Python Basic

Topics Covered

1- Values, Expressions, and Statements

- Numbers & Operators
- Booleans
- Variable Assignment
- Printing
- Strings
- Comparison Operators
- Logic Operators

2- Data Structures

- Lists
- Tuples
- Dictionaries
- Sets

3- Conditional Execution

- if,elif, else Statements
- Logical Conditions
- Logical AND and OR
- Nested if

4- Iterations

- while Loops
- for Loops
- range()
- break and continue
- pass

5- user input

6- type casting

7- Function

- variable scope

8- File Handling

- OS

9- Exception Handling

1- Values Expressions Statements

Numbers & Operators

```
1 + 1
```

```
2
```

```
1 * 3
```

```
3
```

```
1 / 2
```

```
0.5
```

```
# power / exponent
2 ** 4
```

```
16
```

```
# modulus
4 % 2
```

```
0
```

```
5 % 2
```

```
1
```

```
# precedence
(2 + 3) * (5 + 5)
```

```
50
```

```
2 + 3 * 5 + 5
```

```
22
```

Booleans

```
True
```

```
True
```

```
False
```

```
False
```

Variable Assignment

```
# Can not start with number or special characters
name_of_var = 2
```

```
# sdksk
```

```
x = 2
y = 3

z = x + y

z

5

s = 12
```

Printing

```
x = 5
print(x)

5
```

Strings

```
'single quotes'

'single quotes'

"double quotes"

'double quotes'

" wrap lot's of single quotes in doyuble quotes"

' wrap lot's of single quotes in doyuble quotes '

  File "<ipython-input-19-b116179b21d7>", line 1
    ' wrap lot's of single quotes in doyuble quotes '
               ^
SyntaxError: invalid syntax


# .format()
num =124563456
name='ali'
x = 'my name is {} and my number is {}'.format(name,num)
x

'my name is ali and my number is 124563456'

# find type of any object
type(x)

str

s = 'hello world'

# Capitalize first word in string
s.capitalize()
```

```
'Hello world'
```

```
# uppercase
s.upper()
```

```
'HELLO WORLD'
```

```
# lowercase
s.lower()
```

```
'hello world'
```

```
# count a letter in a string
s.count('o')
```

```
2
```

```
# check if string is alpha numeric
s.isalnum()
```

```
False
```

```
# check if string is numeric
s.isnumeric()
```

```
False
```

```
# String multiline
s = """hello
m
ert
l
e"""
print(s)
```

```
hello
m
ert
l
e
```

```
# Break string in code only
multiline = "this is a \
ultiline string"
multiline
```

```
'this is a ultiline string'
```

```
# String slicing
s='hello'
s[0]
```

```
'h'
```

```
s[1:3]
```

```
'el'
```

```
s[:]
```

```
'hello'
```

```
# slicing using -ve numbers for indices
s[:-1]
```

```
'hell'
```

```
s[-3:]
```

```
'llo'
```

```
# Split and Partition
s = "This is a course on machine Learning."
s.split(" ")
```

```
['This', 'is', 'a', 'course', 'on', 'machine', 'Learning.']
```

```
# split on a word or letter
s.split("machine")
```

```
['This is a course on ', ' Learning.']
```

```
# partition on a word or letter
s.partition("machine")
```

```
('This is a course on ', 'machine', ' Learning.')
```

```
# f string
course_name = "machine learning"
duration = 5
s = f"This is a course on {course_name} and its duration is {duration}
days."
s
```

```
'This is a course on machine learning and its duration is 5 days.'
```

```
# is in
print("course" in s)
# not in
print("k" in s)
```

```
True
False
```

Comparison Operators

```
1 > 2
```

```
False
```

```
1 < 2
```

```
True
```

```
1 >= 1
```

```
True
```

```
1 <= 4
```

```
True
```

```
1 == 1
```

```
True
```

```
'hi' == 'bye'
```

```
False
```

```
'hi' != 'bye'
```

```
True
```

Logic Operators

```
# and
(1 > 2) and (2 < 3)
```

```
False
```

```
# or
(1 > 2) or (2 < 3)
```

```
True
```

```
# or
(1 == 2) or (2 == 3) or (4 == 4)
```

```
True
```

2- Data Structures

Data structures are formations in which data can be kept while in memory. Python offers multiple built-in data structures, so that the programmer needs not to implement these from scratch. These include:

- List
- Tuple
- Dictionary
- Set

List

```
# list creation
my_list = [1,2,3]
my_list
```

```
[1, 2, 3]
```

```
# list append
my_list.append('d')
my_list
```

```
[1, 2, 3, 'd']
```

```
# List indexing
my_list[3]
```

```
'd'
```

```
# list slicing
my_list[1:]
```

```
# change/insert a new item in a list on a particular index position
my_list[0] = 'NEW'
my_list
```

```
['NEW', 2, 3, 'd']
```

```
# nested list
nest = [1,2,3,[4,5,['target']]]
```

```
# indexing of a nested list
nest[3]
```

```
[4, 5, ['target']]
```

```
nest[3][2]
```

```
['target']
```

```
nest[3][2][0]
```

```
'target'
```

```
# list of list
l_o_l = [[[5],[3],[15]],[3,4,5],[6,7,8]]
l_o_l
```

```
[[[5], [3], [15]], [3, 4, 5], [6, 7, 8]]
```

```
# size of a list
len(l_o_l)
```

```
3
```

```
# pop an element
a  = l_o_l.pop()
l_o_l
```

```
[]
```

```
# count a particular item in a list
my_list = ['abc', 23, 23, 23]
my_list.count(23)
```

```
3
```

```
# merging two lists
list_1 = [1,2,3,4,5]
list_2 = [5,6,7,8,9]
list_1+list_2
```

```
[1, 2, 3, 4, 5, 5, 6, 7, 8, 9]
```

```
list_1 = [[1],[2,34,99],3,4,5]
len(list_1[1])
```

```
3
```

Tuples

```
# tuple appearance
t = (1,2,3)
```

```
# tuple indexing
t[0]
```

```
1
```

```
# tuple slicing
t[1:]
```

```
(2, 3)
```

```
# tuples are immutable objects so no item can be changed
t[0] = 'NEW'
```

```
----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-89-031d14c6561a> in <module>
      1 # tuples are immutable objects so no item can be changed
----> 2 t[0] = 'NEW'

TypeError: 'tuple' object does not support item assignment
```

```
# tuple merging
t = (1,2,3)
```

```
u = (3,5,6)
v = t+u
print(v)
```

```
(1, 2, 3, 3, 5, 6)
```

```
u
```

```
(3, 5, 6)
```

Dictionaries

```
d = {'key1':'item1','key2':'item2'}
d
```

```
{'key1': 'item1', 'key2': 'item2'}
```

```
# item accessing
d['key1']
```

```
'item1'
```

```
# item replacing
d['key1'] = 5
d
```

```
{'key1': 5, 'key2': 'item2'}
```

```
#Dictionary Nested with other dictionary
d={'k1':{'innerkey':[1,2,3]}}
print(d['k1'])
print(d['k1']['innerkey'])
```

```
{'innerkey': [1, 2, 3]}
[1, 2, 3]
```

```
d['k1']['innerkey'][1]
```

```
2
```

```
pets = {'dogs':5, 'cats':4}
```

```
pets.keys5()
```

```
dict_keys(['dogs', 'cats'])
```

```
pets.items()
```

```
dict_items([('dogs', 5), ('cats', 4)])
```

```
pets.values()
```

```
dict_values([5, 4])
```

```
# deleting all key-value pairs in a dictionary
pets.clear()
pets
```

```
{}
```

```
# check if a certain key or value is in the dictionary
"cats" in pets.keys()
```

```
False
```

Sets

```
st = {1,2,3}
```

```
# sets are collections of unique items
{1,2,3,1,2,1,2,3,3,3,3,2,2,2,1,1,2}
```

```
{1, 2, 3}
```

```
# set item add
st.add(9)
```

```
# set allow intersection, difference, union among other operations
s1 = {1,2,3}
s2 = {3,4,5}
intersection = s1.intersection(s2)
print("intersection is:", intersection)
union = s1.union(s2)
print("union is:", union)
difference = s1.difference(s2)
print("difference is:", difference)
```

```
intersection is: {3}
union is: {1, 2, 3, 4, 5}
difference is: {1, 2}
```

```
# Ashort hand for
a={1,3,4,6,'g'}
b = {2, 3, 4, 5}

print("Intersection: ",a & b)
print("Union: ",a | b)
```

```
Intersection:  {3, 4}
Union:  {'g', 1, 2, 3, 4, 5, 6}
```

3- Conditional Execution

if,elif, else Statements

```
# if
if 1 < 2:
    print('Yep!')
```

```
Yep!
```

```
if 1 < 2:
    print('yep!')
```

```
yep!
```

```
# if - else
if 1 < 2:
    print('first')
else:
    print('last')
```

```
first
```

```
# if -elif - else
if 1 == 2:
    print('first')
elif 3 == 5:
    print('middle')
else:
    print('Last')
```

```
Last
```

Logical Conditions

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

Logical AND, OR

- and
- or

```
a = 10
b = 30
c = 60

if a < 30 and c > b:
    print ("yes")
else:
    print ("No")


if a < 30 or c > 100:
    print ("yes")
```

```
else:
    print ("No")
```

```
yes
yes
```

```
# Check the extras the customer ordered
dietCoke = False
shake = True
fries = True
burger = True

# Evaluate the customer's order
if (dietCoke or shake) and (fries or burger):
    print("The customer wants an extra drink " +
          "(diet coke and/or shake) and extra food " +
          "(french fries and/or burger).")
else:
    print("The customer doesn't want both an " +
          "extra drink *and* extra food.")
```

```
The customer wants an extra drink (diet coke and/or shake) and extra
food (french fries and/or burger).
```

Nested if

```
a = [1,2,3,4,5,6,7]
b = [1,2,3]
c = [1]

if a[0] in b:
    print("first element is also in the second list")
    if a[0] in c:
        print("first element is also in the third list")
```

```
first element is also in the second list
first element is also in the third list
```

4- Iterations/Loops

while Loop

```
i = 0
while i < 5:
    print('i is: {}'.format(i))
    i = i+1
```

```
i is: 0
i is: 1
i is: 2
```

```
i is: 3
i is: 4
```

for Loop

```
seq = [1,2,3,4,5]
```

```
for item in seq:
    print(item)
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
~\AppData\Local\Temp/ipykernel_752/1443112677.py in <module>
----> 1 for item in seq:
      2     print(item)

NameError: name 'seq' is not defined
```

```
for jelly in seq:
    print(jelly+jelly)
```

```
2
4
6
8
10
```

range()

```
range(5)
```

```
range(0, 5)
```

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

```
list(range(5))
```

```
[0, 1, 2, 3, 4]
```

Break and Continue

```
# break
for x in seq:
  if x == 3:
    break
  print(x)
```

```
1
2
```

```
# continue
for x in seq:
    if x == 3:
        continue
    print(x)
```

```
1
2
4
5
```

Pass

```
# cannot left loops, if statements and functions empty. if wanted to
do so use pass
for x in [0, 1, 2]:
    pass
```

5- User Input

```
name = input("Enter your name")
print(name, type(name))
age = input("Enter your age")
age = int(age)
print(type(age))
```

```
Enter your nameab
ab <class 'str'>
Enter your age1
<class 'int'>
```

```
flag = True
while flag:
    inp = input('Enter Fahrenheit Temperature:')
    try:
        fahr = float(inp)
        flag = False
        cel = (fahr - 32.0) * 5.0 / 9.0
        print (cel)
    except:
        print ('Please enter a number')
```

```
Enter Fahrenheit Temperature:44
6.666666666666667
```

6- Type Casting

```
type_1 = "1"
type_2 = 2
print(type_1 + type_2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call 
last)
~\AppData\Local\Temp/ipykernel_752/3777246601.py in <module>
      1 type_1 = "1"
      2 type_2 = 2
----> 3 print(type_1 + type_2)

TypeError: can only concatenate str (not "int") to str
```

```
print(type_1 + str(type_2))
```

```
12
```

7- Functions

```
def my_func(param1='default_parameter'):
    """
    Docstring goes here.
    """
    print(param1)
```

```
my_func
```

```
<function __main__.my_func(param1='default_parameter')>
```

```
my_func()
```

```
default_parameter
```

```
my_func('new param')
```

```
my_func(param1='new param')
```

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

```
out = square(5)
```

```
print(out)
```

```
25
```

8- File Handling

```
# file writing
file = open('data.txt', 'w')
file.write('1 2 3 4\n')
file.write('2 3 4 5\n')
file.write('6 7 8 9\n')
file.write('10 11 12 13\n')
file.write('14 15 16 17\n')
file.close()

# File handling modes:

#     “ r “, for reading.
#    “ w “, for writing.
#    “ a “, for appending.
#    “ r+ “, for both reading and writing.
#    “ a+ “, for both reading and appending.

# For binary file: rb, wb

file = open('data.txt')
data = []
for line in file:
    data.append([int(field) for field in line.split()])
print(data)
file.close()
```

```
[[1, 2, 3, 4], [2, 3, 4, 5], [6, 7, 8, 9], [10, 11, 12, 13], [14, 15,
16, 17]]
```

```
# with open
with open ('data.txt', 'r') as fy:
    for line in fy:
        print(line)
```

```
1 2 3 4

2 3 4 5

6 7 8 9

10 11 12 13

14 15 16 17
```

OS

```
import os

all_files_in_dir = []
current_dir = os.getcwd()
filenames  = os.listdir()
file_path = []

print(current_dir,"\n", filenames)

for file in filenames:
    if file.endswith('.ipynb'):
        file_path.append(os.path.join(current_dir,file))


print(f"------------\nFiles are{file_path}")
print("TOTAL JUPYTER FILES IN CURRENT DIRECTORY: ",len(file_path))
```

```
C:\Users\Administrator\Desktop\Training\Day1\Slot 2
 ['.ipynb_checkpoints', 'data.txt', 'Python Introduction.ipynb']
------------
Files are['C:\\Users\\Administrator\\Desktop\\Training\\Day1\\Slot 2\\
Python Introduction.ipynb']
TOTAL JUPYTER FILES IN CURRENT DIRECTORY:  1
```

```
# same code with list comprehension
files_paths = [os.path.join(os.getcwd(),x) for x in
os.listdir(os.getcwd()) if x.endswith('.ipynb')]
print(len(files_paths))
```

```
1
```

```
files_paths
```

```
['C:\\Users\\Administrator\\Desktop\\Training\\Day1\\Slot 2\\Python
Introduction.ipynb']
```

```
with open(file_path[0], 'r') as f:
    print(f.read())
```

9- Exception Handling

An exception is a Python object that represents an error. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception.

For further details on this topic please see this:
https://www.tutorialspoint.com/python/python_exceptions.htm

```
try:
    with open("abc", "r") as f:
        for line in f:
            print(line)
except Exception as e:
    print(e)

finally:
    print("The file opening task was run")
```

```
[Errno 2] No such file or directory: 'abc'
The file opening task was run
```