Introduction to Python

- History
- Why Python?
- Scalar Data Types
- Loops, Conditions and Functions

History:

- Developed in late 90's
- Guido van Rossum at Centrum Wiskunde and Informatica (CWI) in the Netherlands
- Python is named after Monty Python Guido is fan of Monty Python’s flying circus, Famous T.V show
- Open sourced from beginning

Why Python?

- Top companies using Python Google, Facebook, Instagram and more
- It's a high-level, interpreted, interactive and object-oriented language. Large standard library support
- Uses an elegant syntax, making the programs you write easier to read
- Internet Scripting

Scalar Data Types:

- Integers
- Floats
- Booleans
- Complex

Integers:

- No limit

Operators	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponent
//	Floor Division

Floats:

- Same as C double!

Input

```
1 x = 12.9
2 y = 3.6
3 print(x * y)
```

Output

```
46.440000000000005
```

Booleans:

- True
- False
- Usual Operators:
  - ==, !=
  - <, , >,
- Additionally,
  - 0 < b < 9
  - 0 < b < c < 12

Complex:

- Uses j for square root of (-1)

Input

```
1 a = 7.1 + 4.3j
2 b = -2.5 + 6.4j
3 c = a + b
4 print(c)
5 print(a * b)
```

Output

```
(4.6+10.7j)
(-45.26999999999996+34.69j)
```

Building Blocks of Python:

- Functions
- Variables
- Expressions
- Statements
- Block
- Comments

Naming Rules:

- Names are case sensitive and cannot start with a number
- Must begin with alphabet (a-z, A-Z) or underscore
- other characters can be alphabets, numbers or underscore
- Python is case sensitive i.e. upper case and lower case are treated distinct

Variables in Python:

- Python is a dynamically typed language
- variables need not be declared in python before using them
- Example: set the variable 'age' equal to the value 10, age = 10
- Now, 'age' can be used in any arithmetic operations, such as age after = age + 20 => 30

Blocks:

- Blocks are indicated by indenting
- Recommended standard is 4 spaces or 1 tab
- Wrong indentation may flag a syntax error

Input

```
1 SENIOR_CITIZEN_ELIGIBILITY = 60
2 my_age = 62
3 if my_age > SENIOR_CITIZEN_ELIGIBILITY:
4     print("Eligible for the rates")
5     discount = 0.5
6 else:
7     print("Grow up some more!")
8 print("Thank you")
```

Output

```
Eligible for the rates
Thank you
```

Blocks Summary:

- Colon at the last character
- Indentation to include in block
- Unindent to close block
- Loops, Conditions, Functions ....

If-elif-else loop:

Input

```
1  x = 7
2  if x > 0:
3      print("Positive")
4  elif x < 0:
5      print("Negative")
6  else:
7      print("Zero!")
```

Output

```
Positive
```

While Loop:

- No parentheses required (as for if)
- Possible to end with infinite loops
- Nestable
- Syntax:
    - while expression:

      statement(s)
- Break, continue are available (not recommended using)

Input

```
a, b, n = 0, 1, 0
while n < 10:
    print(a, end = " ")
    a, b = b, a + b
    n += 1
```

Output

```
0 1 1 2 3 5 8 13 21 34
```

For:

- Iterator not a loop
- Ubiquitous

Input

```
1 f = [0, 1, 1, 2, 3, 5, 8]
2 for n in f:
3     print(n, n * n - 1, end = " ")
```

Output

```
0 -1 1 0 1 0 2 3 3 8 5 24 8 63
```

Functions:

- Piece of reusable code
- Solves particular task
- Call function instead of writing code yourself
- Function block begins with the keyword def followed by function name and parentheses (())

Built-In-Functions:

- Math
  - abs( ), divmod( ), pow( ), round( )
- constructors & converters
  - bin( ), bool( ), chr( ), complex( ), dict( )
  - float( ), hex( ), int( ), list( ), oct( ), ord( )
  - range( ), set( ), str( ), tuple( )
- Aggregate
  - len( ), min( ), max( ), sum( ), any( ), all( )
- and more ...

Writing your own functions:

```
1  def isPrime2(n: int) -> bool:
2      if n in [2, 3, 5, 7]:
3          return True
4      if n % 2 == 0:
5          return False
6      r = 3
7      while r * r <= n:
8          if n % r == 0:
9              return False
10         r += 2
11     return True
12 print(isPrime2(12))
13 print(isPrime2(107))
```

Output

```
False
True
```

Thank You