# اعداد در پایتون

در این جلسه، درباره اعداد در پایتون و نحوه استفاده از آنها آموخته خواهیم شد.

در اینجا در مورد موضوعات زیر آموزش خواهیم دید:

1. انواع اعداد در پایتون
2. حساب اعمال ابتدایی
3. تفاوت بین تقسیم کلاسیک و تقسیم به طور قسمتی
4. اختصاص شی در پایتون

# انواع اعداد

پایتون دارای "انواع" مختلفی از اعداد (حروف عددی) است. اصلی‌ترین نوع‌ها اعداد صحیح و اعداد اعشاری (اعداد ممیز شناور) هستند.

اعداد صحیح فقط اعداد صحیح است، مثبت یا منفی. به عنوان مثال: 2 و -2 نمونه‌هایی از اعداد صحیح هستند.

اعداد اعشاری در پایتون به دلیل داشتن علامت اعشاری یا استفاده از نمایش عددی علمی (e) قابل توجه هستند. به عنوان مثال، 2.0 و -2.1 نمونه‌هایی از اعداد اعشاری هستند. همچنین، 4E2 (4 ضرب در 10 به توان 2) نیز نمونه‌ای از عدد اعشاری در پایتون است.

در طول این دوره، اصلی‌ترین کار ما با اعداد صحیح و اعداد اعشاری ساده خواهد بود.

در زیر جدولی از دو نوع اصلی که بیشترین زمان خود را برای کار با آنها صرف خواهیم کرد همراه با برخی مثال‌ها آمده است:

| نوع عددی | مثال |
|---|---|
| Integers | 1,2,-5,1000 |
| Floating-point numbers | 1.2,-0.5,2e2,3E2 |

در ادامه یکسری مثال های مربوط به محاسبات ساده ریاضی را با هم مشاهده کنیم.

### ریاضیات پایه

In [5]:
```
# Addition
2+1
```

Out[5]: 3

In [6]:
```
# Subtraction
2-1
```

Out[6]: 1

In [7]:
```
# Multiplication
2*2
```

Out[7]: 4

In [8]:
```
# Division
3/2
```

Out[8]: 1.5

In [9]:
```
# Floor Division
7//4
```

Out[9]: 1

**عجب! چه اتفاقی افتاد؟ آخرین بار که چک کردم، تقسیم ۷ بر ۴ برابر ۱.۷۵ بود نه ۱!**

دلیل این نتیجه این است که ما از تقسیم "*بدون تقریب*" استفاده می‌کنیم. عامل // (دو خط کشیده به سمت جلو) اعشاری را بدون تقریب قطع می‌کند و نتیجه‌ی یک عدد صحیح را برمی‌گرداند.

**خب اگه فقط باقیمانده تقسیم رو بخواهیم چی؟**

In [10]:
```
# Modulo
7%4
```

Out[10]: 3

۴ یکبار در ۷ وجود دارد بنابراین باقیمانده‌ای معادل ۳. عملگر % باقیمانده را پس از تقسیم برمی‌گرداند.

## ادامه اعمال ریاضی

In [11]:
```
# Powers
2**3
```

Out[11]: 8

In [12]:
```
# Can also do roots this way
4**0.5
```

Out[12]: 2.0

In [13]:
```
# Order of Operations followed in Python
2 + 10 * 10 + 3
```

Out[13]: 105

In [14]:
```
# Can use parentheses to specify orders
(2+10) * (10+3)
```

Out[14]: 156

## انتساب به متغییرها

حالا که دیدیم چگونه می‌توانیم از اعداد به عنوان یک ماشین حساب در پایتون استفاده کنیم، بیایید ببینیم چگونه می‌توانیم نام‌ها را به متغیرها اختصاص دهیم و متغیرها ایجاد کنیم.

ما از یک علامت تک مساوی برای اختصاص برچسب به متغیرها استفاده می‌کنیم. چند نمونه از این کار را بیایید ببینیم.

In [15]:
```
# Let's create an object called "a" and assign it the number 5
a = 5
```

لطفا توجه کنید که در صورتی که *a* را در اسکریپت پایتون خود فراخوانی کنید، پایتون آن را به عنوان عدد ۵ خواهد در نظر گرفت.

In [16]:
```
# Adding the objects
a+a
```

Out[16]: 10

در صورت انتساب مجدد متغیر چه اتفاقی می‌افتد؟ آیا پایتون به ما اجازه می‌دهد آن را دوباره نوشته شود؟

In [17]:
```
# Reassignment
a = 10
```

In [18]:
```
# Check
a
```

Out[18]: 10

بله! پایتون به شما اجازه می‌دهد مقادیر متغیر را مجدداً با مقدار جدید بازنویسی کنید. همچنین، می‌توانید همان متغیرها را در هنگام انجام بازنویسی استفاده کنید. در ادامه، یک مثال برای توضیح این موضوع آورده شده است:

In [19]:
```
# Check
a
```

Out[19]: 10

In [20]:
```
# Use A to redefine A
a = a + a
```

In [21]:
```
# Check
a
```

Out[21]: 20

نام‌هایی که هنگام ایجاد این برچسب‌ها استفاده می‌کنید، باید اصول خاصی را رعایت کنند:

1. نام‌ها نمی‌توانند با یک عدد شروع شوند.
2. نمی‌توانید در نام‌ها از فاصله استفاده کنید؛ به جای آن از خط تیره (_) استفاده کنید.
3. نمی‌توانید از هیچ یک از علائم زیر استفاده کنید: :"',<>/?|()!@#$%^&*~-+
4. توصیه می‌شود (براساس PEP8) نام‌ها با حروف کوچک نوشته شوند.
5. از استفاده از کاراکترهای 'l' (حرف کوچک ال)، 'O' (حرف بزرگ او) یا 'I' (حرف بزرگ آی) به عنوان نام متغیر با یک حرف اجتناب کنید.
6. از استفاده از کلماتی که در پایتون معنای خاصی دارند مانند "list" و "str" خودداری کنید.

در برنامه نویسی روش های متنوعی برای نام گذاری متغییرهای چند کلمه ای مرسوم می باشد:

1. روش پاسکال
2. روش camel case

3. روش snake case ==> (python)
4. روش kebab case

In [ ]:
```
MyFamilyHouse      # Pascal Case

myFamilyHouse      # Camel Case

my_family_house    # Snake Case

my-family-hous     # Kebab Case ==> invalid in python
```

استفاده از نام‌های متغیر می‌تواند روشی بسیار مفید برای پیگیری متغیرهای مختلف در پایتون باشد. به عنوان مثال:

In [22]:
```
# Use object names to keep better track of what's going on in your code!
my_income = 100

tax_rate = 0.1

my_taxes = my_income*tax_rate
```

In [23]:
```
# Show my taxes!
my_taxes
```

Out[23]: 10.0

## پویایی نوع داده‌ها (Dynamic Typing)

پایتون از *پویایی نوع داده‌ها* استفاده می‌کند، به این معنی که شما می‌توانید متغیرها را به نوع داده‌های مختلف دوباره اختصاص دهید. این ویژگی باعث می‌شود پایتون در اختصاص داده‌های نوع، بسیار انعطاف‌پذیر باشد؛ این در مقایسه با زبان‌های دیگری که *نوع‌های استاتیک* دارند، تفاوت دارد.

لطفاً به زبان انگلیسی بنویسید.

In [24]:
```
my_dogs = 2
```

In [25]:
```
my_dogs
```

Out[25]: 2

In [26]:
```
my_dogs = ['Sammy', 'Frankie']
```

In [27]:
```
my_dogs
```

Out[27]: ['Sammy', 'Frankie']

### مزایا و معایب پویایی نوع داده‌ها (Dynamic Typing)

### مزایا پویایی نوع داده‌ها

1. بسیار آسان برای کار کردن
2. زمان توسعه سریعتر

### معایب پویایی نوع داده‌ها

1. ممکن است باعث بروز خطاهای غیرمنتظره شود!
2. باید از type() آگاه باشید
3. کند احتمالی اجرا نسبت به زبان های strict

## چک کردن نوع داده با استفاده از تابع type()

می‌توانید با استفاده از تابع داخلی type() در پایتون، نوع شیءی که به یک متغیر اختصاص داده شده است را بررسی کنید. نوع‌های داده‌های متداول عبارتند از:

- **int** (for integer)
- **float**
- **str** (for string)
- **list**
- **tuple**
- **dict** (for dictionary)
- **set**
- **bool** (for Boolean True/False)

In [28]:
```
1 type(a)
```

Out[28]:
```
int
```

In [29]:
```
1 a = (1,2)
```

In [30]:
```
1 type(a)
```

Out[30]:
```
tuple
```

در نتیجه، چه چیزهایی یاد گرفتیم؟ ما به برخی از مبانی اعداد در پایتون پرداختیم. همچنین یاد گرفتیم چگونه عملیات حسابی را انجام داده و از پایتون به عنوان یک ماشین حساب ساده استفاده کنیم. همچنین، با مفهوم اختصاص متغیر در پایتون آشنا شدیم.

در ادامه، به مبحث رشته‌ها (Strings) خواهیم پرداخت!