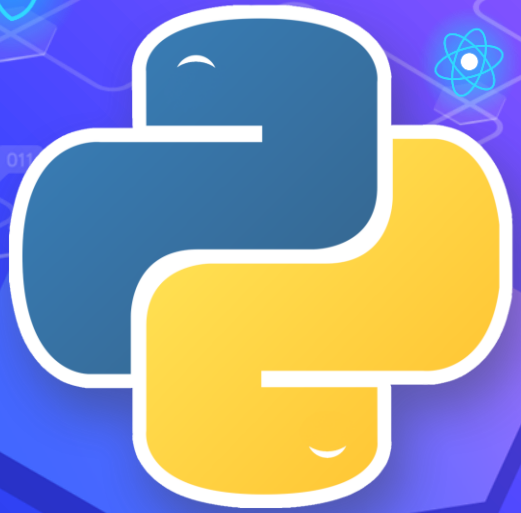
Introduction To Python Programming



Strings

Strings are ordered sequences of characters (alphabets, numbers, etc.)
Individual characters can be accessed using indexing

```
greeting = "Hello World"
greeting = 'Hello World'

print(greeting[0])  # H
print(greeting[2])  # l
print(greeting[-1]) # d
```

String Formatting

A way to inject a variable into a string for convenience

Add an ‘f’ before the string to add formatting,
then add variables using braces {}

```
x = 10
y = x / 2

print(f"Value of x = {x} and value of y = {y}")
# Value of x = 10 and value of y = 5.0
```

Introduction to Python programming Course Outline

- Intro to Computer Science
- Environment Setup (Anaconda)
- Command Line
- Conda & pip package managers
- Jupyter Notebook
- Input & Output
- Variables
- Data types
- Numbers & Math
- Boolean & Comparison and Logic
- If Conditions
- For & While Loops
- Strings
- Lists
- Tuples
- Sets
- Dictionaries
- **Advanced If and Loops**
- List Comprehensions
- Dictionary Comprehensions
- Exceptions
- File Handling
- Functions
- Built-in functions & Operators (zip, enumerate, range, …)
- Map, Filter, Reduce
- Lambda Expressions

Advanced for loop

For loop can be used to iterate over any iterable

Lists, tuples, strings, sets and dictionaries are all examples of Python iterables

```python
name = "Omar"

for ch in name:
  print(ch.upper())

  # O
  # M
  # A
  # R
```

```python
my_list = ['Apple', 'Orange', 'Banana']

for fruit in my_list:
  print(f"Fruit: {fruit}")

  # Fruit: Apple
  # Fruit: Orange
  # Fruit: Banana
```

Advanced for loop

Continue: skip the current iteration and go to the next one
Break: break out of the loop and end the loop

```python
# Print only odd numbers
for x in range(10):
  if x % 2 == 0:
    continue
  print("X:", x)

  # X: 1
  # X: 3
  # X: 5
  # X: 7
  # X: 9
```

```python
# Print numbers up to six
for x in range(10):
  if x == 6:
    break
  print("X:", x)

  # X: 0
  # X: 1
  # X: 2
  # X: 3
  # X: 4
  # X: 5
```

Advanced for loop

We can use ‘else’ with for loops, just like ‘if’

The code block in ‘else’ will only be executed if the loop finishes running normally. If a break happens, the ‘else’ block will not be executed

```python
names = ['Omar', 'Mohamed', 'Karim']

for name in names:
  if name == 'Hussien':
    print("Hello Hussien")
    break
else:
  # This will be executed
  print("I'm finished")

# I'm finished
```

```python
names = ['Omar', 'Mohamed', 'Karim']

for name in names:
  if name == 'Omar':
    # This will be executed
    print("Hello Omar")
    break
else:
  print("I'm finished")

# Hello Omar
```

Advanced while loop

Just like for loops, we can use break and continue using while loops too

```python
x = 0

#Print only even numbers
while x < 10:
  x += 1
  if x % 2 ≠ 0:
    continue
  print(f'X: {x}')

# X: 2
# X: 4
# X: 6
# X: 8
# X: 10
```

```python
x = 0

#Print numbers up to 6
while x < 10:
  x += 1
  if x == 6:
    break
  print(f'X: {x}')

# X: 1
# X: 2
# X: 3
# X: 4
# X: 5
```

Advanced while loop

‘else’ statement works on while too!

```
x = 0

#Print only even numbers
while x < 10:
  x += 1
  if x % 2 ≠ 0:
    continue
  print(f'X: {x}')
else:
  print("I'm finished")

# X: 2
# X: 4
# X: 6
# X: 8
# X: 10
# I'm finished
```

Quiz Time!

What will be the output of the following statements:

Q1

```
x = 0
a = 0
b = -5
if a > 0:
    if b < 0:
        x = x + 5
    elif a > 5:
        x = x + 4
    else:
        x = x + 3
else:
    x = x + 2
print(x)
```

- A. 2
- B. 5
- C. 3

Q2

```
for l in 'Jhon':
   if l == 'o':
      continue
   print(l)
else:
  print("It's John!")
```

- A. J, h, o, n, It’s John!
- B. J, h, n, It’s John!
- C. J, h, n

Practice #5

- Write a Python program that choose only integers from the following mixed list using continue :
- x = [1, so, ‘2’, too, 3, but, 4, soon, 5, every, 6, non, 7, right]

Solution Practice #5

- Write a Python program that choose only integers from the following mixed list using continue :
- x = [1, so, ‘2’, too, 3, but, 4, soon, 5, every, 6, non, 7, right]

```python
my_list = [1, "so", "2","too", 3, "but", 4, "soon", 5, "every", 6, 7 , "8"]
int_list=[]
for item in my_list:
    if type(item)!= int:
        continue
    int_list.append(item)
print(int_list)
```

Introduction to Python programming Course Outline

- Intro to Computer Science
- Environment Setup (Anaconda)
- Command Line
- Conda & pip package managers
- Jupyter Notebook
- Input & Output
- Variables
- Data types
- Numbers & Math
- Boolean & Comparison and Logic
- If Conditions
- For & While Loops
- Strings
- Lists
- Tuples
- Sets
- Dictionaries
- Advanced If and Loops
- List Comprehensions
- Dictionary Comprehensions
- **Exceptions**
- File Handling
- Functions
- Built-in functions & Operators (zip, enumerate, range, …)
- Map, Filter, Reduce
- Lambda Expressions

Exceptions

When an error occurs in Python, the whole program crashes and stops execution

Exception handling is a way of handling errors so that the program can overcome them and continue running normally

```python
x = 5
y = 0

try:
  print(x+y)
  print(x-y)
  print(x*y)
  print(x/y) # This will yield a ZeroDivisonError
  print(x**y)
except:
  print("An error occured")

# 5
# 5
# 0
# An error occured
```

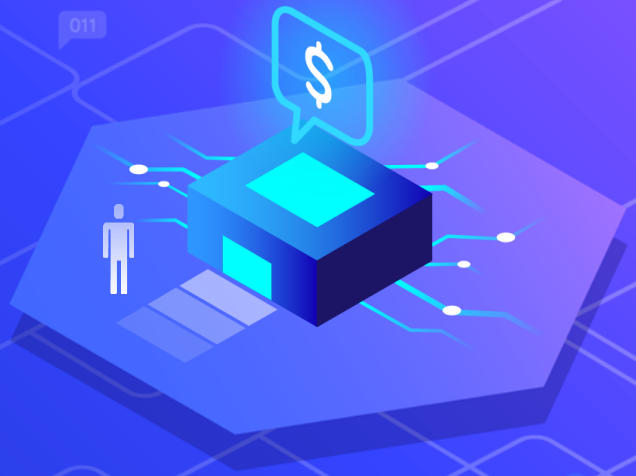
Exceptions

We can make Python check for specific errors

```python
x = 5
y = 0

try:
  print(x+y)
  print(x-y)
  print(x*y)
  print(x/y) # This will yield a ZeroDivisonError
  print(x**y)
except ZeroDivisionError:
  # Error is handled here
  print("Can't divide by zero")
except ValueError:
  print("Encountered value error")

# 5
# 5
# 0
# Can't divide by zero
```



Check Python Error Types

https://docs.python.org/3/library/exceptions.html

Exceptions

Try statements have two extra features:

- ‘else’ : will execute if no errors were caught
- ‘finally’ : will execute whether there were errors caught or not (always execute)

```python
x = 5
y = 2

try:
  print(x+y)
  print(x-y)
  print(x*y)
  print(x/y)
  print(x**y)
except ZeroDivisionError:
  print("Can't divide by zero")
except ValueError:
  print("Encountered value error")
else:
  print("No errors encountered, yay!")
finally:
  print("I will always be executed")

# 7
# 3
# 10
# 2.5
# 25
# No errors encountered, yay!
# I will always be executed
```

Questions ?!