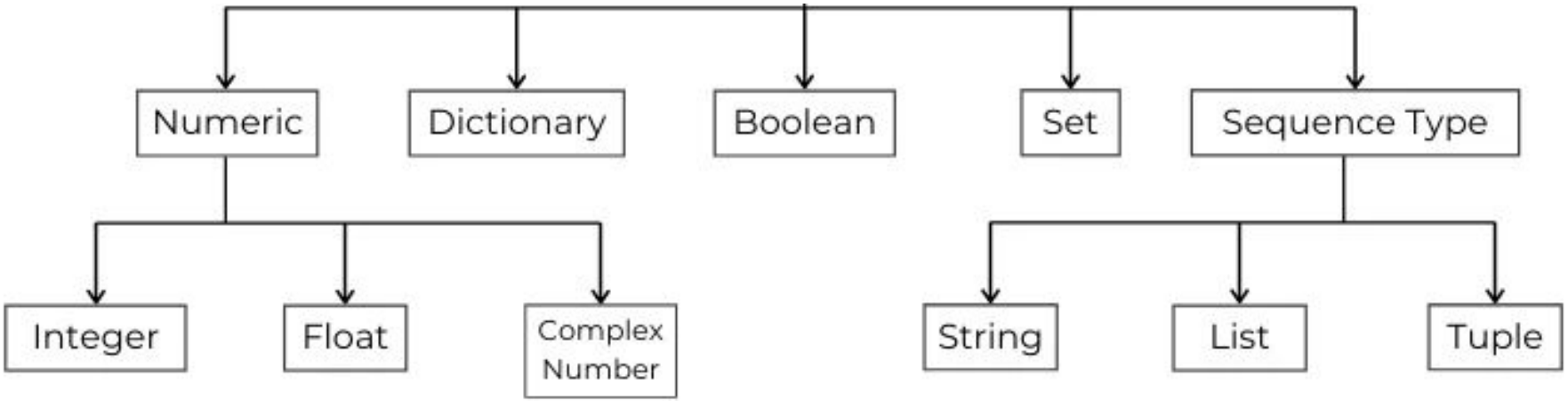
1.1 Elementära datatyper

Holger Rosencrantz

1.1a Intro och numeriska datatyper

Datatyper

- En variabels datatyp definierar hur mycket minne variabelns värde tar upp och hur värdet är formaterat
- Viktigaste datatyper i Python sammanfattas i bilden nedan



Numeriska datatyper: heltal

- Datorn sparar tal i det binära talsystemet (en krets på = 1, en krets av = 0)
- Vanligen lagras heltal i 32 bitar = 4 byte
- Exempel: talet 1:
  `00000000 00000000 00000000 00000001`
- Exempel: talet 2.147.483.648:
  `01111111 11111111 11111111 11111111`
- Det första talet representerar tecknet (positiv/negativ), med 4 byte kan vi alltså representera tal mellan -2.147.483.648 och 2.147.483.648
- I Python motsvaras heltal av datatypen `int`

Exempel heltal i Python

```
x = 5           # heltal
y = 3           # ett till heltal
print(x+y)      # 8
print(x-y)      # 2
print(x*y)      # 15
print(x/y)      # 1.6666…
print(x // y)      # 1 (heltalsdivision)
print(x % y)       # 2 (restdivision)
print(x**y)     # 125, dvs. 5 upphöjt till 3
print(-x)       # -5
print(+x)       # 5
print(abs(y-x))   # 2
```

Numeriska datatyper: flyttal

- Flyttal är ett sätt att representera decimaltal
- Flyttal representeras som en bas och en exponent (fortfarande i binär form)
- Lite mer komplicerat än heltal, du behöver inte förstå exakt hur det går till
- Flyttal kan vara lite luriga, eftersom de avrundar decimaltal till närmaste halva, fjärdedel, åttondel osv. istället för närmaste tiondel, hundradel, tusendel osv. - det kan därför bli problem med exaktheten, men detta har antagligen ingen praktisk betydelse i denna kurs
- I Python motsvaras flyttal av datatypen `double` och `float`

Numeriska datatyper: komplexa tal

- Finns bland Pythons grundläggande typer, men ni som inte har läst Matematik 4 behöver inte bekymra er särskilt mycket

Exempel flyttal och komplexa tal i Python

```
print(float(x))   # 5.0
print(int(3.14))  # 3
print(int("3"))   # 3
z = complex(1, 2) # komplext tal
print(z)      # (1+2j)
print(z.conjugate())    # (1-2j)
print(z*z)    # (-3+4j)
```

Ta reda på vad en datatyp är

```
x = 1

print(type(x)) # <class 'int'>
```

Python är ett dynamiskt typat språk

```
x = "1"
print(type(x))  # str
x = 1
print(type(x))  # int
```

VARNING FÖR ÖVERRASKNINGAR:

```
def double(x):
    print(x+x)
double(1)   # 2
double("1") # 11
```

Idag: Övningar numeriska datatyper

1. En googol är en etta följt av 100 nollor, alltså 10^{100}. Vad är klockan om en googol minuter? (Bortse från kosmologiska begränsningar som att universum inte längre skulle existera eller tid inte längre vara ett meningsfullt begrepp.)
2. En googolplex är en etta följt av en googol nollor. Försök inte räkna med ett så stort tal, för då svämmar minnet över eller så fortsätter programmet i all evighet och måste avbrytas!
3. Kan du skapa flyttal från strängar på liknande sätt som exemplet int("3") ovan?
4. Eulers identitet implicerar att $e^{i\pi}=1$. Kontrollera att detta stämmer (använd e=2,72 och π=3,14)! Vad blir 2^{1+i}?
5. Fungerar heltalsdivision och restdivision med komplexa tal i Python?

Jobba i övrigt med valfria uppgifter på Kattis idag

Försök gärna att pusha ditt arbete i slutet av lektionen till GitHub

1.1b Övriga elementära datatyper: sekvenser m.m.

Sekvenser: strängar, fält och tipplar

- Ett fält (eller en lista) är en datamängd med ett antal element i en viss ordning
- En tippel (t.ex. ett par eller en trippel) liknar ett fält, men kan inte ändras när det väl har skapats
- En sträng kan indexeras på liknande sätt som en tippel, t.ex.

```
print("Hej på dig!"[2]) # "j"
```

- Syntax i Python:
  - strängar med citattecken
  - fält med hakparenteser [,]
  - tipplar med parentestecken (,)

Exempel: sekvenser i Python

```
s = "Hejsan på dejsan!"
print(len(s))                # 17
print(s[0])                  # "H"
print(s[10])                 # "d"
print(s.index("dejsan"))     # 10
print(s.upper())             # "HEJSAN PÅ DEJSAN"
tuple = s.partition("på")
print(tuple)                 # ('Hejsan' ,'på', 'dejsan!')
print(tuple[2])              # "dejsan"
tuple[2] = "mejsan!"         # fel - kan inte ändra en tuppel!
```

Övriga datatyper: associativt fält (dictionary)

Ett associativt fält förknippar (associerar) ett värde till vart och ett av en uppsättning nycklar

```
dic = {"Sverige": "Stockholm", "Norge": "Oslo", "Finland": "Helsingfors"}
print(dic["Sverige"])               # "Stockholm"
print(dic["Norge"])                 # "Oslo"
print(dic["Finland"])               # "Helsingfors"
print(dic["Danmark"])               # KeyError
```

Oftast är nyckeln en textsträng, som t.ex. "Sverige", "Norge" och "Finland" ovan, men det behöver inte nödvändigtvis vara så.

```
dic = {1: "Foo", "Bar": "Baz", True: False}
print(dic[1])                       # "Foo"
print(dic["Bar"])                   # "Baz"
print(dic[True])                    # False
```

Användbara metoder:

```
dic.update({"Estland": "Tallinn"})  # lägga till ett nyckel-värdepar
dic.pop("Sverige")                  # ta bort nyckel-värdeparet med nyckeln "Sverige"
```

Övriga datatyper: Boolean

Antingen sann eller falsk

I likhet med i t.ex. JavaScript så är vissa värden (t.ex. tom sträng eller siffran noll) ”falsy”, dvs. de värderas falska i logiska sammanhang, medan andra är ”truthy”

```
bool = True or 0
print(bool)         # True
bool = True and 1
print(bool)         # 1
bool = not False or ""
print(bool)         # True
bool = False and ""
print(bool)         # False
bool = True and ""
print(bool)         # tom sträng
```

Övriga datatyper: mängd (set)

Endast ett av varje element, ingen särskild ordning

Lägg till ett element med metoden `add`

Kan vara användbar när man vill försäkra sig om att varje element i en datasamling är unikt, dvs. inga dubletter

```
s = set()
s.add("Äpple")
s.add("Banan")
s.add("Päron")
s.add("Äpple")      # Vi har redan ett äpple
print(s)            # {"Banan", "Päron", "Äpple"}
```

Idag: Övningar sekvenser, dictionary, boolean och set

Studera fler exempel från w3schools enligt länkar i lektionsanteckningarna.

1. Skriv ett program som översätter från svenska till rövarspråket.
2. Skriv ett program som skriver en inmatad text baklänges.
3. Utgå från det associativa fältet med länder och huvudstäder ovan. Använd metoden update för att lägga till nyckel-värdeparet {"Danmark": "Köpenhamn"}. Använd sedan metoden pop för att ta bort nyckel-värdeparet {"Finland": "Helsingfors"}.
4. Använd metoden union för att lägga ihop de två mängderna {"Banan", "Päron", "Äpple"} och {"Kiwi", "Ananas", "Päron"}. Hur många element finns i den resulterande mängden?

Jobba i övrigt med valfria uppgifter på Kattis idag

Pusha ditt arbete i slutet av lektionen till GitHub (be om hjälp ifall du behöver)