These are equivalent Python program code built-in functions found in the INSERT for Computer Science 9618/21 and 9618/22.

String and Character Functions

- A string of length 1 may be either of type `CHAR` or `STRING`
- A `CHAR` may be assigned to, or concatenated with, a `STRING`
- A `STRING` of length greater than 1 cannot be assigned to a `CHAR`

```
#ThisString : STRING
ThisString[0:x]
```
returns leftmost `x` characters from `ThisString.`

Example:
```
ThisString = "ABCDEFGH"
print(ThisString[0:3])
```
returns “`ABC`”

```
#ThisString : STRING
ThisString[-x:]
```
returns rightmost `x` characters from `ThisString.`

Example:
```
ThisString = "ABCDEFGH"
print(ThisString[-3:])
```
returns “`FGH`”

```
#ThisString : STRING
ThisString[start:end:step]
```
returns the characters from the `start` value till `end - 1` and the number of steps taken.

Example:
```
ThisString = "ABCDEFGH"
print(ThisString[1:4])
```
returns “`BCD`”

```
#ThisString : STRING
len(ThisString)
```
returns the integer value representing the length of `ThisString.`

Example:
```
ThisString = "Happy Days"
print(len(ThisString))
```
returns `10`

```
#x : STRING
x.upper()
```
returns a string formed by converting all characters of `x` to upper case.

Example:
```
x = "Error 803"
print(x.upper())
```
returns “`ERROR 803`”

```
#x : STRING
x.lower()
```
returns a string formed by converting all characters of `x` to lower case.
```
x = "JIM 803"
print(x.lower())
```
returns “`jim 803`”

```
#x : INTEGER
str(x)
```
returns a string representation of a numeric value.

Example:
```
x = 87.5
print(str(x))
```
returns “`87.5`”

```
#ThisString : STRING
y = ThisString.isnumeric()
```
returns `TRUE` if ThisString represents a valid integer value.

```
ThisString = "-12.36"
print(ThisString.isnumeric())
```
returns `False`

```
#ThisString : STRING
y = ThisString.isdigit()
```
returns `TRUE` if ThisString represents a valid digit value.

Example:
```
ThisString = "12"
print(ThisString.isdigit())
```
returns `True`

```
#ThisString : STRING
y = ThisString.isdecimal()
```
returns `TRUE` if all the values of `ThisString` are between 0 to 9.

Example:
```
ThisString = "12"
print(ThisString.isdecimal())
```
returns `True`

```
ord(ThisChar)
```
returns an integer value (the ASCII value) of character `ThisChar`.

Example:
```
print(ord("A"))
```
returns `65`

```
chr(x)
```
returns the character whose integer value (the ASCII value) is `x`
```
print(chr(65))
```
returns “`A`”

Numeric Functions

```
int(x)
```
returns the integer part of x

Example:
```
print(int(27.5415))
```
returns `27`

```
import random
x = random.uniform(start, end)
```
returns a real number in the range `start` to `end` (not inclusive of `end`).

Example:
```
import random
print(random.uniform(0, 87))
```
may return `35.43`

```
import random
x = random.randint(start, end)
```
returns a integer number in the range `start` to `end` (not inclusive of `end`).
```
import random
print(random.randint(20, 30))
```
may return `22`

Date Functions

Date format is assumed to be YYYY/MM/DD unless otherwise stated.

```
import datetime
x = datetime.datetime(y, m, d)
```
stores the date in `x` in the format YYYY/MM/DD.

```
import datetime
x = datetime.datetime.now()
```
returns the date of the current day.

`ThisDate.month` returns the month from ThisDate
`ThisDate.date` returns the current day number from `ThisDate`

`ThisDate.strftime(“%w”)`
returns the day index number from `ThisDate` where Sunday = 0, Monday = 1 etc.
Example:
`import datetime`
`x = datetime.datetime(2024,5,23)`
`print(x.strftime("%w"))`
returns `4`

`ThisDate.strftime(“%A”)`
returns the day from `ThisDate.`
Example:
`import datetime`
`x = datetime.datetime(2024,5,23)`
`print(x.strftime("%A"))`
returns “`THURSDAY`”

Operators

An error will be generated if an operator is used with a value or values of an incorrect type.

`//`	finds the quotient when one number is divided by another (DIV). Example `10 // 3` evaluates to `3`
%	finds the remainder when one number is divided by another (MOD). Example: `10 % 3` evaluates to `1`