}.")
```

NameError: name 'square' is not defined

squares.py

```
from functions import square

for i in range(5):
    print(f"The square of {i} is {square(i)}.")
```

```
import functions

for i in range(5):
    print(f"The square of {i} is {functions.square(i)}.")
```

classes.py

```
class Point():
    def __init__(self, aX, aY):
        self.mX = aX
        self.mY = aY

p = Point(2, 8)
print(p.mX)
print(p.mY)
```

flight.py

```
class Flight():
    def __init__(self, aCapacity):
        self.mCapacity = aCapacity
        self.mPassengers = []

flight = Flight(3)
```

flight.py

```
class Flight():
    def __init__(self, aCapacity):
        self.mCapacity = aCapacity
        self.mPassengers = []

    def add_passenger(self, aName):
        self.mPassengers.append(aName)

flight = Flight(3)
```

flight.py

```
class Flight():
    def __init__(self, aCapacity):
        self.mCapacity = aCapacity
        self.mPassengers = []

    def add_passenger(self, aName):
        if not self.open_seats():
            return False
        self.mPassengers.append(aName)
        return True

    def open_seats(self):
        return self.mCapacity - len(self.mPassengers)
```

flight.py

```
flight = Flight(3)

people = ["Saif", "Rehan", "Haris", "Gohar"]

for person in people:
    success = flight.add_passenger(person)
    if success:
        print(f"Added {person} to flight successfully.")
    else:
        print(f"No available seats for {person}.")
```

decorators.py

```
def announce(f):
    def wrapper():
        print("About to run the function...")
        f()
        print("Done with the function.")
    return wrapper

@announce
def hello():
    print("Hello, World!")

hello()
```

lambda.py

```
people = [
    {"name": "Saif", "city":"Vehari"},
    {"name": "Rehan", "city":"Burewala"},
    {"name": "Haris", "city":"Lahore"}
]

people.sort()
print(people)
```

TypeError: '<' not supported between instances of 'dict' and 'dict'

lambda.py

```
people = [
    {"name": "Saif", "city":"Vehari"},
    {"name": "Rehan", "city":"Burewala"},
    {"name": "Haris", "city":"Lahore"}
]

def f(person):
    return person["name"]

people.sort(key=f)
print(people)
```

lambda.py

```
people = [
    {"name": "Saif", "city":"Vehari"},
    {"name": "Rehan", "city":"Burewala"},
    {"name": "Haris", "city":"Lahore"}
]

def f(person):
    return person[“city"]    < ---

people.sort(key=f)
print(people)
```

lambda.py

```
people = [
    {"name": "Saif", "city":"Vehari"},
    {"name": "Rehan", "city":"Burewala"},
    {"name": "Haris", "city":"Lahore"}
]

people.sort(key= lambda person:person["name"])
print(people)
```

exceptions.py

```
x = int(input("x: "))
y = int(input("y: "))

result = x / y
print(f"{x} / {y} = {result}")
```

exceptions.py

```
import sys
x = int(input("x: "))
y = int(input("y: "))

try:
    result = x / y
except ZeroDivisionError:
    print("Error: Can't divide by 0.")
    sys.exit(1)

print(f"{x} / {y} = {result}")
```

exceptions.py

```
import sys
try:
    x = int(input("x: "))
    y = int(input("y: "))
except ValueError:
    print("Invalid input")
    sys.exit(1)
```

Q & A