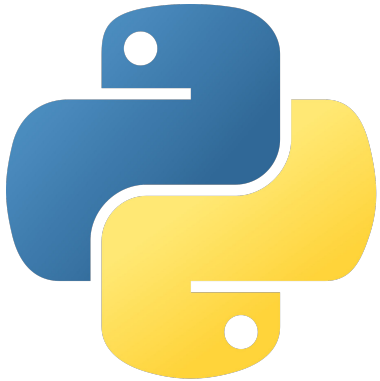
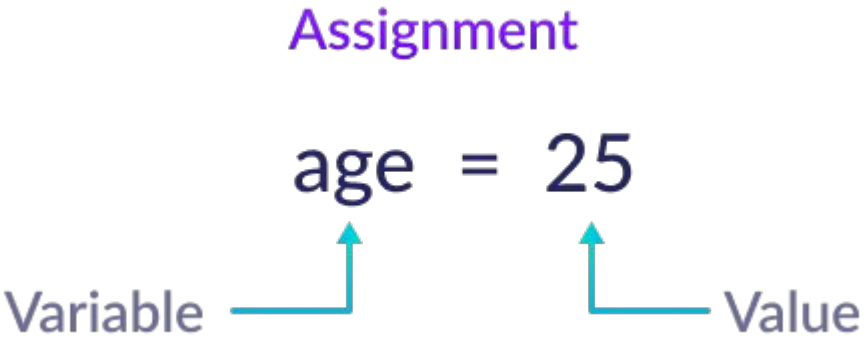
Python Programming Language



Review

Create Variable



Data Type

Type	Example
String	“สวัสดี”
Integer	1
Float	1.00
Binary	0010
List	[1,2,3]
Tuple	(1,2,3)
Dict	{‘name’, ‘Noey’, ‘age’: ‘24’}
Boolean	True, False

Check Data Type

Type of function

```
a = 42
b = 3.14
c = "Hello, World!"
d = [1, 2, 3]
e = (1, 2, 3)
f = {"name": "John", "age": 30}

print(type(a))  # Output: <class 'int'>
print(type(b))  # Output: <class 'float'>
print(type(c))  # Output: <class 'str'>
print(type(d))  # Output: <class 'list'>
print(type(e))  # Output: <class 'tuple'>
print(type(f))  # Output: <class 'dict'>
```

Is instance function

```
a = 42
b = 3.14
c = "Hello, World!"
d = [1, 2, 3]
e = (1, 2, 3)
f = {"name": "John", "age": 30}

print(isinstance(a, int))      # Output: True
print(isinstance(b, float))    # Output: True
print(isinstance(c, str))      # Output: True
print(isinstance(d, list))     # Output: True
print(isinstance(e, tuple))    # Output: True
print(isinstance(f, dict))     # Output: True
print(isinstance(a, (int, str))) # Output: True
```

Control Flow Structure

If-Else Statement

```
a = 200
b = 33

if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Result

```
a is greater than b
```

Control Flow Structure

Loops Statement

```
for x in "banana":
  print(x)
```

Result

```
b
a
n
a
n
a
```

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Result

```
1
2
3
4
5
```

Control Flow Structure

Break, Continue Loops

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

Result

```
1
2
3
```

Break, Continue Loops

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

Result

```
1
2
4
5
6
```

Control Flow Structure

Pass

```
for x in [0, 1, 2]:
  pass

def myfunction():
  pass

class Person:
  pass
```

```
a = 33
b = 200

if b > a:
  pass
```

Result

Module and Package

Module

In Python, Modules are simply files with the ".py" extension containing Python code that can be imported inside another Python Modules Operations Program.

file mymodule.py

```
def welcome(name):
  print("Hello, " + name +" to Analytics Vidhya")
```

file main.py

```
import mymodule
mymodule.welcome("Chirag Goyal")
```

Package

Python Packages are a way to organize and structure your Python code into reusable components. Think of it like a folder that contains related Python files (modules) that work together to provide certain functionality.

```
mypackage/
│
├──   __init__.py
├── mymodule.py
```

```
from mypackage import mymodule

result = mymodule.welcome(3, 5)
```

File Management

Open File

```
f = open("demofile.txt")


f = open("demofile.txt", "rt")
```

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

File Management

demofile.txt

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

Read All

```
f = open("demofile.txt", "r")
print(f.read())
```

Result

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

File Management

Read Some Part

```
f = open("demofile.txt", "r")
print(f.read(5))
```

Result

```
Hello
```

Read Some Line

```
f = open("demofile.txt", "r")
print(f.readline())
```

Result

```
Hello! Welcome to demofile.txt
```

```
//////////////////////////////////////////
f = open("demofile.txt", "r")
print(f.readline())
print(f.readline())
```

Result

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
```

File Management

Read Use Loops

```
f = open("demofile.txt", "r")
for x in f:
  print(x)
```

Result

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

Close Files

It is a good practice to always close the file when you are done with it.

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

File Management

Write File

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
```

Result

```
f = open("demofile2.txt", "r")
print(f.read())
```

```
Hello! Welcome to demofile2.txt
This file is for testing purposes.
Good Luck!Now the file has more content!
```

Open the file "demofile3.txt" and overwrite the content

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()

#open and read the file after the
overwriting:
f = open("demofile3.txt", "r")
print(f.read())
```

Result

```
Woops! I have deleted the content!
```

File Management

Create a New File

To create a new file in Python, use the open() method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

Delete a File

To delete a file, you must import the OS module, and run its os.remove() function:

```
import os
os.remove("demofile.txt")
```

```
//////////////////////////////////////////
```

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

File Management

Exception Handling

```
try:
  print(x)
except:
  print("An exception occurred")


//////////////////////////////////////////

try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("Something else went wrong")
```

```
print(x)
```

Result

```
Traceback (most recent call last):
  File "demo try except_error.py",
line 3, in <module>
    print(x)
NameError: name 'x' is not defined
```