PYTHON PROGRAMMING

PICT IEEE STUDENT BRANCH

Contents

Exception Handling

Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

Try and except statements are used to catch and handle exceptions in Python. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause.

```
# Syntax for exception handling

try:
    '''
    Code to be checked is written in this block.
    '''
except:
    '''
    Exception to be thrown is mentioned in this block.
    '''
```

Example of exception handling

```
# import module sys to get the type of exception
import sys

randomList = ['a', 0, 2]

for entry in randomList:
    try:
        print("The entry is", entry)
        r = 1/int(entry)
        break
    except:
        print("Oops!", sys.exc_info()[0], "occurred.")
        print("Next entry.")
        print()
print("The reciprocal of", entry, "is", r)
```

```
#Syntax for exception handling if custom error is to be generated if the error is known

try:
    """Code block to be tested."""
except '''type of error ''':
    '''Error message to be thrown'''
except '''type of error ''':
    '''Error message to be thrown'''
```

Example:

```
def doSomething(x,y):
    try:
        z = ((x+y)/(x-y))
    except ZeroDivisionError:
        print ("x/y result in 0")
    else:
        print (z)
```

Use of "finally" keyword

Python provides a keyword finally, which is always executed after the try and except blocks. The final block always executes after normal termination of try block or after try block terminates due to some exception.

Syntax --

```
try:
    '''Code block to be tested'''
except:
    '''Exception to be thrown if there is any'''
else:
    '''If no exception is thrown'''
finally:
    '''After the block ends'''
```

Example of "finally" keyword

```
try:
    x = 10//0
    print(x)

except ZeroDivisionError:
    print("Division by 0 is not possible")

finally:
    print('Hello world')
```

Example of "raising" an error

```
try:
    raise NameError("Hello everyone")
except NameError:
    print ("Exception happened")
    raise  """To determine whether the exception was raised or not"""
```

Iterators and Generators

Iterators are containers for objects so that you can loop over the objects. In other words, you can run the "for" loop over the object.

Python generator gives us an easier way to create python iterators. This is done by defining a function but instead of the return statement returning from the function, use the "yield" keyword.

Few examples of Iterators and generators

Generator

```
# A generator
def new_generator():
    p = 1
    print('first')
    yield p

    p += 1
    print('second')
    yield p

    p += 1
    print('last')
    yield p
```

Iterator

```
# Do this in the python shell
>>> var1 = new_generator()

# We can iterate through the items using next().
>>> next(var1)
first
1
# Once the function yields, the function is paused and the control is transferred to the caller.

# Local variables and theirs states are remembered between successive calls.
>>> next(var1)
second
2

>>> next(var1)
last
3
```

Math and Random libraries

Python has a built-in module that you can use for mathematical tasks.
The math module has a set of methods and constants.

```
import math
```

Python has a built-in module that you can use to make random numbers.

```
import random
```

Important functions in math library

sin(x)	returns the sine of value passed as argument. The value passed in this function should be in radians sin(0.88) : 0.77 sin(0) : 0.0
radians(x)	Used to convert specified angle to radians.
floor(x)	Decreases value to the previous integer value floor(4.3) : 4.
pow(x,y)	Used to raise x value toy Example : pow(5,2) : 25
sqrt(x)	Used to get square root of the value passed as argument. sqrt(25) : 5
cos(x)	Returns the cos of value passed as argument. cos(-3) : -0.9899924966 cos(0) : 1.0
log(x,[base])	Used to get natural algorithm of x specified base. log(5.5,2) : 2.45943
factorail(x)	Used to get factorail of the value passed as argument. factorial(23): 2585201673
gcd(x,y)	Used to get common divisor of x and y math.gcd(60, 48) : 12

Important functions in random library

seed()	Initialize the random number generator
getstate()	Returns the current internal state of the random number generator
setstate()	Restores the internal state of the random number generator
getrandbits()	Returns a number representing the random bits
randrange()	Returns a random number between the given range
randint()	Returns a random number between the given range
choice()	Returns a random element from the given sequence
choices()	Returns a list with a random selection from the given sequence
shuffle()	Takes a sequence and returns the sequence in a random order
sample()	Returns a given sample of a sequence
random()	Returns a random float number between 0 and 1
uniform()	Returns a random float number between two given parameters
triangular()	Returns a random float number between two given parameters, you can also set a mode parameter to specify the midpoint between the two other parameters

Python short hand for if-else and for loops

```python
# Python shorthand for if-else blocks
x = int(input())
y = int(input())
print("X") if x > y else print("=") if x == y else print("B")

# List comprehension is done using short-hand notation for for
# loop in python
mylist = ["python", "javascript", "lua"]
[print(i) for i in mylist]
```

Functools and Itertools

With two Python modules, itertools and functools, we can generate elegant solutions. Learn a variety of the functions from itertools and see how it would accelerate coding! functools provides higher-order functions and operations on callable objects and itertools let us compose elegant solutions for a variety of problems with the functions it provides.

Source #1

Source #2

Words of Wisdom

Always try to solve as many as problems on every concept.

Never feel overwhelmed when coming across new libraries. Always remember that libraries are made to make a programmer's life easier and as long as you know the basics,you will be able to use them.

Official Documentation is your best friend!!