# Python Introduction

February 19, 2025

## 1 Python

### 1.1 Basic Syntax

#### 1.1.1 Comments

[ ]:
```
# This is a single line comment

'''
This is a
a multi line comment
'''
```

#### 1.1.2 Variable

[5]:
```
x= 10
name = "Ashish"
```

#### 1.1.3 print statement

[6]:
```
print("hello, Ashish!")
```

```
hello, Ashish!
```

#### 1.1.4 Indentation

[10]:
```
if x > 5:
    print("x is greater than 5")
```

```
x is greater than 5
```

### 1.2 Data Types

#### 1.2.1 Numbers

[11]:
```
int_num = 10          # integer
float_num = 10.5      # floating-point
complex_num = 1 + 2j # complex number
```

### 1.2.2 Strings

[12]:
```
single_quote_str = 'hello'
double_quote_str = "World"
multi_line_str = ''' This is
a multi line String'''
```

### 1.2.3 Lists

[13]:
```
fruits = ["apple", "banana", "cherry"]
```

### 1.2.4 Tuples

[14]:
```
point = (10, 20)
```

### 1.2.5 Dictionaries

[15]:
```
student = {"name": "Ashish", "age": 32, "courses": ["Biology", "Mathematics",␣
↪"Computer science"]}
```

### 1.2.6 Sets

[16]:
```
unique_numbers = {1,2,3,4,4,5}
```

### 1.2.7 Booleans

[17]:
```
is_valid = True
has_access = False
```

## 1.3 Example of basic py code

[19]:
```
name = "Ashish"
age = 32
courses = ["Biology", "Mathematics", "Computer science"]
print(f"{name} is {age} years old and takes {courses[0]}.")
```

```
Ashish is 32 years old and takes Biology.
```

## 1.4 Mutable vs Immutable objects

### 1.4.1 Mutable

Mutable objects can be changed after they are created. This means you can modify their content, such as adding, removing, or changing elements. Here are some common mutable data types: Lists, Dictionaries, Sets

**Lists**

[20]:
```
fruits = ["apple", "banana", "cherry"]
fruits[0] = "orange"  # Change element
```

```
fruits.append("grape")  # Add element
print(fruits)  # Output: ['orange', 'banana', 'cherry', 'grape']
```

```
['orange', 'banana', 'cherry', 'grape']
```

**Dictionaries**

[22]:
```
student = {"name": "Ashish", "age": 32}
student["age"] = 33 # change value
student["courses"] = ["Math", "Science"] # Add key-value pair
print(student) # Output: {'name': 'Ashish', 'age': 33, 'courses': ['Math',␣
 ↪'Science']}
```

```
{'name': 'Ashish', 'age': 33, 'courses': ['Math', 'Science']}
```

**Sets**

[24]:
```
unique_num = {1,2,3}
unique_num.add(4) # Add element
unique_num.remove(2) # Remove element
print(unique_num) # Output: {1,3,4}
```

```
{1, 3, 4}
```

### 1.4.2 Immutable

Immutable objects cannot be changed after they are created. Any attempt to modify them will result in the creation of a new object. Here are some common immutable data types: Strings, Tuples, Numbers

**Strings**

[39]:
```
greeting = "hello"

# greeting.replace("ello", "i") # This will not work because we need to assign␣
 ↪this step to a new variable like below

greeting = greeting.replace("ello", "i") # This will replace "ello" with "i"␣
 ↪and assign the value to greeting variable
print(greeting)
```

```
hi
```

**Tuples**

[30]:
```
point = (10,20)

#point[0] = 15 # This will raise an error

new_point = (15,20) # create a new tuple
print(point) #output: (10,20)
print(new_point) #output: (15,20)
```

```
(10, 20)
(15, 20)
```

**Numbers**

[41]:
```
x = 10
y = x + 5

print(x)
print(y)
```

```
10
15
```

## 1.5 Control Flow

### 1.5.1 Conditional Statements

**if statement**

[42]:
```
x = 10
if x > 5:
    print("x is greater than 5")
```

```
x is greater than 5
```

**if-else statement**

[43]:
```
x = 3
if x > 5:
    print("x is greater than 5")
else:
    print("x is lesser than or equal to 5")
```

```
x is lesser than or equal to 5
```

**elif statement**

[44]:
```
x = 5
if x > 5:
    print("x is greater than 5")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is less than 5")
```

```
x is equal to 5
```

### 1.5.2 Loop statement

**for loop**

[45]:
```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

**while loop**

[48]:
```
x = 0
while x < 5:
    print(x)
    x += 1
```

```
0
1
2
3
4
```

### 1.5.3 Loop control statements

**break statement**

[57]:
```
for i in range(5):
    if i == 3:
        break  # Exits the loop prematurely
    print(i)
```

```
0
1
2
```

**continue statement**

[58]:
```
for i in range(5):
    if i == 3:
        continue  # Skips the current iteration and proceeds to the next␣
 ↪iteration
    print(i)
```

```
0
1
2
4
```

**pass statement**

[59]:
```
for i in range(5):
    if i == 2:
        pass  # Does nothing and is used as a placeholder
    print(i)
```

```
0
1
2
3
4
```

## 1.6 Functions

### 1.6.1 Defining and calling functions

[61]:
```python
#p_name = "Ashish"

def greet(p_name):
    print(f"hello, {p_name}!")

greet("Ashish")
```

```
hello, Ashish!
```

[ ]:
```python

```