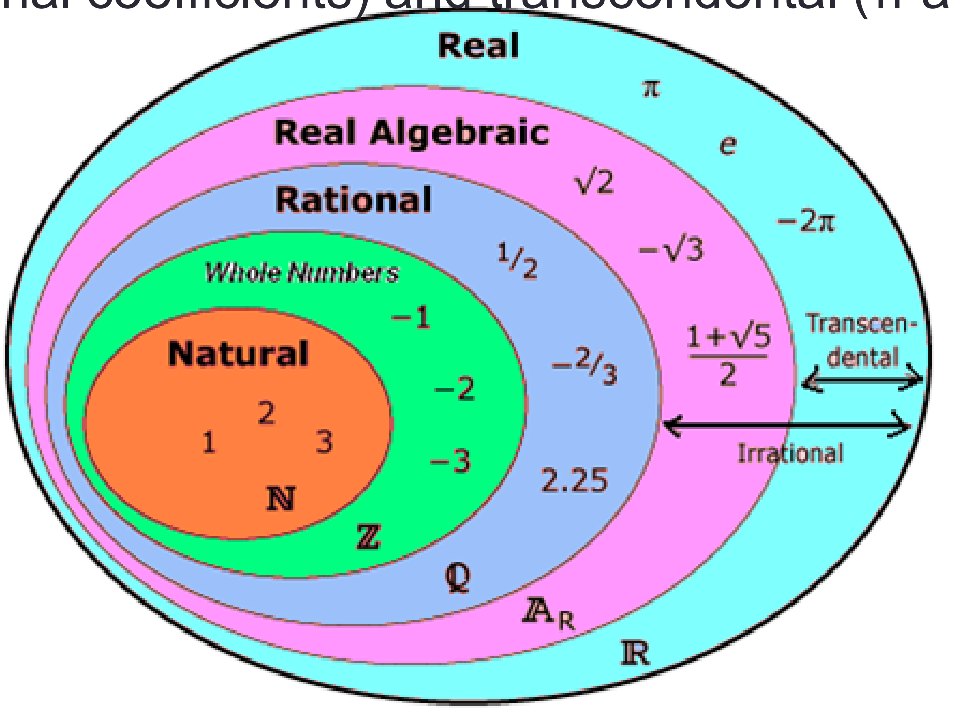
PYTHON: BASIC DATA TYPES

Objectives

- To understand how numbers and characters are represented in computers
- To understand and use operators for numbers
- To understand arithmetic expression
- To understand data type and use type conversion
- To be able to read and write programs that process numerical data.

Real number system

- Recall from your math class, real numbers consist of rational numbers and irrational numbers.
  - A rational number can be represented as i/j for integers i and j.
  - Irrational numbers can be further divided into algebraic (real root of polynomials with rational coefficients) and transcendental (π and e).



Further reading:
https://en.wikipedia.org/wiki/Real_number
https://en.wikipedia.org/wiki/Algebraic_number

Numeric data types

- Computers “simulate” the real number system.
- Two numeric data types:
  - Integer (`int`), e.g., 10, 0, -9999
  - Floating-point number (`float`), e.g., 1.1, 0., -3333.33
- `int` and `float` are two different *data types*.
- A floating-point number can be represented by including an exponent component, e.g., -3.33333×10^3 (try to type -3.33333e3 in Python and see the output)
- Inside the computer, integers and floating point are represented quite differently.
  - Negative integer is usually represented in two's complement (to be covered elsewhere).

EXERCISE 2.1

- Enter a very large integer in your IDLE and see whether the returned value is the same as the entered value.
- Repeat above with a very large floating-point number.
- Is 3.3333333333e3 or 3.33e33 larger?

Rounding

- The displayed value can be rounded (sometimes truncated).
- Several related functions:
  - `round(x,n)` built-in function – round to n decimal places
  - math function – round up
  - `math.flomath.ceil(x) or(x)` math function – round down
- To make use of math.*** functions, you need to
  - `import math`

EXERCISE 2.2

- Try `round(0.45,1)`, `round(1.45,1)`, `round(2.45,1)`, `round(3.45,1)`, …, `round(9.45,1)`. Do you observe any patterns?
- Try `math.ceil(5.45)` and `math.floor(5.45)`.
- Try `math.ceil(-5.45)` and `math.floor(-5.45)`.
- Try `int(5.45)`, `int(-5.45)` and `float(5)`.

String

- Strings in Python can be expressed inside double quotes or single quotes.
  - A string can be empty.
- Strings in Python are represented by UTF-8 using 8 to 32 bits to represent a character.
  - The 8-bit representation is the same as the ASCII (American Standard Code for Information Interchange).
- The **`ord`** function returns the numeric (ordinal) code of a single character, e.g., `ord('A')` is 65.
- The **`chr`** function converts a numeric code to the corresponding character, e.g., `chr(65)` is 'A'.

ASCII table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

EXERCISE 2.3

Try the followings:

- `print('')  # two single quotes without space`
- `print("") # two double quotes without space`
- `print(" ' ")`
- `print(' " ')`
- `print("') # double quote + single quote`
- `ord('') # two single quotes without space`
- `ord(' ') # space inside`
- `ord("") # two double quotes without space`
- `ord(" ") # space inside`

Control characters

- Control characters are special characters that are not displayed on the screen, and they control the display of output (among other things).
- An escape sequence begins with an escape character (\) that causes the sequence of characters following it to "escape" their normal meaning.
- Escape sequences are strings.
- Some useful ones:
  - \' single quote
  - \" double quote
  - \t tab
  - \n give a newline
  - \\ give the backslash itself
  - \ooo gives the ASCII character represented by ooo_{oct}, e.g. "\063" = "3".
  - \xhh gives the ASCII character represented by hh_{hex}, e.g. "\x41" = "A".
  - \uhhhh gives Unicode character represented by $hhhh_{hex}$, e.g. "\u2190" = "←", "\u5927" = "大", "\u3042" = "あ", "\u3184" = "ㆄ".

EXERCISE 2.4

Try

- `print("1\t2\t3")`
- `print("1\n2\n3")`
- `print("\"")`
- `print("\\")`
- `print("\")`
- `print("\u5927")`

Assignment statements

- Simple assignment: `<variable> = <expr>`
  variable is an identifier, expr is an expression
- The expression on the RHS (right hand side) is evaluated to produce a value which is then associated with the variable named on the LHS (left hand side).

EXERCISE 2.5

Ask users to input two numbers and print out the two numbers in a reversed order.

Simultaneous Assignment

- Several values can be calculated at the same time.
- `<var>, <var>, … = <expr>, <expr>, …`
- Evaluate the expressions in the RHS and assign them to the variables on the LHS.
- E.g., `x, y = y, x`
- E.g., `sum, diff = x+y, x-y`
- E.g., `x, y = eval(input("Input the first and second numbers separated by a comma: "))`

EXERCISE 2.6

Simplify your codes in exercise 2.5 using simultaneous assignment statements.

Expressions

- The fragments of code that produce or calculate new data values are called *expressions*.
- A (numeric/string) *literal,* which is the simplest kind of expression, is used to represent a specific value, e.g. 10 or “Mickey”.
  - A simple identifier can also be an expression.
- Simpler expressions can be combined using *operators* `+`, `-`, `*`, `/`, and `**` (special operator // for integer division).
  - The normal mathematical precedence applies.
  - Only round parentheses can be used to change the precedence, e.g., `((x1 - x2) / 2*n) + (spam / k**3)`.
- Try `print("I" + "love" + "Mickey")`.
- Try `print("I","love","Mickey")`.

Python built-in numeric operations

Operation	Result
`x + y`	sum of *x* and *y*
`x - y`	difference of *x* and *y*
`x * y`	product of *x* and *y*
`x / y`	quotient of *x* and *y*
`x // y`	floored quotient of *x* and *y*
`x % y`	remainder of `x / y`
`-x`	*x* negated
`+x`	*x* unchanged
`abs(x)`	absolute value or magnitude of *x*
`int(x)`	*x* converted to integer
`float(x)`	*x* converted to floating point
`complex(re, im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.
`c.conjugate()`	conjugate of the complex number *c*
`divmod(x, y)`	the pair `(x // y, x % y)`
`pow(x, y)`	*x* to the power *y*
`x ** y`	*x* to the power *y*

$$\begin{array}{r} 2 \\ 4\overline{)11} \\ \underline{8} \\ 3 \end{array}$$

Source: Charles Dierbach. 2013. Introduction to Computer Science Using Python. Wiley.

Operator precedence and associativity

- The operators ** and – (negation) have higher precedence than the four operators (+, –, *, /).
- For operators of the same precedence, the associativity determines the order of their operations.

Operator	Associativity
`**` (exponentiation)	right-to-left
`-` (negation)	left-to-right
`*` (mult), `/` (div), `//` (truncating div), `%` (modulo)	left-to-right
`+` (addition), `-` (subtraction)	left-to-right

Source: Charles Dierbach. 2013. Introduction to Computer Science Using Python. Wiley.

EXERCISE 2.7

Try the followings:

- `2**3**4`
- `8+4-2`
- `8-4-2`
- `8*4/3`
- `8/4/2`

$$2^{(3^4)} \qquad (2^3)^4$$

Data types

- A data type is a set of values, and a set of operators that may be applied to those values.
  - Integer, floating-point numbers and string are built-in types in Python.
- An internal representation could have different meanings:
  - $01000001_{\textbf{bin}}$ can be interpreted as "A" (ASCII) or $65_{\textbf{dec}}$.
- Each literal or variable is associated with a data type (`int` and `float` for now).
- A **`type`**`(x)` function returns the data type of `x` which could be a literal or variable.
- Explicit type conversion
  - Built-in functions **`int`**`(x)` and **`float`**`(x)`.

EXERCISE 2.8

- Try out the `type()` function for both numeric and string literals and variables.
- Assign 10 to `x` and find out the type of `x`, and assign 10.0 to `x` and find out its type.
- Try to use `int()` instead of `eval()` to get your age, and explore with different inputs.

EXERCISE 2.9

What are the data types of the following arithmetic expressions: 6+3, 6.0+3.0, 6.0+3, 6.00+3.00, 6*3, 6.0*3.0, 6.0*3, 6/3, 6.0/3.0, 6.0/3?

Try to answer yourself before asking Python for the answer.

EXERCISE 2.10

- Try the following
  - `int(11.1)`
  - `int("11")`
  - `int("11.1")`
  - `float(11)`
  - `float("11")`
  - `float("11.1")`
  - `float("1.1111111111111111")`

How can you get answer to int("11.1"), i.e., 11?

END

References

- A Tutorial on Data Representation: Integers, Floating-point Numbers, and Characters: https://www3.ntu.edu.sg/home/ehchua/programming/java/DataRepresentation.html